Customer training workshop

TRAVEO™ T2G Nonvolatile Memory Programming

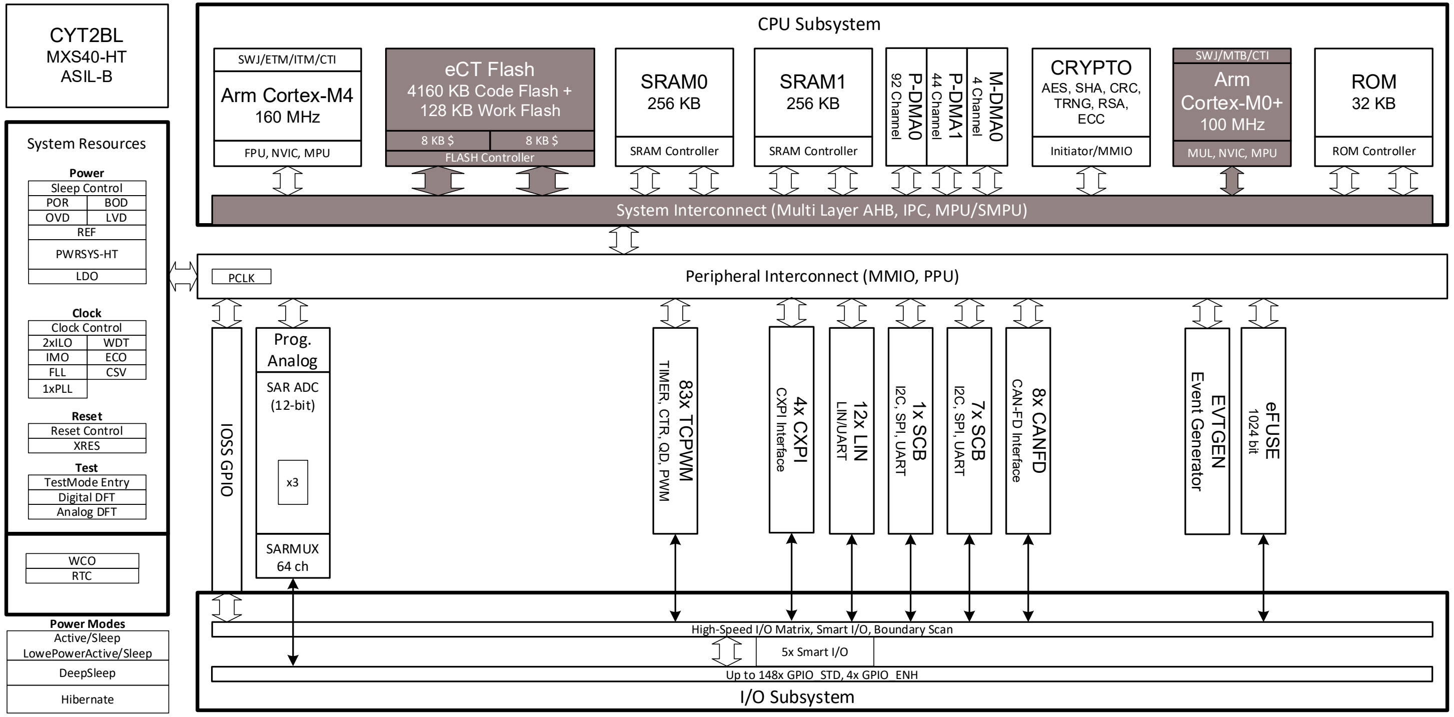
Q1 2024

Target products

› Target product list for this training material

Family Category	Series	Code Flash Memory Size
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B6	Up to 576 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B7	Up to 1088 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2B9	Up to 2112 KB
TRAVEO™ T2G Automotive Body Controller Entry	CYT2BL	Up to 4160 KB
TRAVEO™ T2G Automotive Body Controller High	CYT3BB/4BB	Up to 4160 KB
TRAVEO™ T2G Automotive Body Controller High	CYT4BF	Up to 8384 KB
TRAVEO™ T2G Automotive Body Controller High	CYT6BJ	Up to 16768 KB
TRAVEO™ T2G Automotive Cluster Entry	CYT2CL	Up to 4160 KB
TRAVEO™ T2G Automotive Cluster 2D	CYT3DL	Up to 4160 KB
TRAVEO™ T2G Automotive Cluster 2D	CYT4DN	Up to 6336 KB

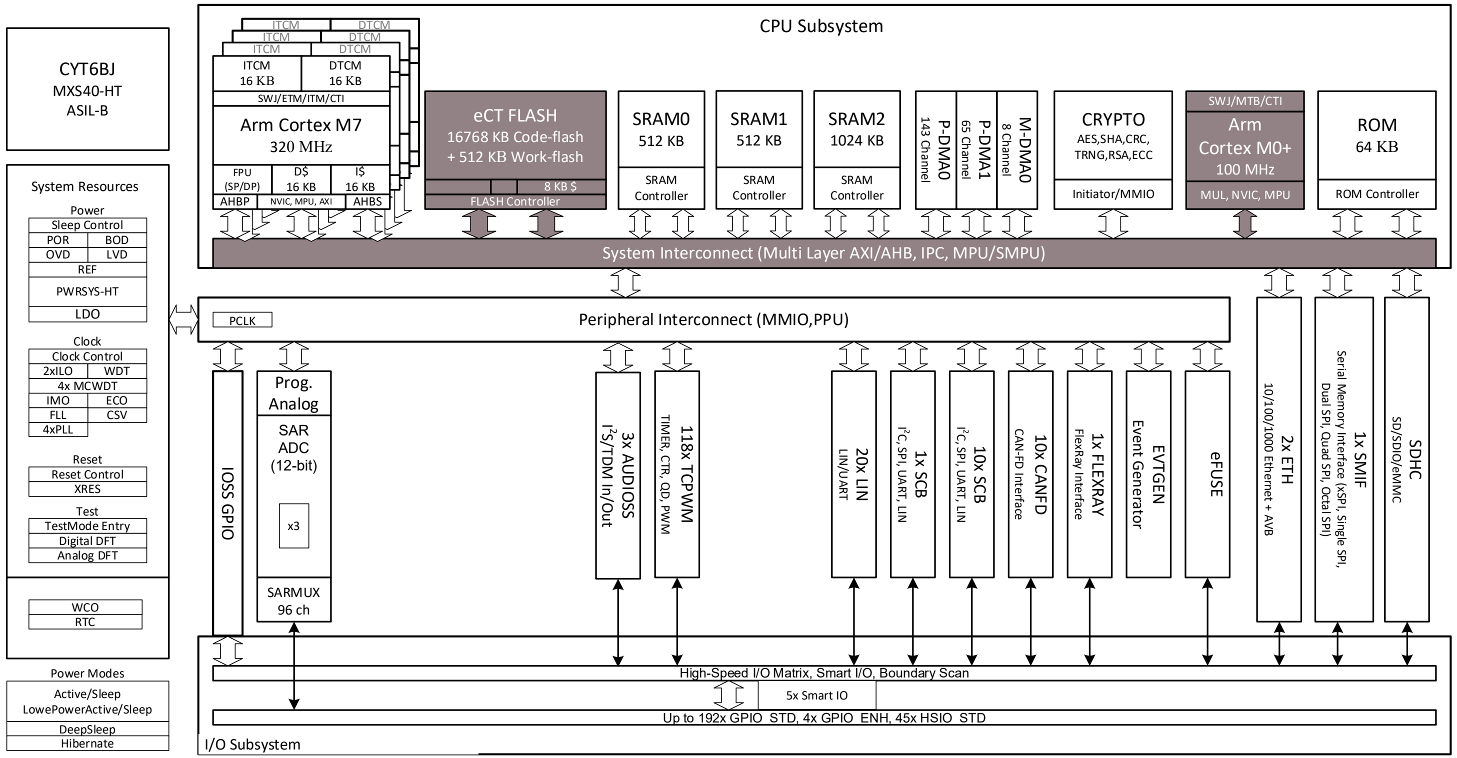
Introduction to TRAVEO™ T2G Body Controller Entry



