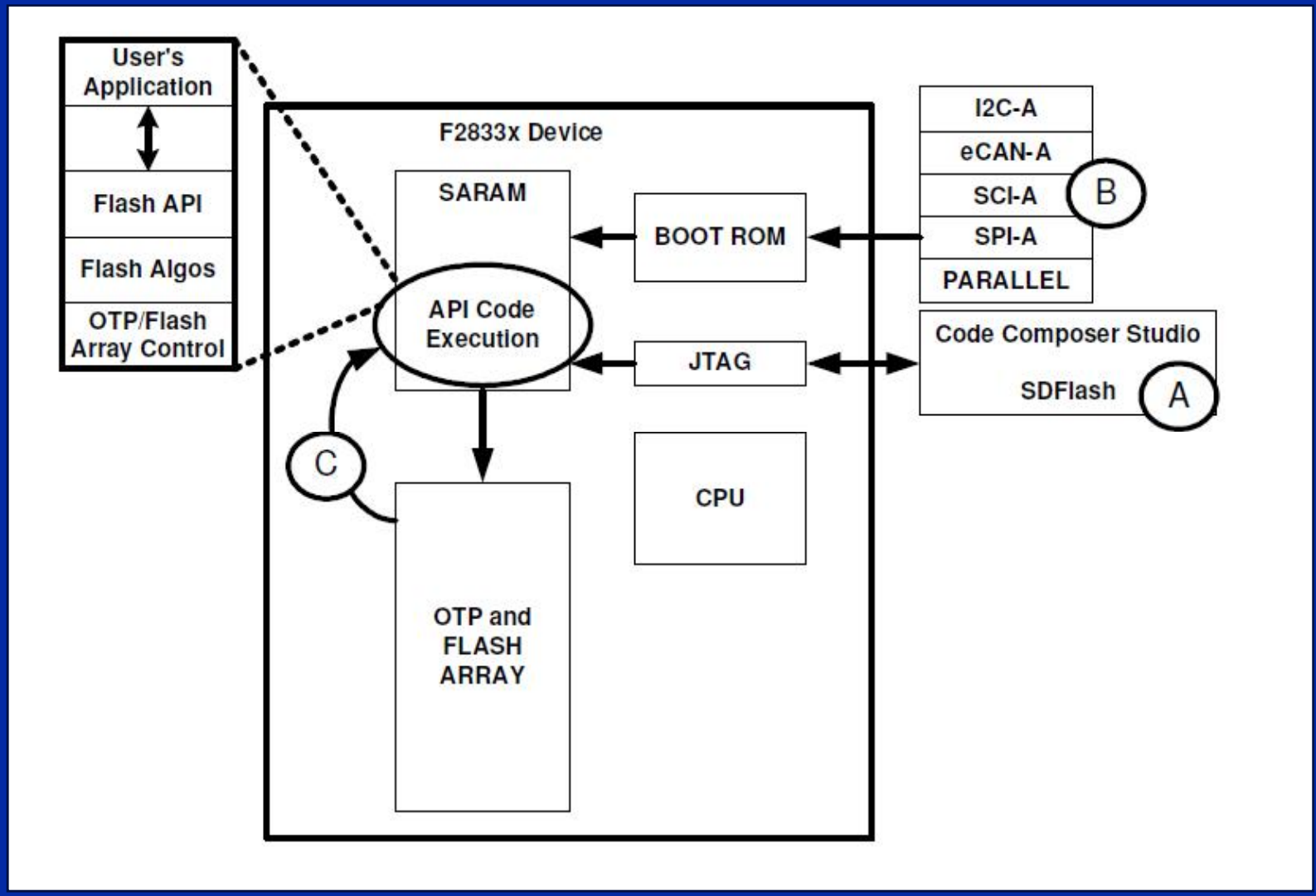
Modul 16: FLASH Memory API

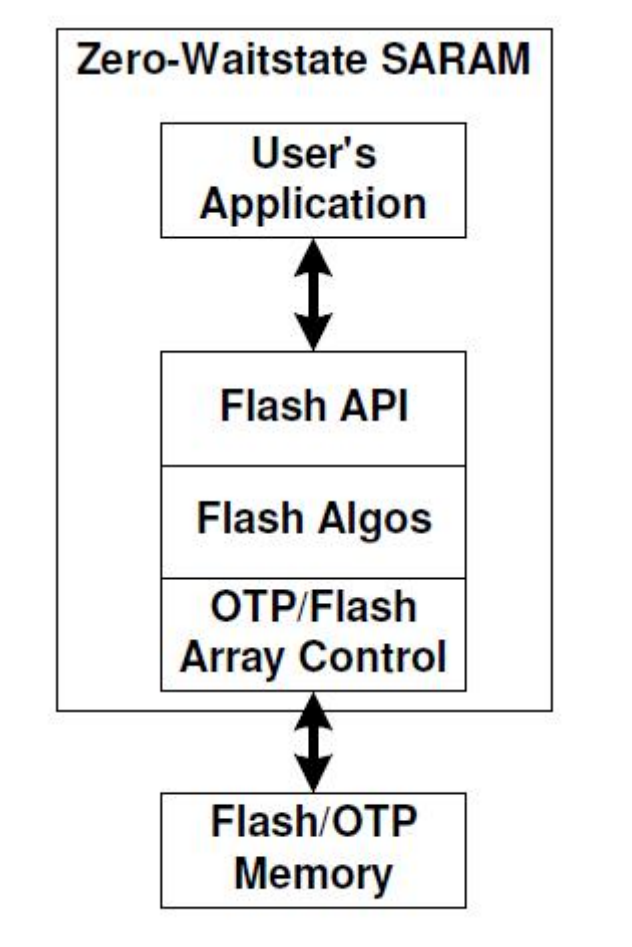
32-Bit-Digital Signal Controller TMS320F2833x

Texas Instruments Incorporated

TMS320F2833x FLASH Load Options



FLASH – API - Instalacija



1. Download from: www.ti.com:
   - F2833x: “SPRC539”
   - F2823x: “SPRC665”
2. Default Installation path:
   - C:\tidcs\c28\FLASH28_API\
3. Read included documentation
   - “Flash2833x_API_Readme.pdf”

FLASH – API – Osnove

Erase:

- **Pre - Compact**
  - osiguranje da bitovi nisu u “depleted” stanju
- **Pre – Condition**
  - obriši sve bitove na ‘0’ u cilju ujednačenog brisanja
- **Erase**
  - spostavi sve bitove na ‘1’ (= Erased state)
- **Post – Conditioning**
  - osiguranje da bitovi nisu u “depleted” stanju

Program:

- **programiranje selektovanih bitova sa ‘1’ na ‘0’**

Verify:

- **CPU čita i upoređuje FLASH i *image***

Integracija FLASH – API u program

Priprema programa:

1. Modifikacija “Flash2833x_API_Config.h”
2. “Include” Flash2833x_API_Library.h u izvorni kod
3. Dodati FLASH-API – biblioteku u project

Modifikacija programa:

4. Initializacija PLL i čekanje na ,,lock,,
5. Provera da PLL nije u “limp” – režimu
