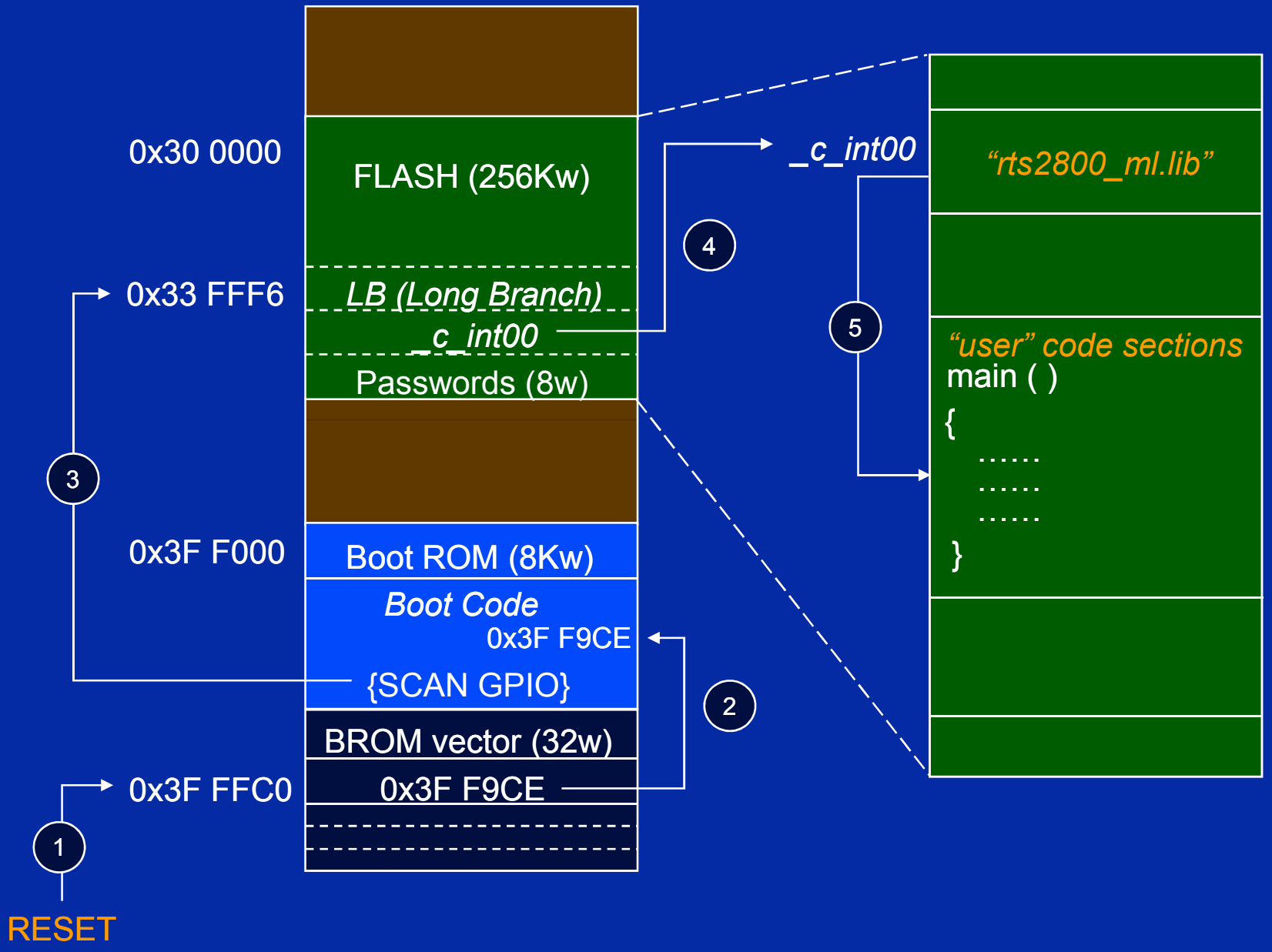
Modul 14: Programiranje “Flash” memorije