Hint Bar

Review TRM chapter 33 for additional details

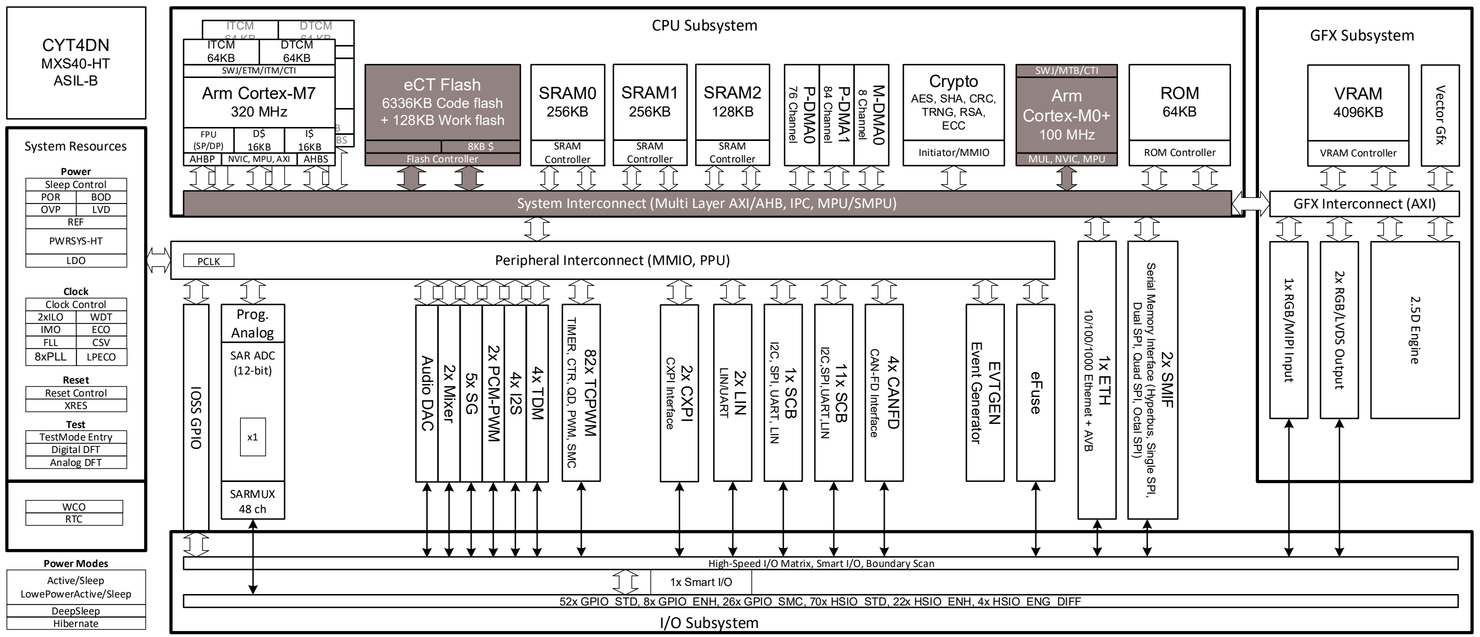
Introduction to TRAVEO™ T2G Body Controller High



Hint Bar

Review TRM chapter 41 for additional details

Introduction to TRAVEO™ T2G Cluster



Hint Bar

Review TRM chapter 39 for additional details

Nonvolatile memory (NVM) programming overview

- NVM programming supports flash-specific operations including:
  - Erase, Program, NORMAL access restrictions in SFlash[1], and storing public key
- CYT2B6/B7/B9/BL supports programming through the debug access port (DAP), Cortex®-M4, and Cortex®-M0+
- CYT3BB/4BB/4BF/6BJ, CYT3DL/4DN supports programming through DAP, Cortex®-M7, and Cortex®-M0+
- eFuse memory
  - eFuse memory consists of a set of eFuse bits
  - Some eFuse bits store fixed device parameters, including factory trim settings, life-cycle stages, DAP security settings, and encryption keys
  - Limited set of eFuse bits are available for customer use

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B and chapter 39 for CYT3D/4D for additional details

[1] Supervisory flash

eFuse memory overview

- eFuse bits can be programmed (or “blown”) in a manufacturing environment
  - eFuse bits cannot be programmed on the field
- Multiple eFuses can be read at the bit- or byte-level through an SROM call
  - An unblown eFuse reads as logic 0 and a blown eFuse reads as logic 1
  - There are no hardware connections from eFuse bits to elsewhere in the device
- There are 1024 eFuse bits, of which 192 are available for custom purposes

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B, chapter 39 for CYT3D/4D and eFuse Memory for additional details

Flash programming operations for CYT2

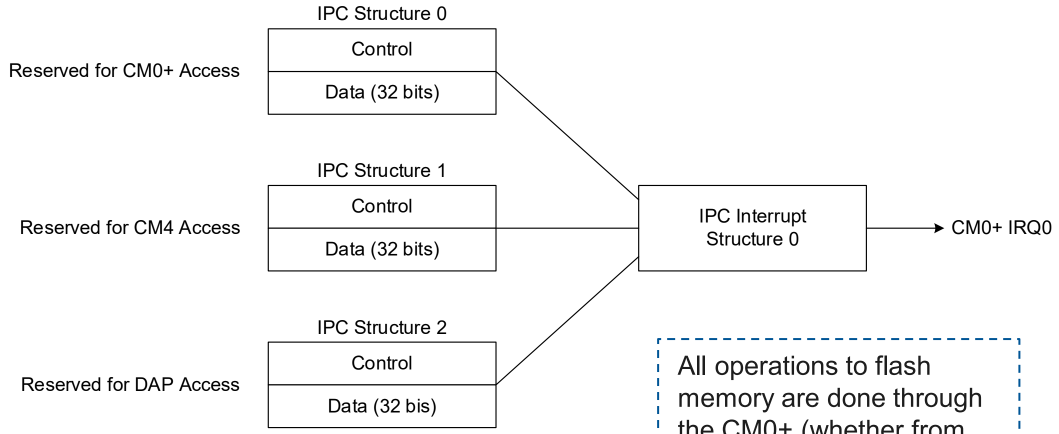
- Flash programming operations are implemented as system calls
- System calls are executed inside Cortex®-M0+ (CM0+) IRQ0
- System calls can be performed by CM0+, Cortex®-M4, or DAP
  - Each has a reserved IPC structure through which it can request CM0+ to perform a system call

