AURIX 2G Architecture

Sherry Li
IFCN ATV SMD GC SAE MC

Agenda

1 Architectural Overview

2 Memory Concept

3 On-chip Bus Connectivity

4 NVM System

5 TC1.6.2 Architecture

6 Summary

Agenda

1 Architectural Overview

2 Memory Concept

3 On-chip Bus Connectivity

4 NVM System

5 TC1.6.2 Architecture

6 Summary

AURIX™ TC3xx Architecture Evolution
(enhancements to AURIX TC2xx)

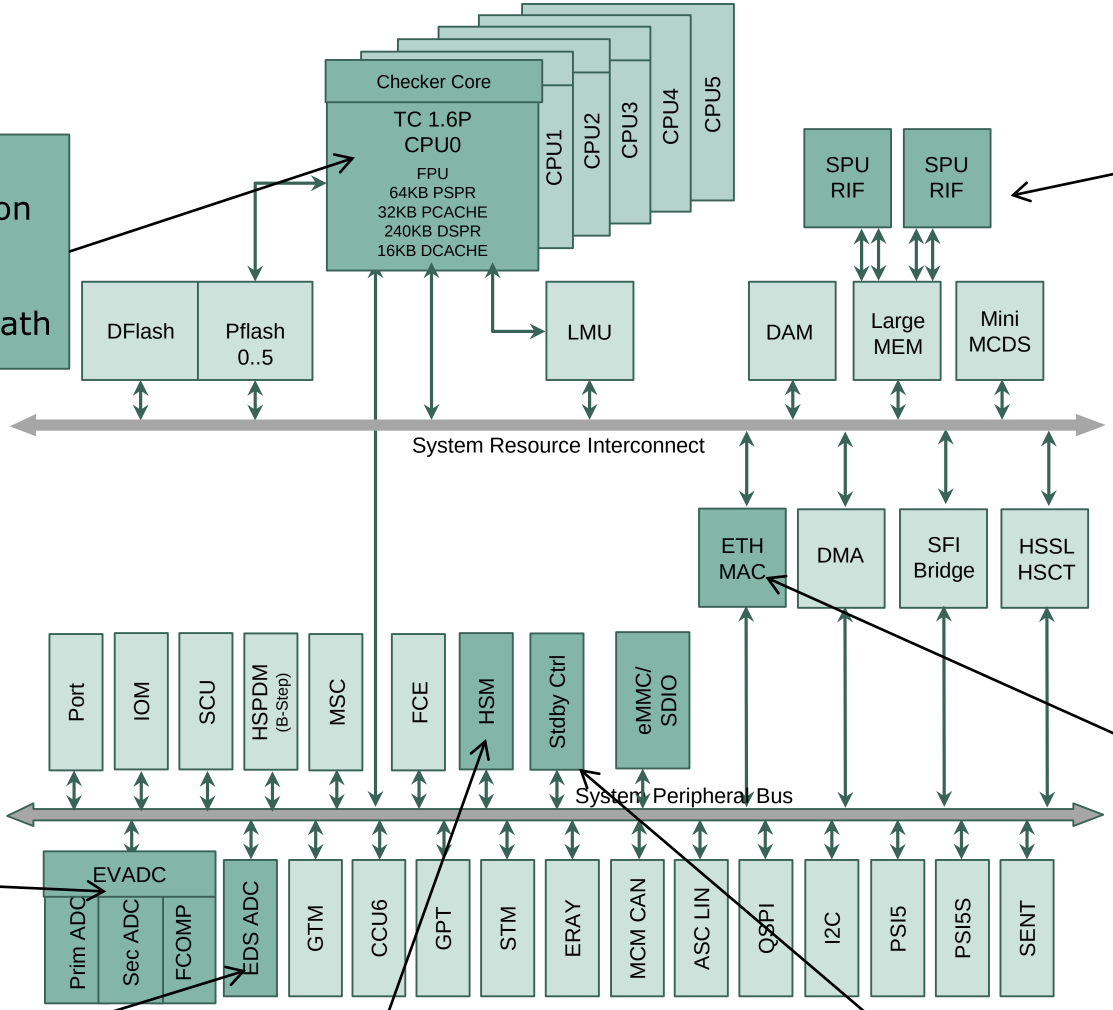
INNOVATION

Performance
- New TriCore 162 generation
- New instructions
- up to 6 CPUs @300MHz
- New direct Flash access path

Memories
- Larger SRAM
- SRAM/Flash ratio increased
- enhanced MPU

ADC
- Improvement of existing ADC
- Reduction of capacitive load

Delta-Sigma: enhanced concept



HSM: Full Evita compliance
- New accelerators ECC256 / SHA256
- Available on all devices