32-Bit-Digital Signal Controller TMS320F2833x

Texas Instruments Incorporated

“Startup” sekvenca iz “Flash” memorije



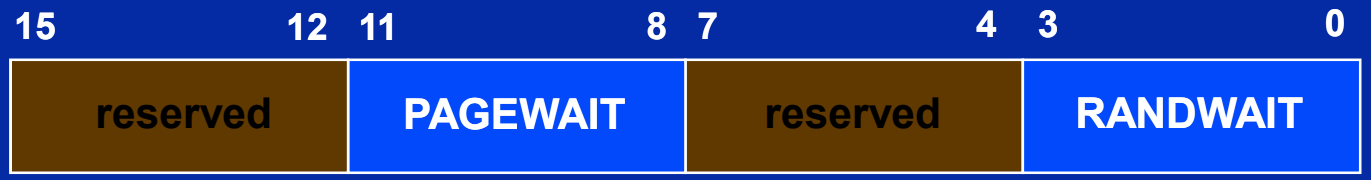
TMS320F28335 Flash memorijska mapa

Address Range	Data & Program Space
0x30 0000 – 0x30 7FFF	Sector H; 32K x 16
0x30 8000 – 0x30 FFFF	Sector G; 32K x 16
0x31 0000 – 0x31 7FFF	Sector F; 32K x 16
0x31 8000 – 0x31 FFFF	Sector E; 32K x 16
0x32 0000 – 0x32 7FFF	Sector D; 32K x 16
0x32 8000 – 0x32 FFFF	Sector C; 32K x 16
0x33 0000 – 0x33 7FFF	Sector B; 32K x 16
0x33 8000 – 0x33 FF7F	Sector A; (32K-128) x 16
0x33 FF80 – 0x33 FFF5	Program to 0x0000 when using Code Security Mode !
0x33 FFF6 – 0x33 FFF7	Flash Entry Point; 2 x 16
0x33 FFF8 – 0x33 FFFF	Security Password; 8 x 16

Basic Flash Operation

- Flash organizovan po stranicama veličine 128 adresa (reči)
- Wait stanja za sekvencijalne adrese unutar stranice, i proizvoljan pristup između stranica
- OTP ima samo proizvoljan pristup
- Neophodno specificirati broj SYSCLKOUT wait-stanja
  - *Reset definiše maksimalne vrednosti!*
- Konfiguracija Flash memorije ne sme se izvršavati iz Flash-a!