For user programming

System calls can be made from the CM0+ or CM4 at any point during code execution

For debugger and programmer

When the debug interface is acquired, the boot ROM enters the busy-wait loop and waits for commands issued by the DAP



All operations to flash memory are done through the CM0+ (whether from the DAP or from the CPU)

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B and chapter 39 for CYT3D/4D for additional details

Flash programming operations for CYT3/CYT4

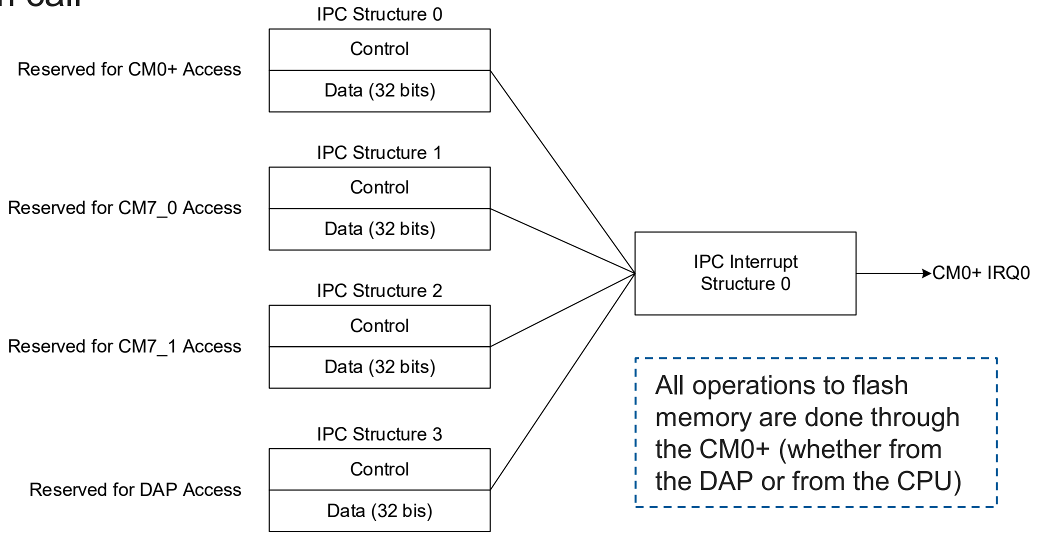
- Flash programming operations are implemented as system calls
- System calls are executed inside CM0+ IRQ0
- System calls can be performed by CM0+, CM7_0, CM7_1, or DAP
  - Each has a reserved IPC structure through which it can request CM0+ to perform a system call

For user programming

System calls can be made from the CM0+ or CM7_0 or CM7_1 at any point during code execution

For debugger and programmer

When the debug interface is acquired, the boot ROM enters the busy-wait loop and waits for commands issued by the DAP



All operations to flash memory are done through the CM0+ (whether from the DAP or from the CPU)

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B and chapter 39 for CYT3D/4D for additional details

Flash programming operations for CYT6

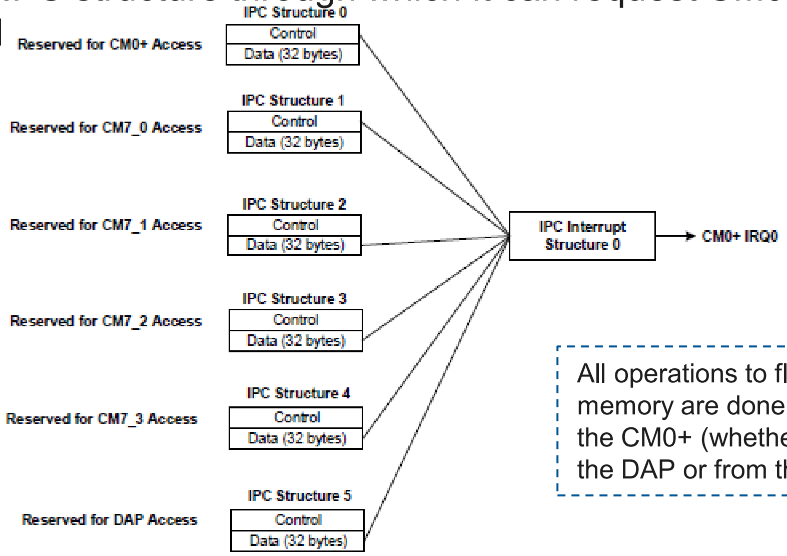
- Flash programming operations are implemented as system calls
- System calls are executed inside CM0+ IRQ0
- System calls can be performed by CM0+, CM7_0, CM7_1, CM7_2, CM7_3, or DAP
  - Each has a reserved IPC structure through which it can request CM0+ to perform a system call

For user programming

System calls can be made from the CM0+ or CM7_0 or CM7_1 or CM7_2 or CM7_3 at any point during code execution

For debugger and programmer

When the debug interface is acquired, the boot ROM enters the busy-wait loop and waits for commands issued by the DAP



All operations to flash memory are done through the CM0+ (whether from the DAP or from the CPU)

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B and chapter 39 for CYT3D/4D for additional details

SROM API Library (1/4)

No.	System Call	Opcode	Description	Access Allowed		
				Normal[1]	Secure	Dead
1	BlankCheck	0x2A	Performs blank check on the addressed Work Flash	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
2	BlowFuseBit	0x01	Blows an eFuse bit	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2]	
3	CheckFactoryHash	0x27	Generates the FACTORY_HASH as per TOC1 and compares with the FACTORY1_HASH fuses	CM0+, Main CPU[2], DAP		
4	CheckFMStatus	0x07	Returns the status of the flash operation	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
5	Checksum	0x0B	Calculates the checksum of a flash region	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
6	ComputeBasicHash	0x0D	Computes the hash value of a flash region	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
7	ConfigureFMInterrupt	0x08	Configures the flash macro interrupt	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
8	DirectExecute	0x0F	Directly executes code located at a configurable address	DAP		
9	EraseAll	0x0A	Erases all flash	CM0+, Main CPU[2], DAP		DAP
10	EraseResume	0x23	Resumes a suspended erase operation	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B and chapter 39 for CYT3D/4D for additional details

Refer to the Device Security training section for additional details about Normal/Secure/Dead for Access Allowed

[1] Refer to TRM chapter 14 (Chip Operational Modes)
[2] The main CPU refers to CM4 or CM7 CPU in the MCU.