Standby Control Unit
- Low power modes

ADAS
New SPU concept

Safety
- LBIST
- MBIST upgrade

Ethernet
- 1GBit/s ETH
- QoS services

IO Pads
all 5V/3.3V

AURIX TC3xx Feature Table

Feature Set		9x Series eXtension (16MB)	8x Series (10MB)	7x Series eXtended (6MB)	7x Series (6MB)	6x Series (4MB)	5x Series (4MB)	3x Series +eXtension (2MB)	3x Series (2MB)	2x Series (1MB)
TriCore 1.6	# Cores / Checker	6/4	4/2	3/3	3/2	2/2	3/2	2/1	1/1	1/1
	Frequency	300MHz	300MHz	300MHz	300MHz	300MHz	300MHz	200MHz	200MHz	160MHZ
Accelerator	Signal processing Unit (SPU)	2xSPU					2xSPU	1xSPU		
Flash	Program Flash	16MB	10MB	6MB	6MB	4MB	4MB	2MB	2MB	1MB
	Data Flash (physical/logical)	1024kB	512kB	256kB	256kB	128kB	128kB	128kB	128kB	96kB
SRAM	Total (DMI , PMI, LMU, AMU)	6912KB	1568kB	4208kB	1136kB	672kB	3152kB	1328kB	248kB	152kB
DMA	Channels	128	128	128	128	64	64	16	16	16
ADC	Modules Primary / Sec / FC / DS	8/4/8/14	8/4/4/10	4/4/2/6	4/4/2/6	4/2/2/4	2/0/0/0	4/2/0/0	2/2/0/0	2/2/0/0
	Channels Primary / Sec / FC /DS	64/64/8/14	64/64/4/10	32/64/2/6	32/64/2/6	32/32/2/4	16/0/0/0	>16/32/0/0	16/32/0/0	16/32/0/0
Timer	GTM TIM / (A)TOM / MCS	64 / 192 / 10	56 / 152 / 7	40 / 88 / 5	40 / 88 / 5	24 / 64 / 3	-	16 / 32 / 0	16 / 32 / 0	16 / 32 / 0
	CCU / GPT modules / bit streaming	2/1/1	2/1/0	2/1/0	2/1/-	2/1/0	2/1/1	2/1/1	2/1/0	2/1/0
Interfaces	FlexRay (#/ch.)	2 /4	2/4	1/2	1/2	1/2	1/2	1/2	1/2	0/0
	CAN-FD / TT	12/1	12/1	12/1	8/1	8/1	8/0	8/0	8/0	6/0
	QSPI / ASCLIN / I2C	6 /12/2	5 /24/2	5/12/1	5/12/1	4/12/1	4/4/0	4/12/0	4/12/0	4/6/0
	SENT / PSI5 / PSI5S	25/4/1	25/4/1	15/2/1	15/2/1	10/2/1	0/0/0	6/0/0	6/0/0	6/0/0
	HSSL / MSC / EBU	2/4/1	1/3/0	1/2/0	1/2/0	1/1/0	0/0/0	0/0/0	0/0/0	0/0/0
	Ethernet 100Mbps/1Gbps	1/1	1/1	2/2*	1/1	1/1	1/1	1/1	-/-	-/-
	eMMC/SDIO	1/1		1/1				1/1		
	Radar /ext. ADC IF (RIF)	12x400Mbps LVDS	-	-	-	-	12x400Mbps LVDS	6x100Mbps LVDS	-	-
	Camera IF (CIF)	-	-	1	-	-	-	-	-	-
Security	HSM	HSM+ECC256	HSM+ECC256	HSM+ECC256	HSM+ECC256	HSM+ECC256	HSM+ECC256	HSM+ECC256	HSM+ECC256	HSM+ECC256
Safety	SIL Level	ASIL D	ASIL D	ASIL D	ASIL D	ASIL D	ASIL D	ASIL D	ASIL D	ASIL D
Power	EVR	Yes (3.3V/5V)	Yes (3.3V/5V)	Yes (3.3V/5V)	Yes (3.3V/5V)	Yes (3.3V/5V)	Yes (3.3V/5V)	Yes (3.3V/5V)	Yes (3.3V/5V)	Yes (3.3V/5V)
	Standby Control Unit	yes	yes	yes	yes	yes	yes	yes	yes	yes