FBANKWAIT
@ 0x00 0A86



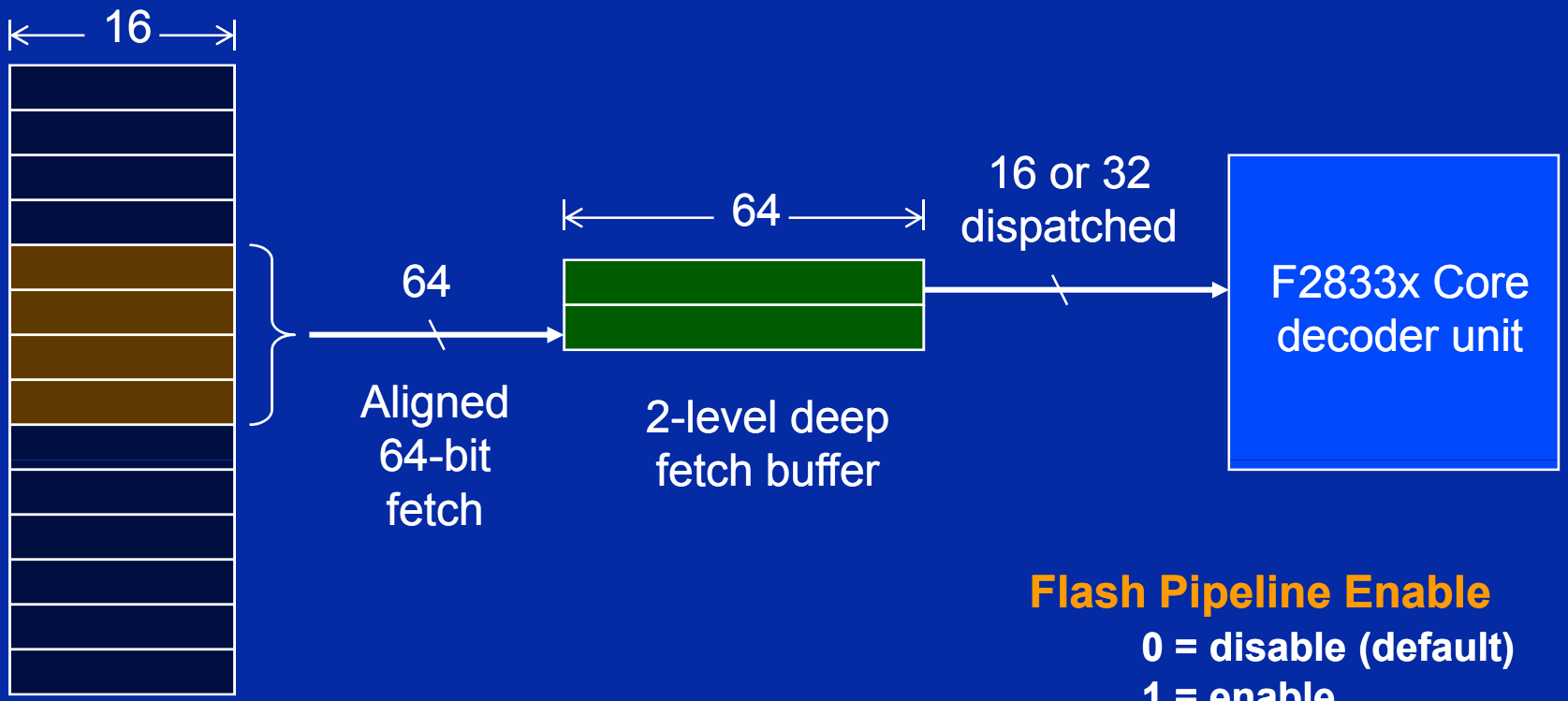
FOTPWAIT
@ 0x00 0A87

For 150 MHz, PAGEWAIT = 5, RANDWAIT = 5, OTPWAIT = 8

For 100 MHz, PAGEWAIT = 3, RANDWAIT = 3, OTPWAIT = 5

Ubrzavanje izvršavanja iz Flash-a:

Flash Pipelining (samo za učitavanje koda)