SROM API Library (2/4)

No.	System Call	Opcode	Description	Access Allowed		
				Normal[1]	Secure	Dead
11	EraseSector	0x14	Erases a flash sector	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
12	EraseSuspend	0x22	Suspends an ongoing erase operation	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
13	GenerateHash	0x1E	Returns the truncated SHA-256 of the Flash boot programmed in SFlash	CM0+, Main CPU[2], DAP		
14	SwitchOverRegulators[3]	0x11	Switches between REGHC and linear regulators	CM0+	CM0+	
15	ConfigureRegulator[4]	0x15	Configures high-current regulator (REGHC) for devices that include REGHC, or PMIC for devices that use PMIC control without REGHC	CM0+	CM0+	
16	ProgramRow	0x06	Programs the addressed flash page	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
17	ProgramWorkFlash	0x30	Programs the addressed work flash page	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
18	ReadFuseByte	0x03	Reads addressed eFuse byte	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	
19	ReadFuseByteMargin	0x2B	Reads addressed eFuse byte marginally	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B and chapter 39 for CYT3D/4D for additional details

Refer to the Device Security training section for additional details about Normal/Secure/Dead for Access Allowed

[1] Refer to TRM chapter 14 (Chip Operational Modes)
[2] The main CPU refers to CM4 or CM7 CPU in the MCU.

SROM API Library (3/4)

No.	System Call	Opcode	Description	Access Allowed		
				Normal[1]	Secure	Dead
20	ReadSWPU	0x2C	Reads the identified SWPU from SRAM	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	
21	ReadUniqueID	0x1F	Reads the unique ID of the die from flash	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
22	SetEnforcedApproval	0x2E	Sets the EnforcedApproval bit in SRAM	CM0+	CM0+	CM0+
23	SiliconID	0x00	Returns Family ID, Revision ID, Silicon ID, and protection state	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
24	SoftReset	0x1B	Provides system reset or Main CPU[2] only reset	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
25	TransitiontoRMA	0x28	Converts parts from SECURE to RMA life-cycle stage		CM0+, Main CPU[2], DAP	
26	TransitiontoSecure	0x2F	Converts parts to Secure life-cycle stage	CM0+, Main CPU[2], DAP		
27	WriteRow	0x05	Programs SFlash	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
28	WriteSWPU	0x2D	Updates the identified SWPU in SRAM	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	
29	DebugPowerUpDown[3]	0x12	Enables/disables CM7 debugging	CM0+, DAP	CM0+, DAP	CM0+, DAP
30	LoadRegulatorTrims[4]	0x16	Sets proper trims to PWR_TRIM_HT_PWRSYS_CTL	CM0+	CM0+	

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B and chapter 39 for CYT3D/4D for additional details

Refer to the Device Security training section for additional details about Normal/Secure/Dead for Access Allowed

[1] Refer to TRM chapter 14 (Chip Operational Modes)
[2] The main CPU refers to CM4 or CM7 CPU in the MCU.
[3] DebugPowerUpDow is only for CYT3/4/6
[4] LoadRegulatorTrims is only for CYT3/4/6

SROM API Library (4/4)

No.	System Call	Opcode	Description	Access Allowed		
				Normal[1]	Secure	Dead
31	CheckFmStatus2	0x0C	Returns the status of the flash operation on the 2nd Flash Controller	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
32	Checksum2	0x19	Calculates the checksum of a flash region on the 2nd Flash Controller	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
33	ComputeBasicHash2	0x04	Sets the EnforcedApproval bit in SRAM	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
34	ConfigureFmInterrupt2	0x17	Computes the hash value of a flash region on the 2nd Flash Controller	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
35	EraseAll2	0x18	Erases all flash on the 2nd Flash Controller	CM0+, Main CPU[2], DAP		DAP
36	EraseResume2	0x26	Resumes a suspended erase operation on the 2nd Flash Controller	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
37	EraseSector2	0x1C	Erases a flash sector on the 2nd Flash Controller	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU2, DAP
38	EraseSuspend2	0x25	Suspends and ongoing erase operation on the 2nd Flash Controller	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
39	ProgramRow2	0x09	Programs the addressed flash page on the 2nd Flash Controller	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP
40	ProgramWorkFlash2	0x31	Programs the addressed work flash page on the 2nd Flash Controller	CM0+, DAP	CM0+, Main CPU[2], DAP	CM0+, Main CPU[2], DAP

Hint Bar

These APIs are only available for CYT6BJ. Review TRM chapter 41 for CYT6B for additional details

Refer to the Device Security training section for additional details about Normal/Secure/Dead for Access Allowed

[1] Refer to TRM chapter 14 (Chip Operational Modes)
[2] The main CPU refers to CM7 CPU in the MCU.

API summary (1/5)

No.	System Call	Summary
1	BlankCheck	Performs blank check on the addressed Work Flash.
2	BlowFuseBit	Blows the addressed eFuse bit. The read value of a blown eFuse bit is ‘1’.
3	CheckFactoryHash	Generates FACTORY_HASH as per TOC1 and compares with the FACTORY1_HASH fuses.
4	CheckFMStatus	Returns the status of the flash operation.
5	CheckFMStatus2	CheckFmStatus2 is a copy of the CheckFmStatus system call. User shall call this function if he accesses the upper 8 MB of flash for the CYT6BJ.
6	Checksum	Reads either the whole flash or a row of flash and returns the sum of each byte read.
7	Checksum2	Checksum2 is a copy of the Checksum system call. User shall call this function if he accesses the upper 8 MB of flash for the CYT6BJ.
8	ComputeBasicHash	Generates the hash of the flash region provided using the formula: H(n+1) = {H(n)*2+Byte}% 127; where H(0) = 0 This function returns an invalid address status if called on an out-of-bound flash region.
9	ComputeBasicHash2	ComputeBasicHash2 is a copy of the ComputeBasicHash system call. User shall call this function if he accesses the upper 8 MB of flash for the CYT6BJ.
10	ConfigureFMInterrupt	Configures the flash macro interrupt. The functionalities provided are: - Set interrupt mask - Clear interrupt mask - Clear interrupt

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B, chapter 39 for CYT3D/4D, and System Calls for additional details

API summary (2/5)

No.	System Call	Summary
11	ConfigureFMInterrupt2	ConfigureFmInterrupt2 is a copy of the ConfigureFmInterrupt. Call this function if you need to access the upper 8 MB of flash for the CYT6BJ.
12	EraseAll	Erases the entire flash macro that is specified. This API will erase only the Code Flash. The API returns a fail status if the user does not have write access to flash based on the SMPU settings.
13	EraseAll2	EraseAll2 is a copy of the EraseAll. Call this function if you need to access the upper 8 MB of flash for the CYT6BJ.
14	EraseResume	Resumes a suspended erase operation.
15	EraseResume2	EraseResume2 is a copy of the EraseResume. Call this function if you need to access the upper 8 MB of flash for the CYT6BJ.
16	EraseSector	Starts an erase operation on a specified sector and cannot be called on SFlash[1].
17	EraseSector2	EraseSector2 is a copy of the EraseSector. Call this function if you need to access the upper 8 MB of flash for the CYT6BJ.
18	EraseSuspend	Suspends an ongoing erase operation. Do not read from a suspended sector. The Program Row API function returns an error if invoked on the suspended sector.
19	EraseSuspend2	EraseSuspend2 is a copy of the EraseSuspend. Call this function if you need to access the upper 8 MB of flash for the CYT6BJ.
20	GenerateHash	Returns the truncated SHA-256 of the flash boot programmed in SFlash and optionally includes the public key and other objects as indicated in the Table of Contents (TOC). Gets the flash boot size from TOC. Typically, this function is called to check if the HASH blown into eFuse matches with what the ROM boot expects it to be.