* In discussion

Agenda

1 Architectural Overview

2 Memory Concept

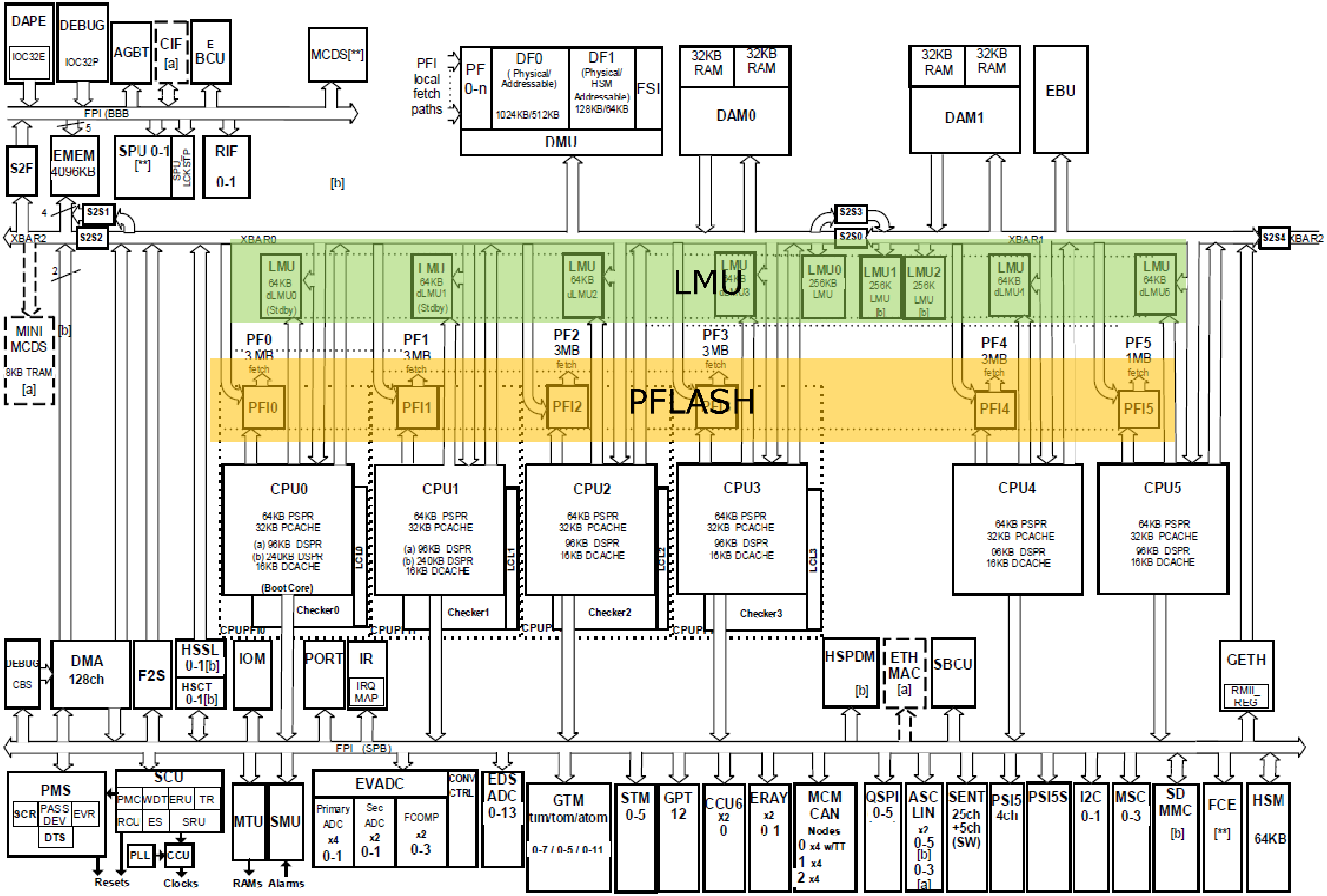
3 On-chip Bus Connectivity

4 NVM System

5 TC1.6.2 Architecture

6 Summary

Local Memory Access with Global Contiguous Mapping



([a] = module available only in A-step. [b] = module not available in TC39x A-Step, [**] = Module has significantly different functionality in TC39x A-Step. See chapter for details)

AURIX™ TC3xx Memory Concept

Local And Global Memory

- Enhanced protection in AURIX™ - TC3xx
  All memory data and addresses and all accesses to all memories are protected
- Enhanced local and global memory concept
  - Local memories are providing enhanced performance to the specified CPU:
    PSPR, DSPR, DLMU, PFLASH, (Data Cache, Program Cache)
  - All CPUs can access all local memories of other CPUs with “global” performance
  - Global memories have no preferences: global LMU, DAM, EMEM, D-Flash
    All CPUs can access all global memories with “global” performance
- memory access performance

local access wait states

access type	data read	data write	code fetch
local PSPR	6	7	0
local DSPR	0	0	6
local DLMU	1	2	7
local PFLASH	3+Flash		2+Flash

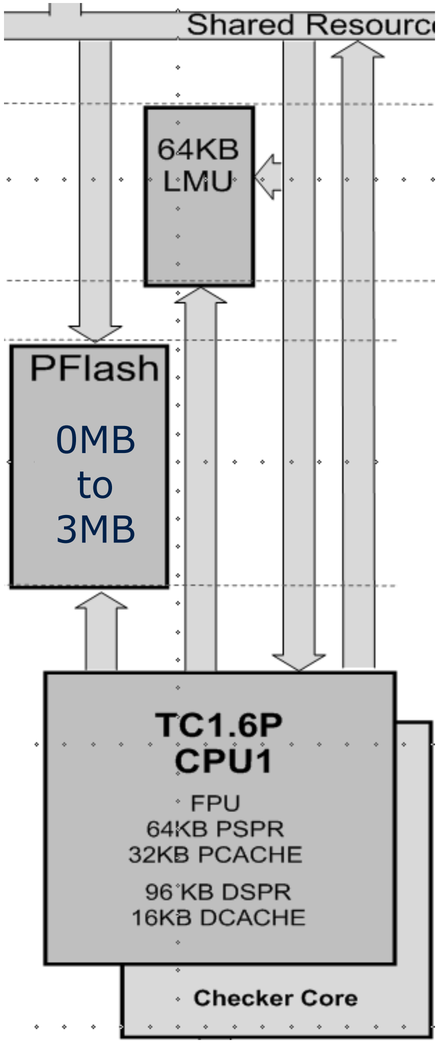
global access wait states

access type	data read	data write	code fetch
other PSPR	6	7	6
other DSPR	6	6	6
other local DLMU	7	8	7
other PFLASH	9+Flash	N.A.	8+Flash
DFLASH	8+Flash		N.A.
global LMU, DAM	7	8	7
EMEM	18	10	18