6. Kopiranje svih API – funkcija iz FLASH-a u SARAM
7. Inicijalizacija globalne prom. “Flash_CPUScaleFactor”
8. Inicijalizacija callback – pointera “Flash_CallbackPtr”
9. Poziv API - funkcije

1. Modifikacija “Flash2833x_API_Config.h”

Skloniti komentar sa odgovarajuće linije

```
/*-----------------------------------------------------------------
* Flash2833x_API_Config.h
*-----------------------------------------------------------------*/
…
//#define CPU_RATE 6.667L // for a 150MHz CPU clock speed (SYSCLKOUT)
#define CPU_RATE 8.000L // for a 125MHz CPU clock speed (SYSCLKOUT)
//#define CPU_RATE 10.000L // for a 100MHz CPU clock speed (SYSCLKOUT)
//#define CPU_RATE 13.330L // for a 75MHz CPU clock speed (SYSCLKOUT)
…
…
…
…

/*-----------------------------------------------------------------
* Flash2833x_API_Config.h
*-----------------------------------------------------------------*/
…
#define SCALE_FACTOR 1048576.0L*( (200L/CPU_RATE) )
…
```

2: “Include” Flash28335_API_Library.h

```
#include "FLASH28335_API_Library.h"
```

Dodati *path* do ovog fajla u *build* opcije za pretraživanje

```
<>\Flash28_API\Flash28335_API_V210\include
```