Flash Pipeline Enable

0 = disable (default)
1 = enable

FOPT @ 0x00 0A80

Flash Konfiguracioni registri

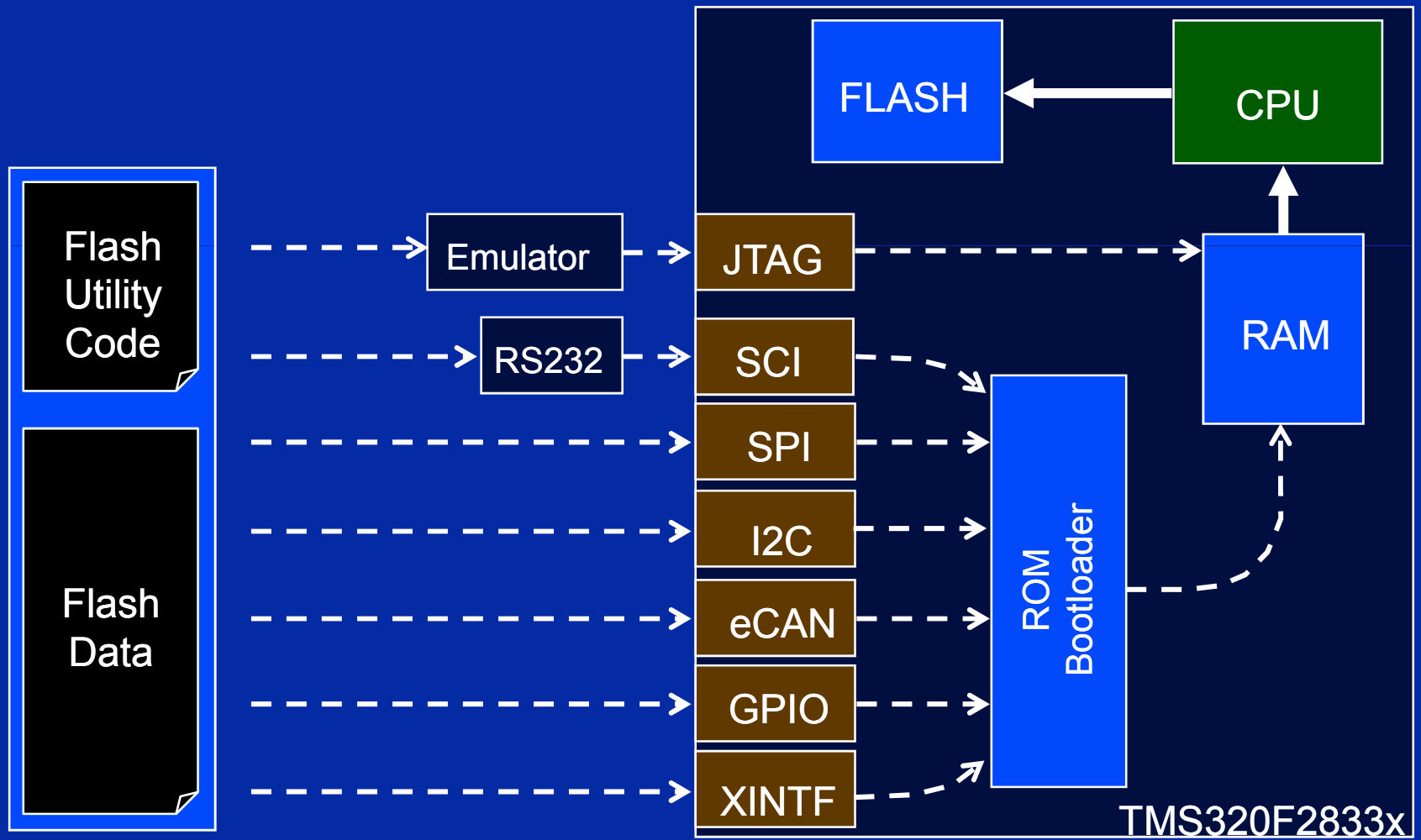
Address	Name	Description
0x00 0A80	FOPT	Flash option register
0x00 0A82	FPWR	Flash power modes registers
0x00 0A83	FSTATUS	Flash status register
0x00 0A84	FSTDBYWAIT	Flash sleep to standby wait register
0x00 0A85	FACTIVEWAIT	Flash standby to active wait register
0x00 0A86	FBANKWAIT	Flash read access wait state register
0x00 0A87	FOTPWAIT	OTP read access wait state register

- **FPWR:** Smanjenje potrošnje stavljanjem Flash/OTP u ‘Sleep’ ili ‘Standby’ režim; Flash automatski prelazi u aktivno stanje ako se pristupi Flash/OTP - u
- **FSTATUS:** Različiti statusni bitovi (PWR mode)
- **FSTDBYWAIT:** Specificira broj ciklusa čekanja u toku *wake-up* iz *sleep* stanja u *standby*
- **FACTIVEWAIT:** Specificira broj ciklusa čekanja u toku *wake-up* iz *standby* u aktivno stanje

Defoltne vrednosti najčešće zadovoljavaju sve potrebe

Osnove programiranja Flash-a

- Samoprogramiranje flash memorije
- CPU izvršava sekvencu programiranja iz RAM
- U RAM sem smeštaju "Flash utility code" i "Flash data"



Osnove programiranja Flash-a

- Sekvenca programiranja:

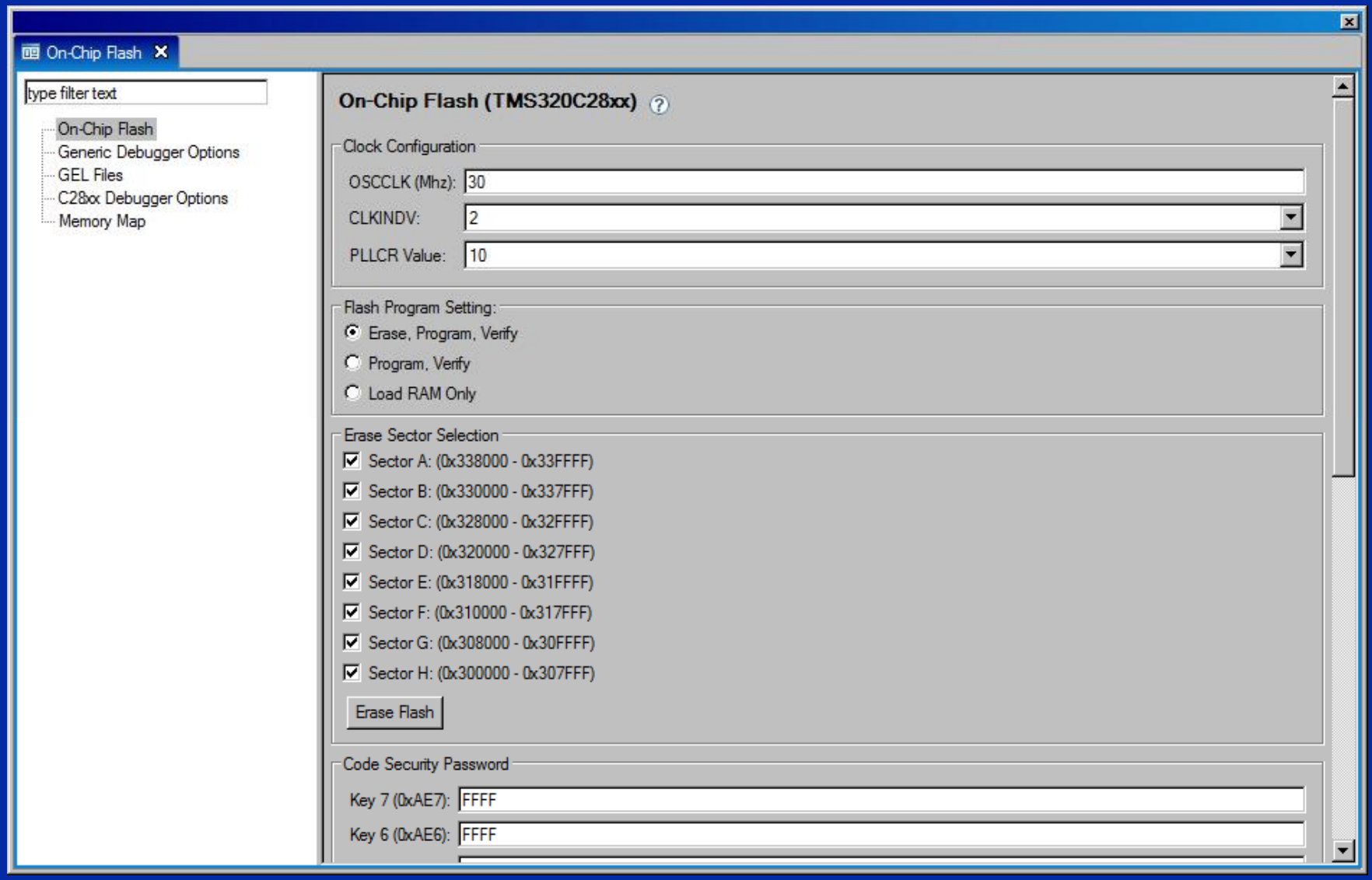
Algoritam	Funkcija
1. Erase	- Postavi sve bitove na nulu, zatim na jedinicu
2. Program	- Programiranje izabranih bitova na nulu
3. Verify	- Provera sadržaja

- Minimalna veličina za brisanje (Erase) je sector
- Minimalna veličina za programiranje je bit!
- Važno da ne dođe do nestanka napajanja u toku brisanja: Ako *CSM passwords* budu svi na nuli, CSM će biti zaključan za stalno!
- Šanse za ovo male! (korak Erase se izvršava sektor po sektor)