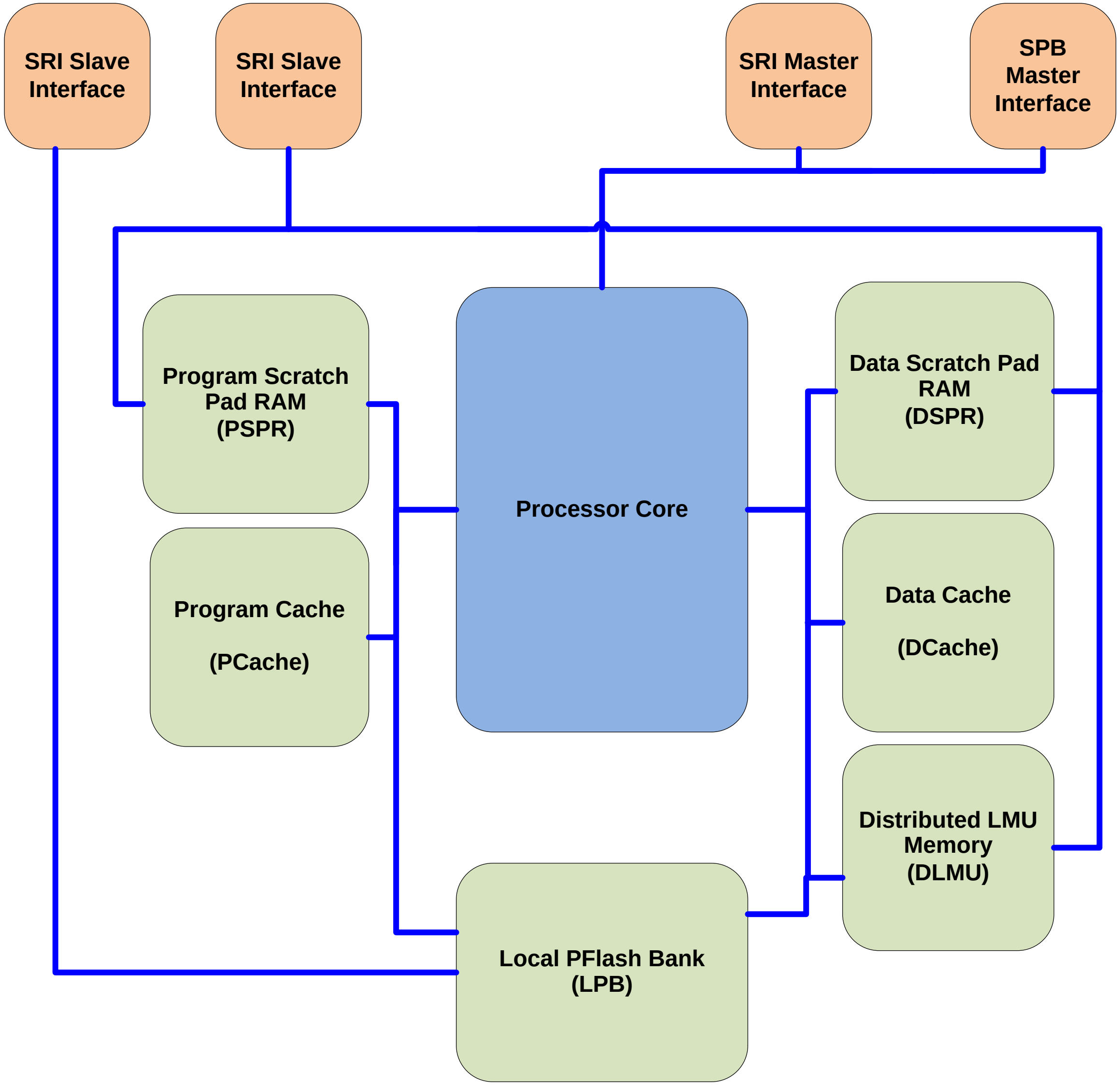
Enhanced Local Memory Concept

For Tricore CPUs

- All PFLASH modules are mapped in a continuous address space starting at 8000 0000 / A000 0000
- Local DLMU and global LMU, are mapped in a continuous address space starting at 9000 0000
- All CPUs will provide the same local memory structures
  - 64kB PSPR: 0WS for code fetch
  - 32kB Program Cache
  - 96 to 240kB DSPR: 0WS for data access
  - 16kB Data Cache
  - 64kB local DLMU : 1(2)WS for data read (write)
  - Local path to one PFLASH bank with up to 3MB



CPU Core Local Memories and Connectivity



Memory Map

Seg.	Address Range	Memory Type	
0	0x00000000 – 0x0FFFFFFF	Reserved	
1	0x10000000 – 0x1FFFFFFF	CPU5 DSPR, DCACHE, DTAG, CPU5 PSPR, PCACHE, PTAG	
2	0x20000000 – 0x2FFFFFFF	Reserved	
3	0x30000000 – 0x3FFFFFFF	CPU4 DSPR, DCACHE, DTAG, CPU4 PSPR, PCACHE, PTAG	
4	0x40000000 – 0x4FFFFFFF	CPU3 DSPR, DCACHE, DTAG, CPU3 PSPR, PCACHE, PTAG	
5	0x50000000 – 0x5FFFFFFF	CPU2 DSPR, DCACHE, DTAG, CPU2 PSPR, PCACHE, PTAG	
6	0x60000000 – 0x6FFFFFFF	CPU1 DSPR, DCACHE, DTAG, CPU1 PSPR, PCACHE, PTAG	
7	0x70000000 – 0x7FFFFFFF	CPU0 DSPR, DCACHE, DTAG, CPU0 PSPR, PCACHE, PTAG	

Memory Map

Seg.	Address Range	Memory Type	
8	0x80000000 – 0x802FFFFF	Program Flash 0 (PF0, 3MB),	
	0x80300000 – 0x805FFFFF	Program Flash 1 (PF1, 3MB),	
	0x80600000 – 0x808FFFFF	Program Flash 2 (PF2, 3MB),	
	0x80900000 – 0x80BFFFFF	Program Flash 3 (PF3, 3MB),	
	0x80C00000 – 0x80EFFFFF	Program Flash 4 (PF4, 3MB),	
	0x80F00000 – 0x80FFFFFF	Program Flash 5 (PF5, 1MB),	
	0x81000000 – 0x811FFFFF	Reserved (for PFLASH, 2MB),	
	0x82000000 – 0x8FFFFFFF	EBU, OLDA	
9	0x90000000 – 0x9000FFFF	CPU0 DLMU (64KB),	
	0x90010000 – 0x9001FFFF	CPU1 DLMU (64KB),	
	0x90020000 – 0x9002FFFF	CPU2 DLMU (64KB),	
	0x90030000 – 0x9003FFFF	CPU3 DLMU (64KB),	
	0x90040000 – 0x9007FFFF	LMU0 LMURAM (256KB),	
	0x90080000 – 0x900BFFFF	LMU1 LMURAM (256KB),	
	0x900C0000 – 0x900FFFFF	LMU2 LMURAM (256KB),	
	0x90100000 – 0x9010FFFF	CPU4 DLMU (64KB),	
	0x90110000 – 0x9011FFFF	CPU5 DLMU (64KB),	
	0x90400000 – 0x90407FFF	DAM0 RAM0 (32KB),	
	0x90408000 – 0x9040FFFF	DMA0 RAM1 (32KB),	
	0x90410000 – 0x90417FFF	DAM1 RAM0 (32KB),	
	0x90418000 – 0x9041FFFF	DAM1 RAM1 (32KB),	
	0x90420000 – 0x9FFFFFFF	MINIMCDS Trace RAM (TRAM),	
		EMEM	