[1] Supervisory flash

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B, chapter 39 for CYT3D/4D, and System Calls for additional details

API summary (3/5)

No.	System Call	Summary
21	SwitchOverRegulators	Switch between the high-current regulator (REGHC or PMIC without REGHC) required to run CM7 and the linear regulator (LDO). It should be called to switch from LDO to REGHC before enabling CM7. Call the Configure Regulator system before using this function.
22	ConfigureRegulator	Configure the high-current regulator (REGHC) for devices that include REGHC, or PMIC for devices that use PMIC control without REGHC. It should be called to configure the desired regulator only once before switching to the regulator using the Switch Over Regulators system call.
23	ProgramRow	Programs the addressed flash page (Code Flash or Work Flash). The user must provide the data to be loaded and the flash address to be programmed. The flash page should be in the erased state. Any system call using Flash Programming cannot be aborted or cancelled.
24	ProgramRow2	ProgramRow2 is a copy of the ProgramRow. Call this function if you need to access the upper 8 MB of flash for the CYT6BJ.
25	ProgramWorkFlash	Programs the addressed work flash. The flash page should be in the erased state.
26	ProgramWorkFlash2	ProgramWorkFlash2 is a copy of the ProgramWorkFlash. Call this function if you need to access the upper 8 MB of flash for the CYT6BJ.
27	ReadFuseByte	Returns the value of an eFuse. The read value of a blown eFuse bit is ‘1’ and that of an unblown eFuse bit is ‘0’. This API inherits the client protection context.
28	ReadFuseByteMargin	Returns the eFuse contents of the addressed byte read marginally. The read value of a blown eFuse bit is ‘1’ and that of an unblown eFuse bit is ‘0’. This API inherits the client’s protection context.
29	ReadSWPU	Reads the identified SWPU from SRAM. The PU ID is based on the storage of SWPU in SFlash. Only one contiguous SWPU will index in SFlash[1] even though there are two physically separate storage areas.

[1] Supervisory flash

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B, chapter 39 for CYT3D/4D, and System Calls for additional details

API summary (4/5)

No.	System Call	Summary
30	ReadUniqueID	Returns the unique ID of the die from SFlash[1].
31	SetEnforcedApproval	Sets the EnforcedApproval bit in SRAM. EnforcedApproval bit is stored in PC1 private SRAM. If this bit is set, the API checks for a supervised marker.
32	SiliconID	Returns a 12-bit family ID, 16-bit silicon ID, 8-bit revision ID, and the current protection state.
33	SoftReset	Resets the system by setting the CM0+ AIRCR system reset bit. This results in a system-wide reset except for debug logic. This API can also be used to selectively reset just the CM4/CM7_0/CM7_1 cores based on 'type' parameter. CM4/CM7_0/CM7_1 should be in DeepSleep mode when it resets selectively.
34	TransitiontoRMA	Converts parts from SECURE to RMA life-cycle stage.
35	TransitiontoSecure	Validates the FACTORY_HASH and programs SECURE_HASH, secure access restrictions, and dead access restrictions into eFuse. Programs secure or secure with debug fuse to transition to SECURE or SECURE with DEBUG life-cycle stage. Only allowed in NORMAL_PROVISIONED stage.
36	DirectExecute	Directly executes code located at a configurable address. The API is allowed in VIRGIN state. In NORMAL, SECURE, and DEAD states, the API is allowed only if the corresponding DIRECT_EXECUTE_DISABLE bit (in SFlash[1]/eFuse) is 0[2].
37	WriteRow	Programs flash. User must provide data to be loaded and flash address to be programmed. This API can be called only on SFlash[1]. Performs pre-program, erase, and then programs the flash page with contents provided in SRAM.

[1] Supervisory flash

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B, chapter 39 for CYT3D/4D, and System Calls for additional details

API summary (5/5)

No.	System Call	Summary
38	WriteSWPU	Updates the identified SWPU in SRAM if the client has appropriate access. The PU ID is based on the storage of SWPU in SFlash. Only one contiguous SWPU indexes in SFlash even though there are two physically separate storages.
39	DebugPowerUpDown	Used for handling the power transitions of CM7_0/1 power domains to properly connect/disconnect debug probe to/from the device.
40	LoadRegulatorTrims	Used to adapt the output voltage for internal regulators during handover.
41	OpenRMA	Enables full access to the device in the RMA life-cycle stage upon successful execution

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B, chapter 39 for CYT3D/4D, and System Calls for additional details

Example of Flash memory operation with API (1/2)

- Use case
  - Flash erase by CM4/CM7 master using the Erase All API
    - CM4/CM7 requests a system call to CM0+
    - API parameters are passed using IPC
    - Erase All API parameters are as follows

Master (CM4/7) setting parameters of Erase All API

Address	Value to be Written	Description
IPC_DATA0 Register		
Bits [31:0]	SRAM_SCRATCH_ADDR	SRAM address where the API parameters are stored. This must be a 32-bit aligned address
SRAM_SCRATCH_ADDR		
Bits [31:24]	0x0A	Erase All opcode
Bits [23:0]	0xXXXXXX	Not used

Return of API Execution Result from CM0+

Address	Value to be Written	Description
SRAM_SCRATCH_ADDR		
Bits [31:28]	0xA = SUCCESS 0xF = ERROR	
Bits [27:0]	Error Code	A failure status is indicated by 0xF00000XX

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B, chapter 39 for CYT3D/4D, System Calls, and System Call Status for additional details

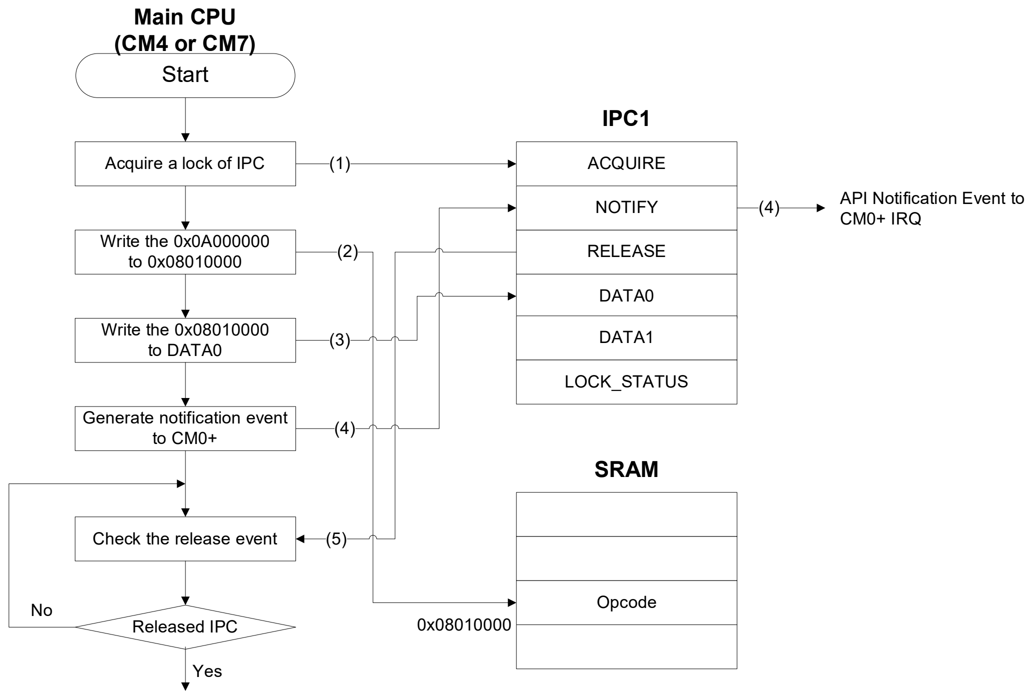
Refer to the Inter-processor Communication training section for additional IPC details