Uslužni programi za programiranje Flash-a

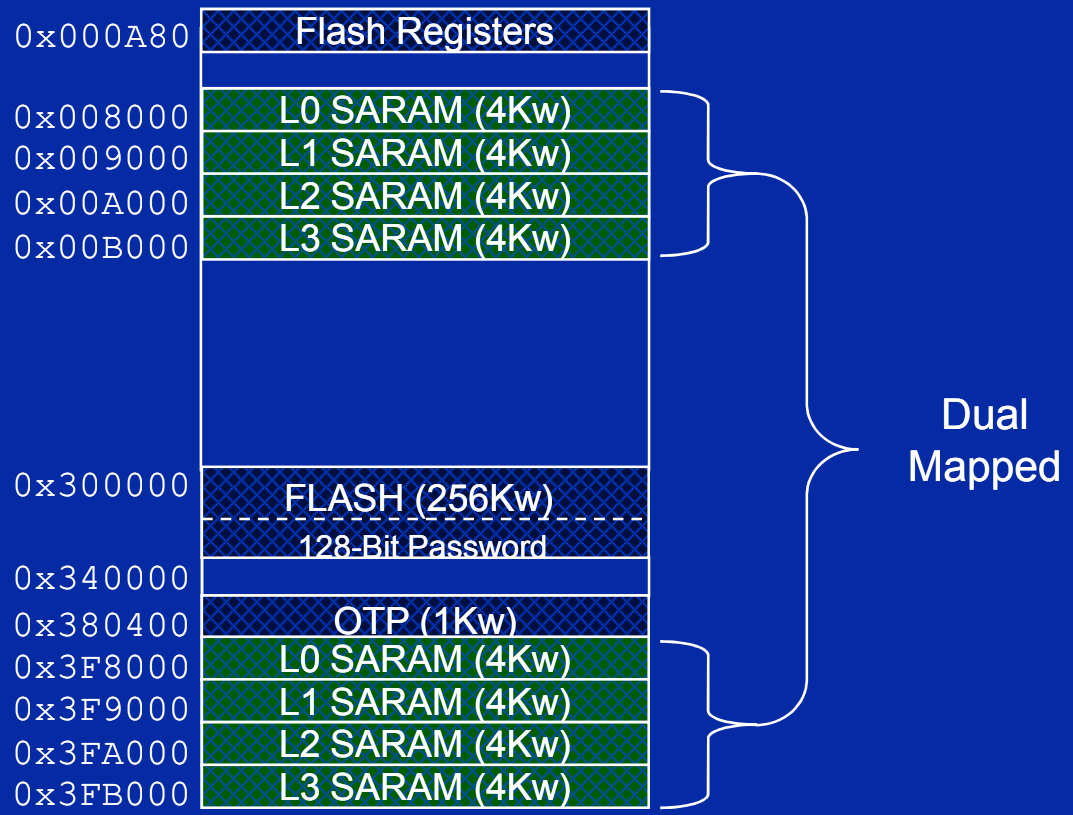
- Code Composer Studio Plug-in (koristi JTAG)
- Third-party JTAG utilities
  - SDFlash JTAG from Spectrum Digital (SD emulator)
  - Signum System Flash utilities (Signum emulator)
  - BlackHawk Flash utilities (Blackhawk emulator)
- SDFlash Serial utility (SCI boot)
- Gang Programmers (GPIO boot)
  - BP Micro programmer
  - Data I/O programmer
- Izrada sopstvenih uslužnih programa
  - Korišćenje drugačijih ROM bootloader-a
  - Programiranje fleša kroz aplikaciju
  - Flash API algorithmi koje je obezbedio TI

Code Composer Studio Flash Plug-In



Code Security Module (CSM)

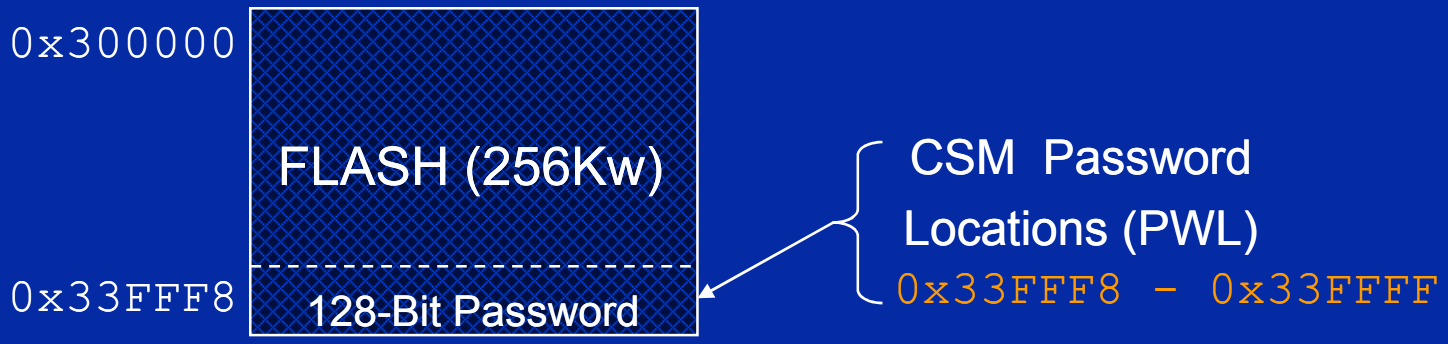
- Ograničen pristup sledićim oblastima:



- Čitanje i upis podataka iz zaštićenih oblasti moguće samo ako se izvršava iz zaštićenih oblasti
- Sva ostal čitanja i upisi su blokirani:

  JTAG emulator/debugger, ROM boot loader, izvršavanje programa iz spoljašnje memorije ili nezaštićenog prostora

CSM Password



- 128-bit korisnički definisani password smešten u Flash memoriji
- 128-bit KEY registri za *lock* i *unlock* kola
  - Mapirani u prostor 0x00 0AE0 – 0x00 0AE7
  - Registri “EALLOW” zaštićeni

CSM Registers

Key Registers – accessible by user; EALLOW protected

Address	Name	Reset Value	Description
0x00 0AE0	KEY0	0xFFFF	Low word of 128-bit Key register
0x00 0AE1	KEY1	0xFFFF	2nd word of 128-bit Key register
0x00 0AE2	KEY2	0xFFFF	3rd word of 128-bit Key register
0x00 0AE3	KEY3	0xFFFF	4th word of 128-bit Key register
0x00 0AE4	KEY4	0xFFFF	5th word of 128-bit Key register
0x00 0AE5	KEY5	0xFFFF	6th word of 128-bit Key register
0x00 0AE6	KEY6	0xFFFF	7th word of 128-bit Key register
0x00 0AE7	KEY7	0xFFFF	High word of 128-bit Key register
0x00 0AEF	CSMSCR	0xFFFF	CSM status and control register

PWL in memory – reserved for passwords only

Address	Name	Reset Value	Description
0x33 7FF8	PWL0	user defined	Low word of 128-bit password
0x33 7FF9	PWL1	user defined	2nd word of 128-bit password
0x33 7FFA	PWL2	user defined	3rd word of 128-bit password
0x33 7FFB	PWL3	user defined	4th word of 128-bit password
0x33 7FFC	PWL4	user defined	5th word of 128-bit password
0x33 7FFD	PWL5	user defined	6th word of 128-bit password
0x33 7FFE	PWL6	user defined	7th word of 128-bit password
0x33 7FFF	PWL7	user defined	High word of 128-bit password