Memory Map

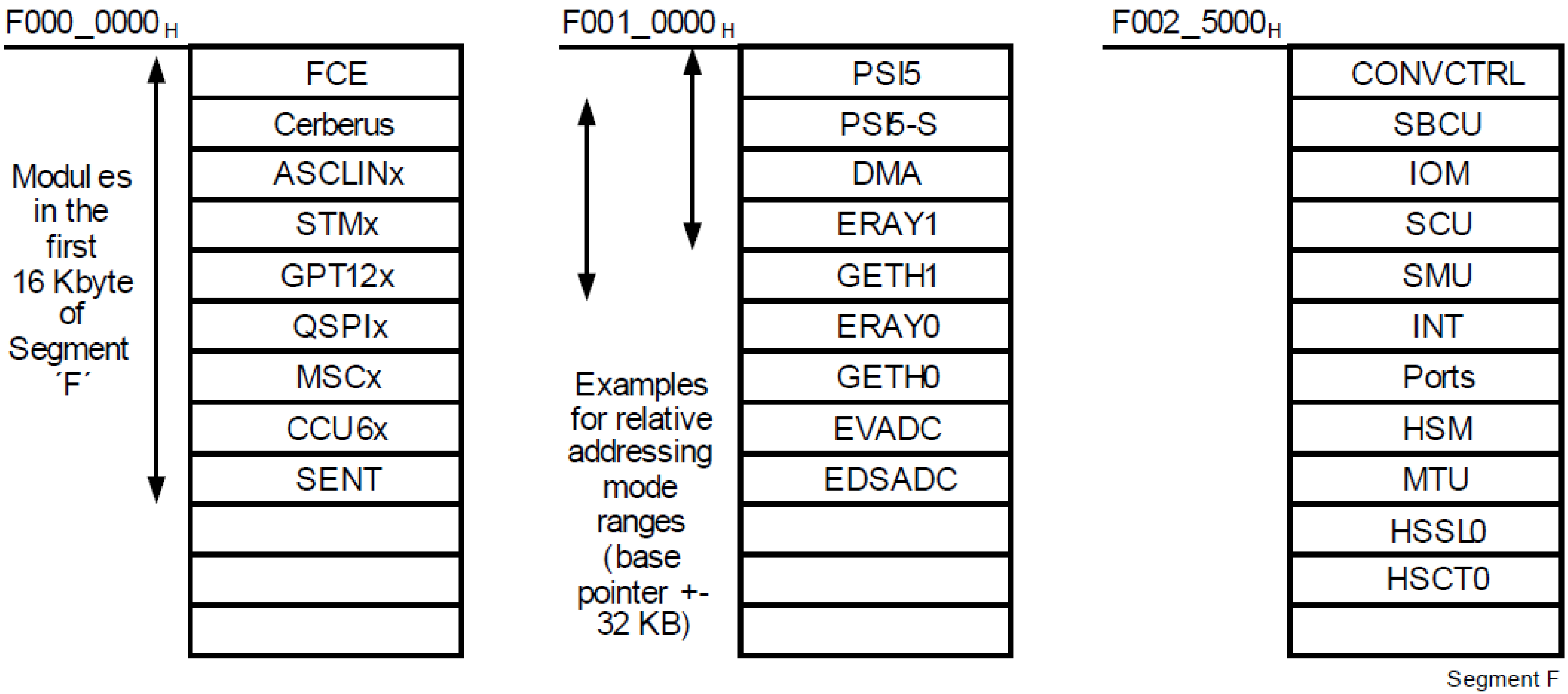
Seg.	Address Range	Memory Type	
10	0xA0000000 – 0xA02FFFFF	Program Flash 0 (PF0, 3MB),	
	0xA0300000 – 0xA05FFFFF	Program Flash 1 (PF1, 3MB),	
	0xA0600000 – 0xA08FFFFF	Program Flash 2 (PF2, 3MB),	
	0xA0900000 – 0xA0BFFFFF	Program Flash 3 (PF3, 3MB),	
	0xA0C00000 – 0xA0EFFFFF	Program Flash 4 (PF4, 3MB),	
	0xA0F00000 – 0xA0FFFFFF	Program Flash 5 (PF5, 1MB),	
	0xA1000000 – 0xA11FFFFF	Reserved (for PFLASH, 2MB),	
	0xA2000000 – 0xAFFFFFFF	Erase Counter 0 (EC0, 16KB),	
		PFI User Register 0 (PFI0, 256KB),	
		Erase Counter 1 (EC0, 16KB),	
		PFI User Register 1 (PFI0, 256KB),	
		Erase Counter 2 (EC0, 16KB),	
		PFI User Register 2 (PFI0, 256KB),	
		Erase Counter 3 (EC0, 16KB),	
		PFI User Register 3 (PFI0, 256KB),	
		Erase Counter 4 (EC0, 16KB),	
		PFI User Register 4 (PFI0, 256KB),	
		Erase Counter 5 (EC0, 16KB),	
		PFI User Register 5 (PFI0, 256KB),	
	0xAF000000 – 0xAF0FFFFF	Data Flash 0 EEPROM (DF0, 1MB),	
	0xAF400000 – 0xAF405FFF	Data Flash 0 UCB (DF0, 24KB),	
	0xAF800000 – 0xAF80FFFF	Data Flash 0 CFS (DF0, 64KB),	
	0xAFC00000 – 0xAFC3FFFF	Data Flash 1 EEPROM (DF1, 128KB),	
	0xAFE00000 – 0xAFFFFFFF	EBU, OLDA, BROM	