3: Linkovati odgovarajuću Flash API biblioteku u projekt

```
<>\Flash28_API\Flash28335_API_V210\lib\Flash28335_API_V210.lib
```

4: Inicijalizacija PLL Control registra (PLLCR)

Provera korektne taktne frekvencije
Dovoljno vremena za “hvatanje” PLL-a

“DSP2833x_SysCtrl.c”

5: Provera statusa PLL-a / Limp režim

Kolo se resetuje i postavlja bit PLLSTS u registru MCLKSTS

6: Kopiranje Flash API funkcja u SARAM

Ograničenja pri izvršavanju programa iz Flash-a
- samo jedna operacija u jednom trenutku
- precizan tajming

```
/*---------------------------------------------------------------
* User's .cmd file: TMS320F28335 API Library Group Section Example
*---------------------------------------------------------------*/
...
   SECTIONS
   {
        Flash28_API:
        {
                -lFlash28335_API_V210.lib(.econst)
                -lFlash28335_API_V210.lib(.text)
        }               LOAD = FLASHD,
                        RUN = RAML0,
                        LOAD_START(_Flash28_API_LoadStart),
                        LOAD_END(_Flash28_API_LoadEnd),
                        RUN_START(_Flash28_API_RunStart),
                        PAGE = 0
...
}
```

6: Kopiranje Flash API funkcja u SARAM

```
/*-------------------------------------------------------------------
* Flash2833x_API_Library.h
*-------------------------------------------------------------------*/
...
extern Uint16 Flash28_API_LoadStart;
extern Uint16 Flash28_API_LoadEnd;
extern Uint16 Flash28_API_RunStart;
...
/*-------------------------------------------------------------------
* User's application .c file: Example call to a memory copy routine
*-------------------------------------------------------------------*/
#include Flash2833x_API_Library.h
...
Example_MemCopy(&Flash28_API_LoadStart, &Flash28_API_LoadEnd, \
&Flash28_API_RunStart);
...
/*-------------------------------------------------------------------
* User's application .c file: Example memory copy routine
*-------------------------------------------------------------------*/
void Example_MemCopy(Uint16 *SourceAddr, Uint16* SourceEndAddr, Uint16*
DestAddr)
{
        while(SourceAddr < SourceEndAddr) { *DestAddr++ = *SourceAddr++;
        return;
}
```

7: Inicijalizacija Flash_CPUScaleFactor

Flash_CPUScaleFactor – globalna promenljiva definisana u API

```
Flash_CPUScaleFactor = SCALE_FACTOR;
```

8: Initializacija pointera na Callback Funkciju

```
/*---------------------------------------------------------------
* User's Application with a Callback Function
*---------------------------------------------------------------*/
#include Flash2833x_API_Library.h
...
void MyCallbackFunction(void); // My Callback function prototype
...
Flash_CallbackPtr = &MyCallbackFunction;
...
...
void myCallbackFunction(void)
{
// User's application code to execute during the callback function
// This must not execute from flash/OTP or read data from flash/OTP
}

/*---------------------------------------------------------------
* User's Application without a Callback Function
*---------------------------------------------------------------*/
#include Flash2833x_API_Library.h
#include <stdio.h> // NULL is defined here
...
Flash_CallbackPtr = NULL;
```

Funkcija brisanja - Erase

TMS320F28335:

```
extern Uint16 Flash28335_Erase(
        Uint16 SectorMask, // Sector mask
        FLASH_ST *FEraseStat // Pointer to the status structure
);
```

Parameter:	**Description:**
Uint16 SectorMask	0 1 2 3 4 5 6 7 14:8 A B C D E F G H ignored
FLASH_ST *FEraseStat	Pointer na flash status structuru, definisanu u

Flash2833x_API_Library.h:

```
typedef struct {
        Uint32 FirstFailAddr;
        Uint16 ExpectedData;
        Uint16 ActualData;
}FLASH_ST;
```