Use case:

API operation (1/3)

› Steps to activate Erase All API flow
  – SRAM_SCRATCH_ADDR = 0x08010000[1]
  – Erase All opcode = 0x0A000000

1. Acquire a lock
2. Write the opcode of Erase All to SRAM_SCRATCH_ADDR
3. Write SRAM_SCRATCH_ADDR to DATA0
4. Generate notification event by writing to the IPC_NOTIFY register
5. Wait to be released by the IPC by polling the IPC_RELEASE register or RELEASE event



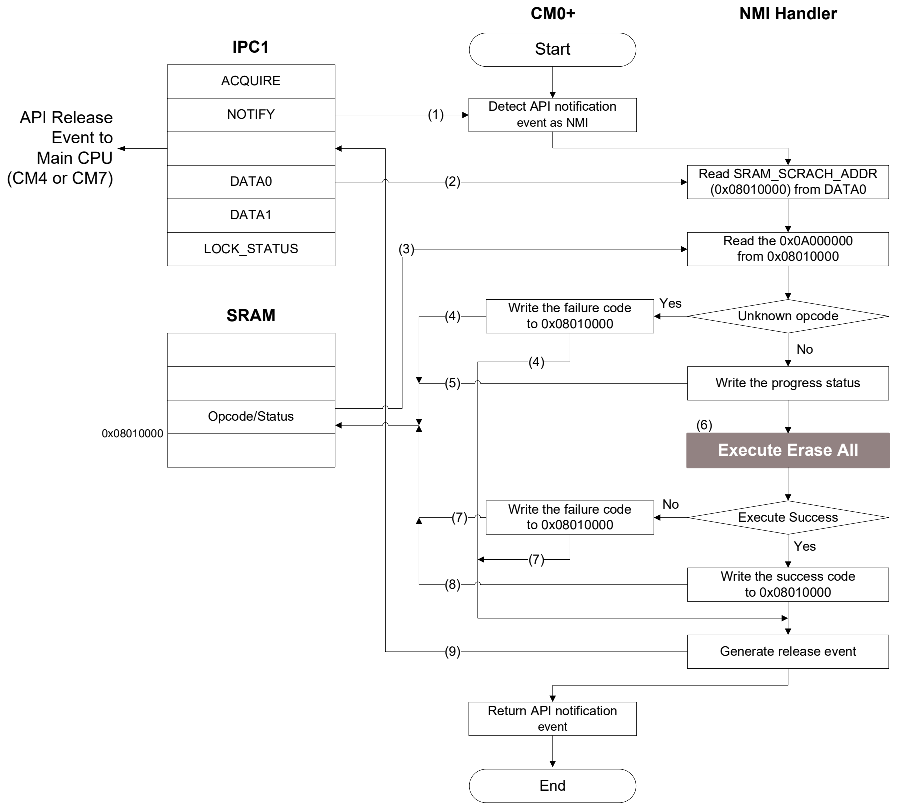
[1] SRAM addresses for the CY2B7/B9 series.

Use case:

API operation (2/3)

› Steps to execute API flow of Erase All API
  – Execution of API is performed by CM0+
  – SRAM_SCRATCH_ADDR area is used as Status code after reading opcode

1. Detect API notification event
2. Read SRAM_SCRACH_ADDR (0x080100001) from DATA0
3. Read the opcode (0x0A000000) from SRAM_SCRATCH_ADDR (0x08010000)
4. If opcode is unknown, write the failure code to SRAM_SCRATCH_ADDR (0x08010000)
5. Write the progress code to SRAM_SCRATCH_ADDR (0x08010000)
6. Execute Erase All
7. If the result is fail, write the failure code to SRAM_SCRATCH_ADDR (0x08010000)
8. Write the success code to SRAM_SCRATCH_ADDR (0x08010000)
9. Generate a release event by writing to the IPC_RELEASE register



CM0+
NMI Handler
IPC1
Start
ACQUIRE
NOTIFY
Detect API notification event as NMI
API Release Event to Main CPU (CM4 or CM7)
DATA0
Read SRAM_SCRACH_ADDR (0x08010000) from DATA0
DATA1
LOCK_STATUS
Read the 0x0A000000 from 0x08010000
SRAM
Write the failure code to 0x08010000
Yes
Unknown opcode
No
Write the progress status
0x08010000
Opcode/Status
Execute Erase All
Write the failure code to 0x08010000
No
Execute Success
Yes
Write the success code to 0x08010000
Generate release event
Return API notification event
End
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)

[1] SRAM addresses for the CY2B7/B9 series.

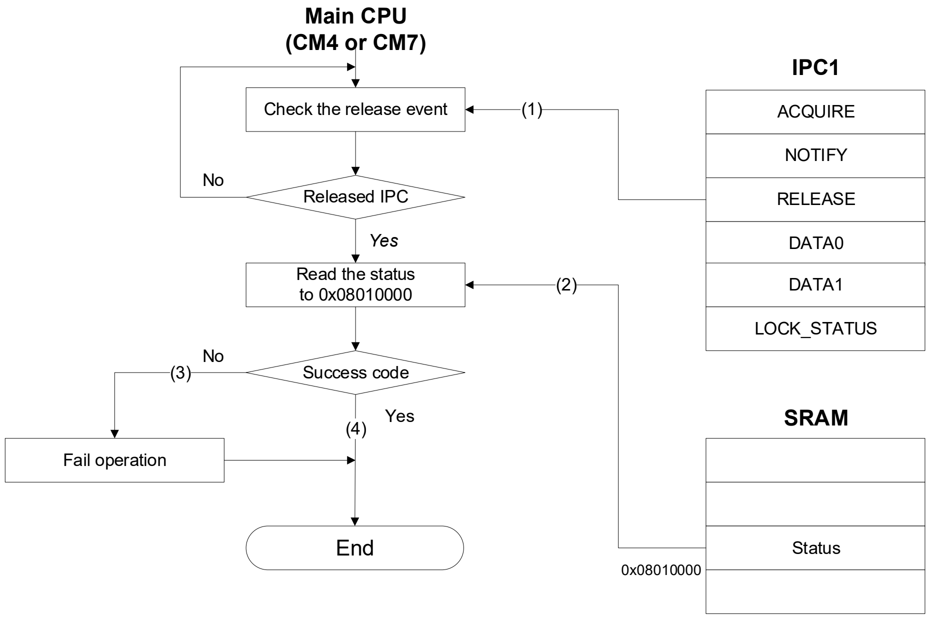
Use case:

API operation (3/3)

- Closed API flow of Erase All-API
- SRAM_SCRATCH_ADDR = 0x08010000[1]
  - Erase All opcode = 0x0A000000
  - SRAM_SCRATCH_ADDR area is stored in Status code

1. Detect API release event
2. Read the Status from DATA0
3. If the status code is failure, transfer to Fail operation and end API
4. If the status code is success, the API ends normally

- Release event can also be reported as an interrupt



[1] SRAM addresses for the CY2B7/B9 series.