Memory Map

Seg.	Address Range	Memory Type	
11	0xB0000000 – 0xB000FFFF 0xB0010000 – 0xB001FFFF 0xB0020000 – 0xB002FFFF 0xB0030000 – 0xB003FFFF 0xB0040000 – 0xB007FFFF 0xB0080000 – 0xB00BFFFF 0xB00C0000 – 0xB00FFFFF 0xB0100000 – 0xB010FFFF 0xB0110000 – 0xB011FFFF 0xB0400000 – 0xB0407FFF 0xB0408000 – 0xB040FFFF 0xB0410000 – 0xB0417FFF 0xB0418000 – 0xB041FFFF 0xB0420000 – 0xBFFFFFFF	CPU0 DLMU (64KB), CPU1 DLMU (64KB), CPU2 DLMU (64KB), CPU3 DLMU (64KB), LMU0 LMURAM (256KB), LMU1 LMURAM (256KB), LMU2 LMURAM (256KB), CPU4 DLMU (64KB), CPU5 DLMU (64KB), DAM0 RAM0 (32KB), DMA0 RAM1 (32KB), DAM1 RAM0 (32KB), DAM1 RAM1 (32KB), MINIMCDS Trace RAM (TRAM), EMEM, Extra Trace Memory (XTM)	
12	0xC0000000 – 0xCFFFFFFF	Reserved	
13	0xD0000000 – 0xDFFFFFFF	Reserved	
14	0xE0000000 – 0xEFFFFFFF	Reserved	
15	0xF0000000 – 0xFFFFFFFF	Peripheral Registers	

Memory Map

- Segment 15



F000_0000 H

Modules in the first 16 Kbyte of Segment ´F´

- FCE
- Cerberus
- ASCLINx
- STMx
- GPT12x
- QSPIx
- MSCx
- CCU6x
- SENT

F001_0000 H

Examples for relative addressing mode ranges (base pointer +- 32 KB)