Za Erase, koristi se samo FirstFailAddr struktura

Funkcija brisanja - Erase

```
/*-------------------------------------------------------------------
* Flash2833x_API_Library.h
*-------------------------------------------------------------------*/
#define SECTORA (Uint16)0x0001
#define SECTORB (Uint16)0x0002
#define SECTORC (Uint16)0x0004
#define SECTORD (Uint16)0x0008
#define SECTORE (Uint16)0x0010
#define SECTORF (Uint16)0x0020
#define SECTORG (Uint16)0x0040
#define SECTORH (Uint16)0x0080
```

```
/*-------------------------------------------------------------
* User's Application: Erase Sector D then erase sector C and B
*-------------------------------------------------------------*/
#include "Flash2833x_API_Library.h"
...
Uint16 Status;
FLASH_ST EraseStatus;
...
Flash_CPUScaleFactor = SCALE_FACTOR;
Flash_CallbackPtr = NULL;
...
// Code to set PLLCR and wait for PLL lock
...
// Erase Sector D
// Following is defined in Flash2833x_API_Library.h
// #define SECTORD (Uint16)0x0008
// User's Code:
Status = Flash_Erase(SECTORD,&EraseStatus);
if(Status != STATUS_SUCCESS) Error(Status);
...
// Erase Sector C and Sector B
// Following is defined in Flash2833x_API_Library.h
// #define SECTORC (Uint16)0x0004
// #define SECTORB (Uint16)0x0002
// User's Code:
Status = Flash_Erase((SECTORC|SECTORB),&EraseStatus);
if(Status != STATUS_SUCCESS) Error(Status);
...
```

Funkcija programiranja

```
extern Uint16 Flash28335_Program(
        Uint16 *FlashAddr, // Pointer to the first flash/OTP loc
        Uint16 *BufAddr, // Pointer to the buffer
        Uint32 Length, // Number of 16-bit values to program
        FLASH_ST *FProgStatus // Pointer to the status structure
);
```

```
/*-------------------------------------------------------------
* Example: Program 0x400 values into the flash starting at 0x330000
*-------------------------------------------------------------*/
#include Flash2833x_API_Library.h
#define WORDS_IN_FLASH_BUFFER 0x400
...
volatile Uint16 Buffer[WORDS_IN_FLASH_BUFFER];
Uint16 *Flash_ptr; // Pointer to a location in flash
Uint32 Length; // Number of 16-bit values to be programmed
FLASH_ST ProgStatus; // Status structure
Uint16 Status; // Return status
...
Flash_CPUScaleFactor = SCALE_FACTOR;
Flash_CallbackPtr = NULL;
...
// Code to set PLLCR and wait for PLL lock
...
// Fill the buffer with some data to program into the flash
for(i=0; i<0x400; i++) Buffer[i] = 0x8000+i;
...
Flash_ptr = (Uint16 *)0x00330000;
Length = 0x400;
...
// Call the program API function
Status = Flash_Program(Flash_ptr,Buffer,Length,&ProgStatus);
if(Status != STATUS_SUCCESS) Error(Status);
```

Funkcija verifikacije

```
extern Uint16 Flash28335_Verify(
        Uint16 *FlashAddr, // Pointer to the first flash/OTP loc
        Uint16 *BufAddr, // Pointer to the buffer
        Uint32 Length, // Number of 16-bit values to verify
        FLASH_ST *FVerifyStat // Pointers to the status structure
        );
```

Funkcija verifikacije

```
/*-----------------------------------------------------------
* Example: Verify 0x400 values in the flash starting at 0x330000
*-----------------------------------------------------------*/
#include Flash2833x_API_Library.h
#define WORDS_IN_FLASH_BUFFER 0x400
...
volatile Uint16 Buffer[WORDS_IN_FLASH_BUFFER];
Uint16 *Flash_ptr; // Pointer to a location in flash
Uint32 Length; // Number of 16-bit values to be programmed
FLASH_ST VerifyStatus; // Status structure
Uint16 Status; // Return status
...
Flash_CPUScaleFactor = SCALE_FACTOR;
Flash_CallbackPtr = NULL;
...
// Code to set PLLCR and wait for PLL lock
...
// Fill the buffer with some data to verify against
for(i=0; i<0x400; i++) Buffer[i] = 0x8000+i;
...
Flash_ptr = (Uint16 *)0x00330000;
Length = 0x400;
...
// Call the verify API function
Status = Flash_Verify(Flash_ptr,Buffer,Length,&VerifyStatus);
if(Status != STATUS_SUCCESS) Error(Status);
```