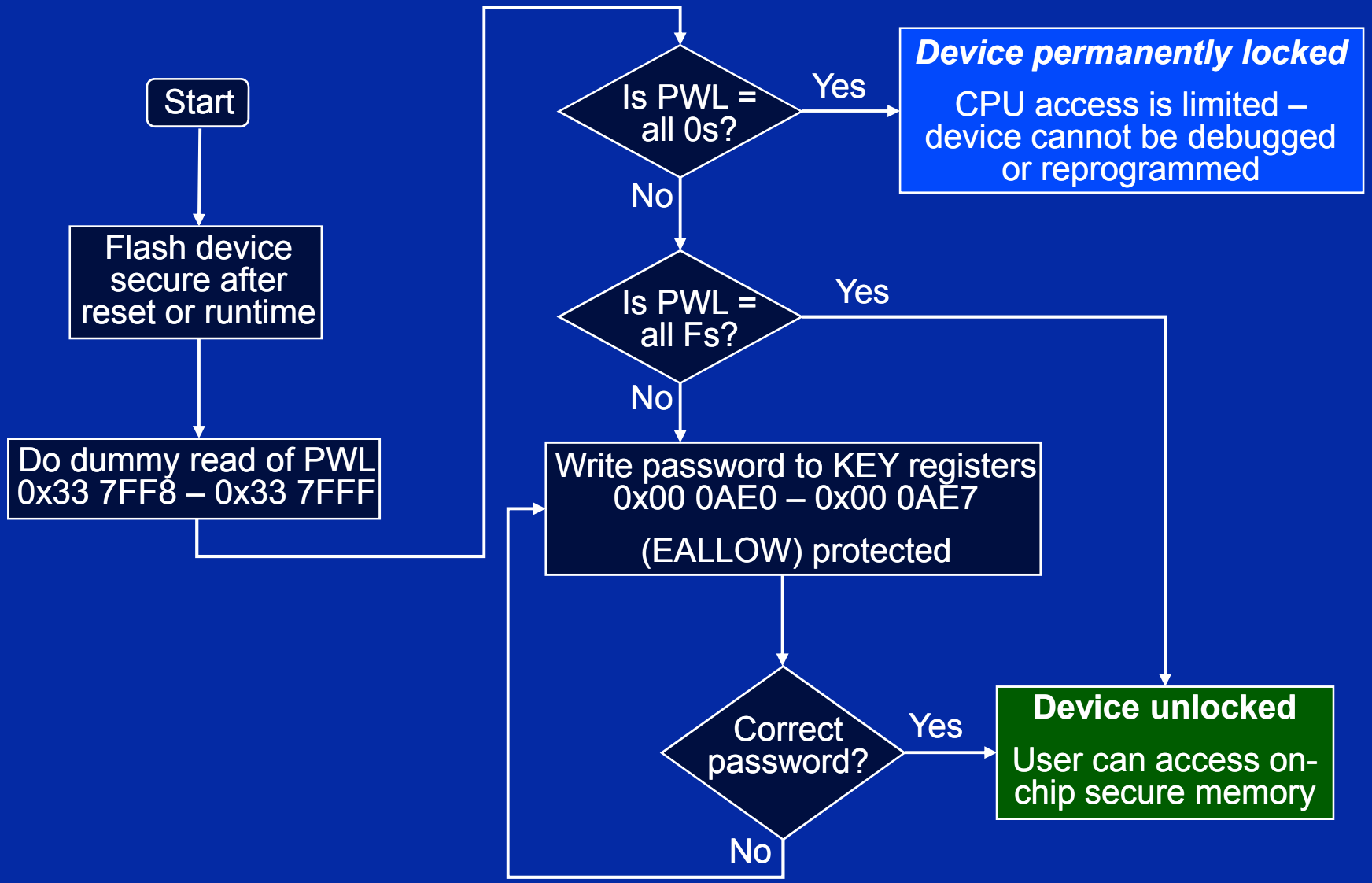
Zaključavanje i otključavanje CSM

- CSM je zaključan pri *power-up* i *reset*
- Otključavanje CSM:
  - Izvršiti *dummy read* svakog *password*-a u Flash memoriji
  - Izvršiti upis korektnog *password*-a u *key* registre
- Novi čip (PWL ima sve 0xFFFF):
  - Kada su svi *password*-i 0xFFFF – dovoljno je samo čitanje PWL da bi se čip otključao

Upozorenja za CSM

- Nikada ne programirati sve PWL sa 0x0000
  - *Permanentno zaključavanje CSM*
- Flash adrese 0x337F80 - 0x337FF5, moraju biti programirane na 0x0000 da bi se omogućilo zaključavanje CSM
- Treba upamtiti da program koji se izvršava iz nezaštićenog RAMa ne može pristupiti podacima iz zaštićene memorije
  - Ne linkovati stek u zaštićeni RAM ako postoji kod koji se izvršava u nezaštićenom RAMu
- Ne ugrađivati password u kod!
  - CSM se otključava samo u cilju debgiranja
  - Code Composer Studio može da otključa čip

CSM Password Match Flow



CSM C-Code Examples

Unlocking the CSM:

```
volatile int *PWL = &CsmPwl.PSWD0;    //Pointer to PWL register file
volatile int i, tmp;

for (i = 0; i<8; i++) tmp = *PWL++;   //Dummy reads of PWL locations

asm (" EALLOW");                      //KEY regs are EALLOW protected
CsmRegs.KEY0 = PASSWORD0;             //Write the passwords
CsmRegs.KEY1 = PASSWORD0;             //to the Key registers
CsmRegs.KEY2 = PASSWORD2;
CsmRegs.KEY3 = PASSWORD3;
CsmRegs.KEY4 = PASSWORD4;
CsmRegs.KEY5 = PASSWORD5;
CsmRegs.KEY6 = PASSWORD6;
CsmRegs.KEY7 = PASSWORD7;
asm (" EDIS");
```

Locking the CSM:

```
asm(" EALLOW");                       //CSMSCR reg is EALLOW protected
CsmRegs.CSMSCR.bit.FORCESEC = 1;      //Set FORCESEC bit
asm ("EDIS");
```