- PSI5
- PSI5-S
- DMA
- ERAY1
- GETH1
- ERAY0
- GETH0
- EVADC
- EDSADC

F002_5000 H

- CONVCTRL
- SBCU
- IOM
- SCU
- SMU
- INT
- Ports
- HSM
- MTU
- HSSL0
- HSCT0

Segment F

Agenda

1 Architectural Overview

2 Memory Concept

3 On-chip Bus Connectivity

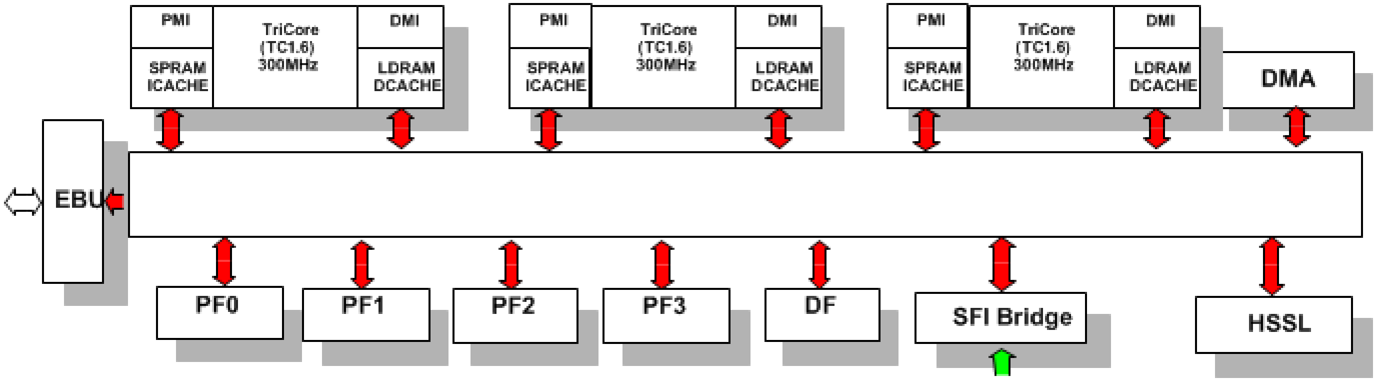
4 NVM System

5 TC1.6.2 Architecture

6 Summary

AURIX 1st Gen used a single crossbar

TC29X SRI Masters and Slaves

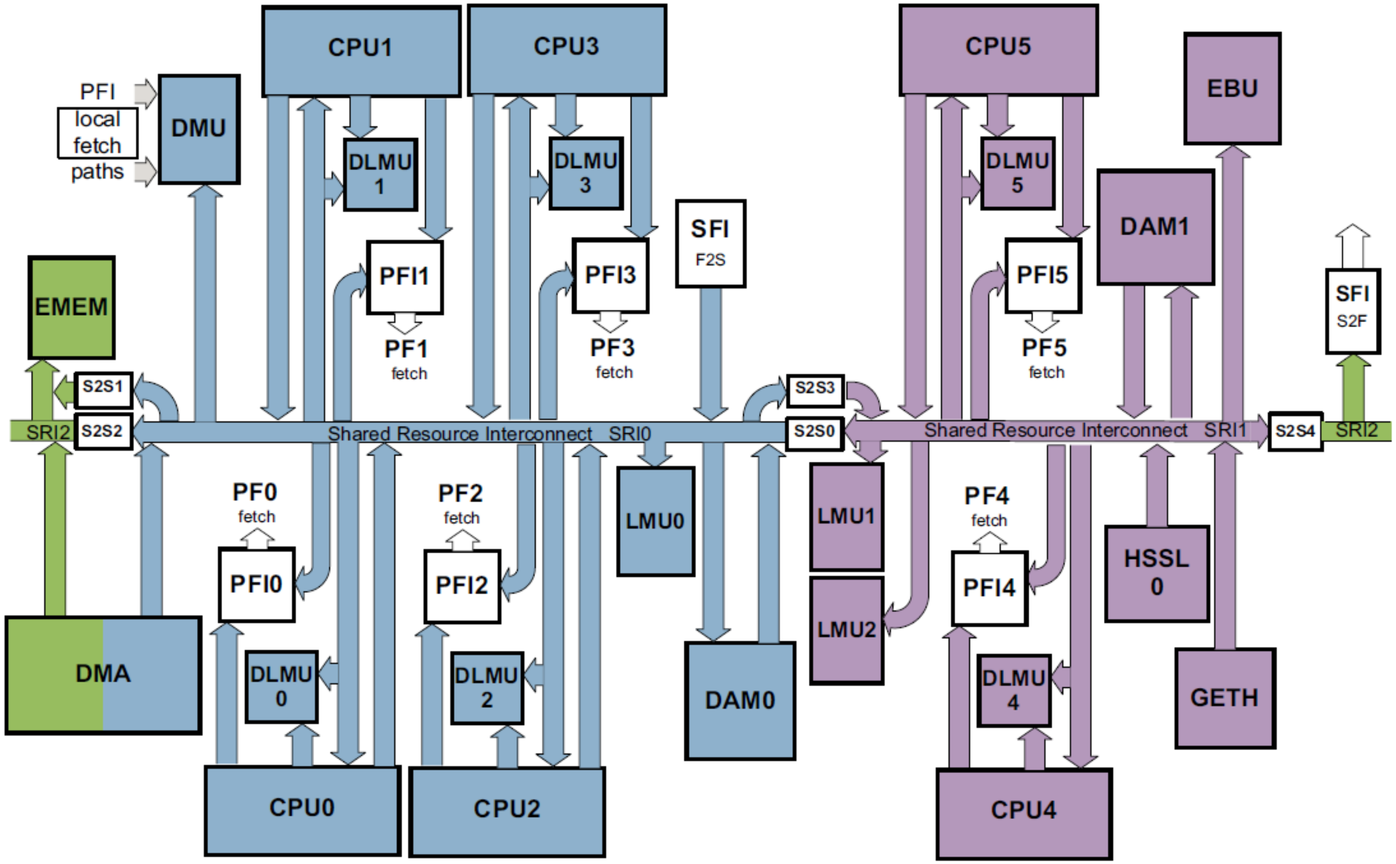


- TC29X supports 10 masters and 11 slaves with uniform access characteristics
  - Equally fast ☺ or
  - Equally slow ☹
- One cycle extra latency for masters than in TC27X

- 2nd Gen needs to support more high performance agents
  - Scaling a single crossbar would harm latency uniformly

System Resource Interconnect (SRI)

- Domain 0 with 4 CPUs, Domain 1 with 2 CPUs,
- Domain 2 with ADAS and debug functionality.
- S2S bridges are special in that they are present in two domains.



SRI Connectivity

- Key Changes
  - Multiple crossbar domains implemented for high-end TC3xx family members, v.s. single crossbar only for first generation
    - Access latencies vary more with multiple crossbar domains than in AURIX
    - However, latencies are typically the same or lower within a single crossbar (compared to AURIX TC29X)
  - Simplified SRI arbitration scheme, two-priority round robin (high priority and low priority rounds)
    - The configuration is easier and no starvation is possible.
- Minor changes
  - Removed OCDS Level1 debug capture at SRI slaves