Example of flash memory operation with API (2/2)

- Use case
  - Program operation by CM4/CM7 using the Program Row API
    - CM4/CM7 requests a system call to CM0+
    - API parameters are passed using IPC
    - Program Row API parameters are as follows
  - The example shows write operation with 64-bit test data (0x55AA55AA x 2) into code flash

Master (CM4/CM7) Setting Parameters of Program Row API

Address	Value to be Written	Description
IPC_DATA0 Register		
Bits [31:0]	SRAM_SCRATCH_ADDR	SRAM address where the API parameters are stored. This must be a 32-bit aligned address
SRAM_SCRATCH_ADDR		
Bits [31:24]	0x06	Program Row opcode
Bits [7:0]	0xXXXXXX	Not used

Return of API Execution Result from CM0+

Address	Value to be Written	Description
SRAM_SCRATCH_ADDR		
Bits [31:28]	0xA = SUCCESS/Program command ongoing in background0xF = ERROR	
Bits [27:0]	Error Code	A failure status is indicated by 0xF00000XX

Hint Bar

Review TRM chapter 33 for CYT2B, chapter 41 for CYT3B/4B/6B, chapter 39 for CYT3D/4D, System Calls, and System Call Status for additional details

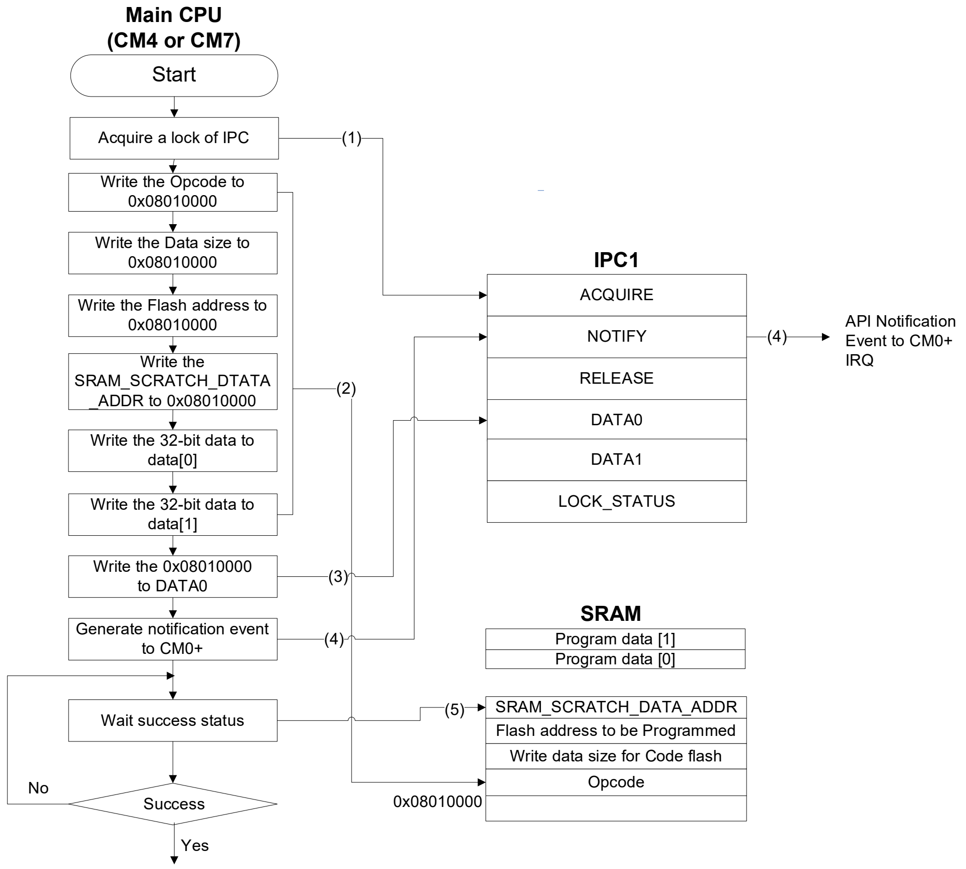
Refer to the Inter-processor Communication training section for additional IPC details

Use case:

API operation (1/3)

- Steps to activate Program Row API flow
  - SRAM_SCRATCH_ADDR = 0x08010000[1]
  - Program Row opcode = 0x06000000

1. Acquire a lock
2. Write the opcode of Program row/Data size/Flash address/SRAM_SCRATCH_DATA_ADDR to SRAM_SCRATCH_ADDR
   Write the 32-bit data (0x55AA55AA) to data[0]
   Write the 32-bit data (0x55AA55AA) to data[1]
3. Write SRAM_SCRATCH_ADDR to DATA0
4. Generate notification event by writing to the IPC_NOTIFY register
5. Wait to be returned success status (0xA)



Main CPU (CM4 or CM7)
Start
Acquire a lock of IPC
Write the Opcode to 0x08010000
Write the Data size to 0x08010000
Write the Flash address to 0x08010000
Write the SRAM_SCRATCH_DTATA _ADDR to 0x08010000
Write the 32-bit data to data[0]
Write the 32-bit data to data[1]
Write the 0x08010000 to DATA0
Generate notification event to CM0+
Wait success status
Success
No
Yes
(1)
(2)
(3)
(4)
(5)
IPC1
ACQUIRE
NOTIFY
RELEASE
DATA0
DATA1
LOCK_STATUS
(4)
API Notification Event to CM0+ IRQ
SRAM
Program data [1]
Program data [0]
SRAM_SCRATCH_DATA_ADDR
Flash address to be Programmed
Write data size for Code flash
Opcode
0x08010000

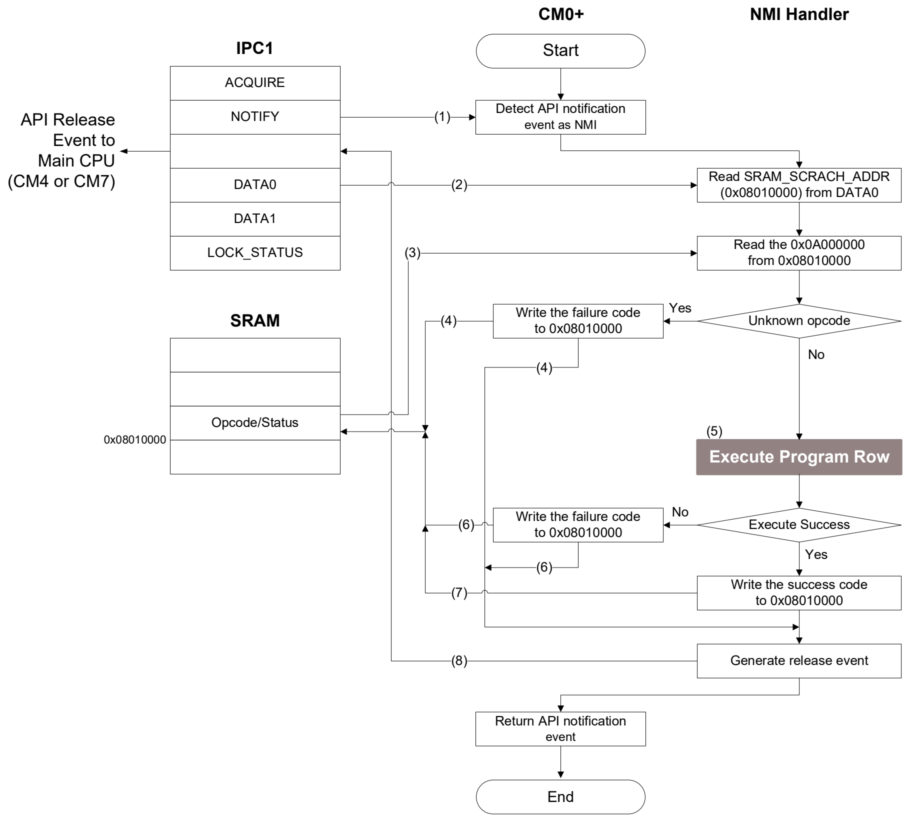
[1] SRAM addresses for the CY2B7/B9 series.

Use case:

API operation (2/3)

› Steps to execute Program Row API flow
  – Execution of API is performed by CM0+
  – SRAM_SCRATCH_ADDR area is used as status code after reading opcode

1. Detect API notification event
2. Read the SRAM_SCRACH_ADDR (0x08010000[1]) from DATA0
3. Read the opcode (0x06000000) from SRAM_SCRATCH_ADDR (0x08010000)
4. If opcode is unknown, write the failure code to SRAM_SCRATCH_ADDR (0x08010000)
5. Execute Program Row
6. If the result is fail, write the failure code to SRAM_SCRATCH_ADDR (0x08010000)
7. Write the success code to SRAM_SCRATCH_ADDR (0x08010000)
8. Generate a release event by writing to the IPC_RELEASE register



CM0+
NMI Handler
IPC1
Start
ACQUIRE
NOTIFY
Detect API notification event as NMI
API Release Event to Main CPU (CM4 or CM7)
DATA0
DATA1
LOCK_STATUS
Read SRAM_SCRACH_ADDR (0x08010000) from DATA0
Read the 0x0A000000 from 0x08010000
SRAM
Write the failure code to 0x08010000
Yes
Unknown opcode
No
Opcode/Status
0x08010000
Execute Program Row
Write the failure code to 0x08010000
No
Execute Success
Yes
Write the success code to 0x08010000
Generate release event
Return API notification event
End
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)

[1] SRAM addresses for the CY2B7/B9 series.

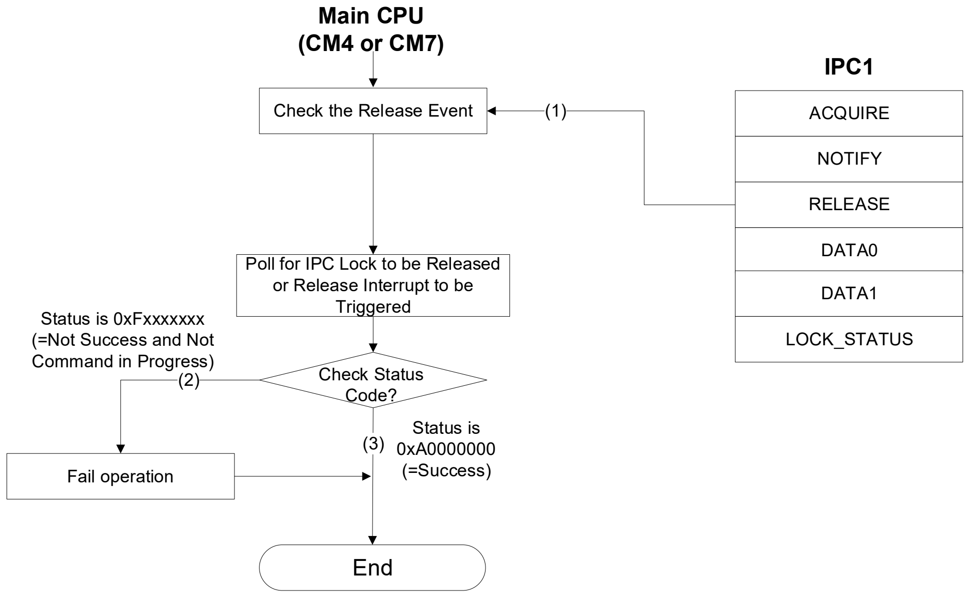
Use case:

API operation (3/3)

› Closed API flow of Program Row API
  – SRAM_SCRATCH_ADDR = 0x08010000[1]
  – Program Row opcode = 0x06000000
  – SRAM_SCRATCH_ADDR area is stored in Status code

1. Detect API release event
2. If the status code is failure, transfer to Fail operation and end API
3. If the status code is success, API ends normally

› Release event can also be reported as an interrupt


Main CPU (CM4 or CM7)
IPC1
ACQUIRE
NOTIFY
RELEASE
DATA0
DATA1
LOCK_STATUS
(1)
Check the Release Event
Poll for IPC Lock to be Released or Release Interrupt to be Triggered
Check Status Code?
Status is 0xFxxxxxxx (=Not Success and Not Command in Progress)
(2)
Fail operation
(3)
Status is 0xA0000000 (=Success)
End

[1] SRAM addresses for the CY2B7/B9 series.

Revision History

Revision	ECN	Submission Date	Description of Change
**	6136844	04/17/2018	Initial release
*A	6409117	12/12/2018	Added the note descriptions. Added CYT2B9 and CYT4BF to Introduction. Updated the Block Diagram in Introduction, Flash Memory Operation with API table and API Summary table. Added eFuse section. Fixed the diagram of P17 (API Operation (2/3); IPC0 to IPC1).
*B	6639127	07/29/2019	Added CYT4DN to the introduction and added information in all sections. Updated page 2. Added Program Row API use case in Example of Flash Memory Operation with API in pages 20 to 23.
*C	7072326	01/21/2021	Updated page 2, 3. Removed "Inject Public Key", "Write Normal Access Restriction", "Write TOC2“, “EnterFlashMarginMode” and “ExitFlashMarginMode” APIs. Added “SwitchOverRegulators”, “ConfigureRegulator”, “DebugPowerUpDown”, “LoadRegulatorTrims”, and “OpenRMA” APIs.

Revision History (contd.)

Revision	ECN	Submission Date	Description of Change
*D	8016631	03/25/2024	Updated page 2, 4 for CYT6BJ Added page 10, 14 Updated page 15 to 19 for CYT6BJ