SRI Master Request Latency

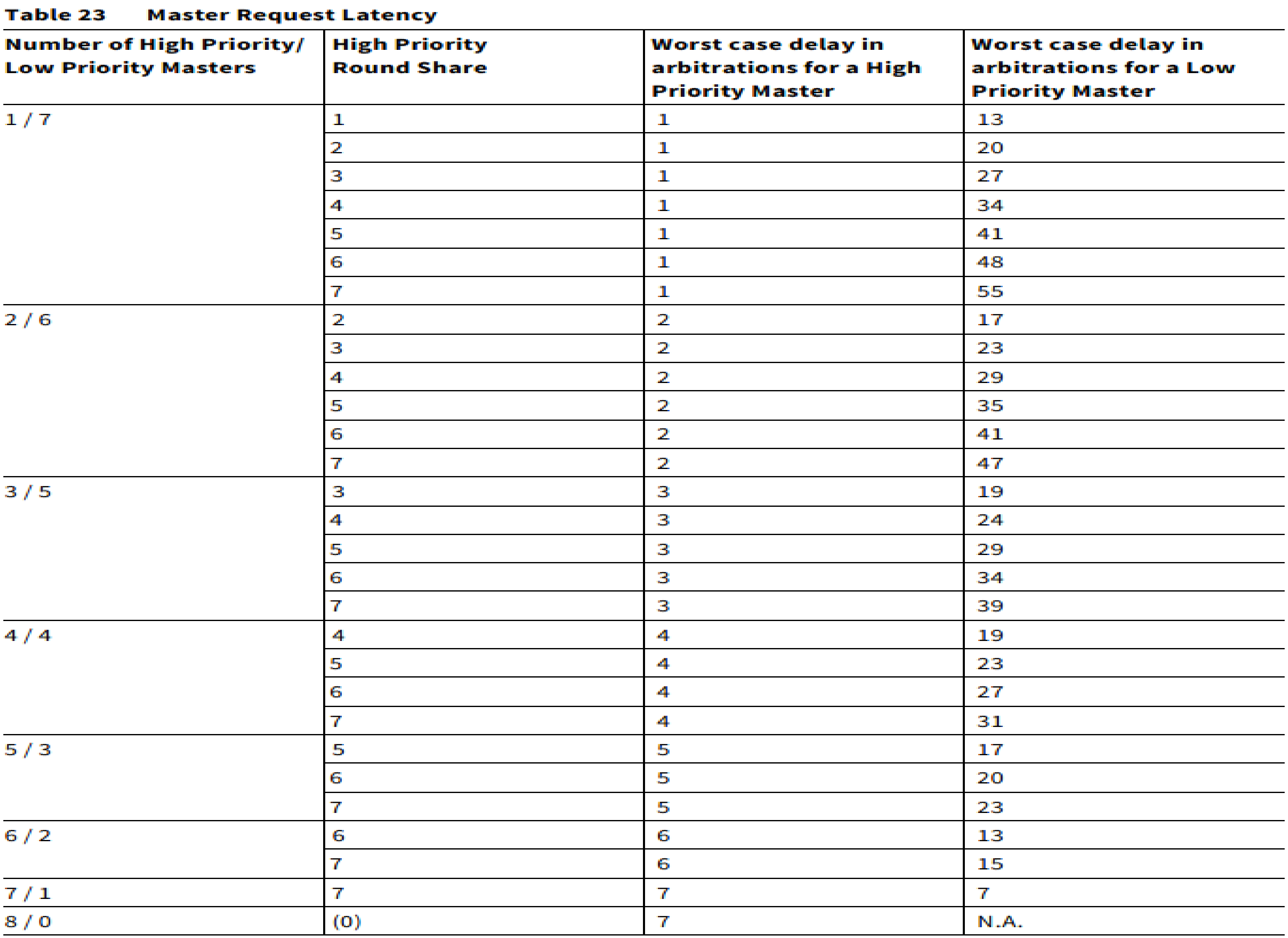
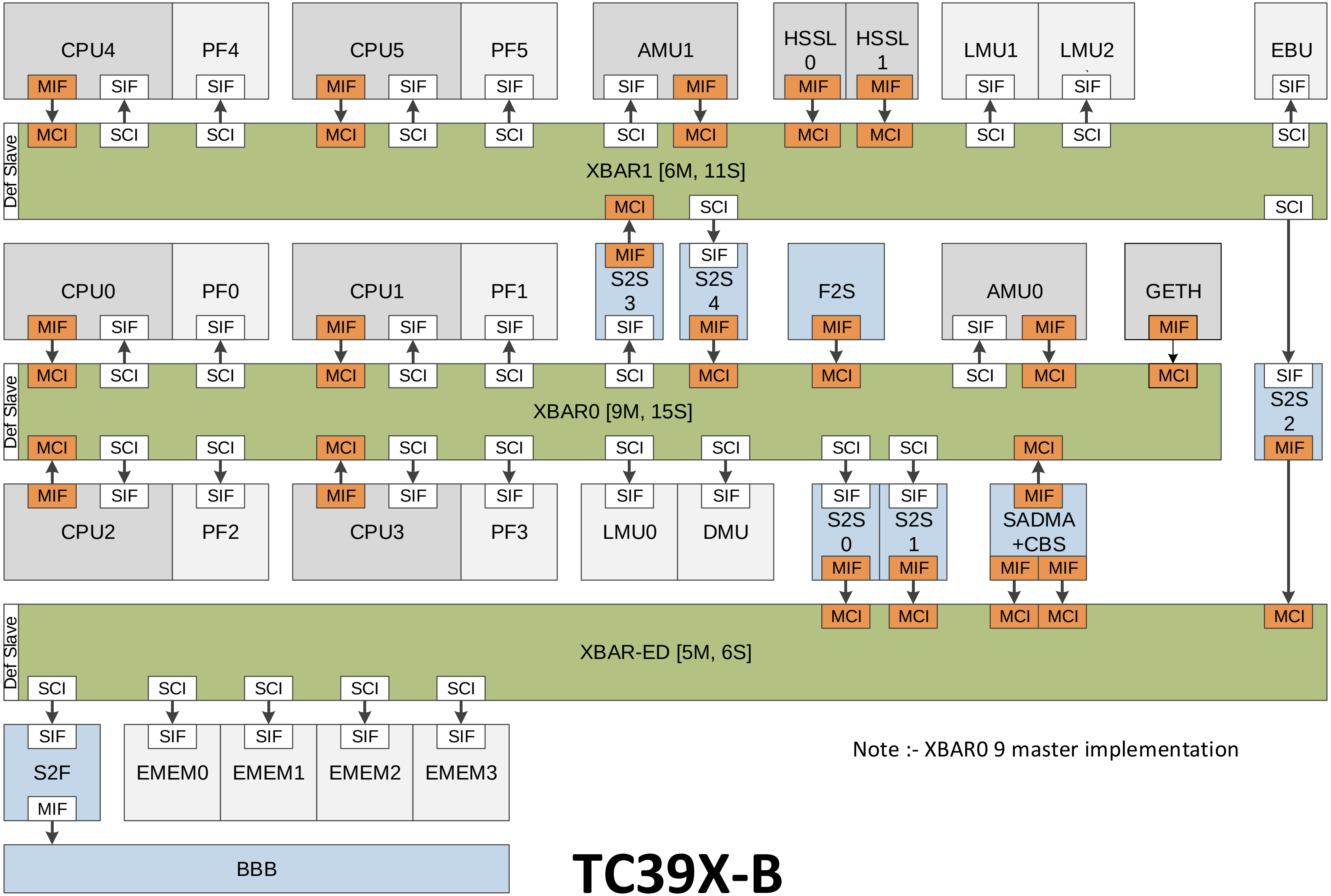


Table 23 Master Request Latency

Number of High Priority/ Low Priority Masters	High Priority Round Share	Worst case delay in arbitrations for a High Priority Master	Worst case delay in arbitrations for a Low Priority Master
1 / 7	1	1	13
	2	1	20
	3	1	27
	4	1	34
	5	1	41
	6	1	48
	7	1	55
2 / 6	2	2	17
	3	2	23
	4	2	29
	5	2	35
	6	2	41
	7	2	47
3 / 5	3	3	19
	4	3	24
	5	3	29
	6	3	34
	7	3	39
4 / 4	4	4	19
	5	4	23
	6	4	27
	7	4	31
5 / 3	5	5	17
	6	5	20
	7	5	23
6 / 2	6	6	13
	7	6	15
7 / 1	7	7	7
8 / 0	(0)	7	N.A.

T39x View on Interconnect



Note :- XBAR0 9 master implementation

TC39X-B

Agenda

1 Architectural Overview

2 Memory Concept

3 On-chip Bus Connectivity

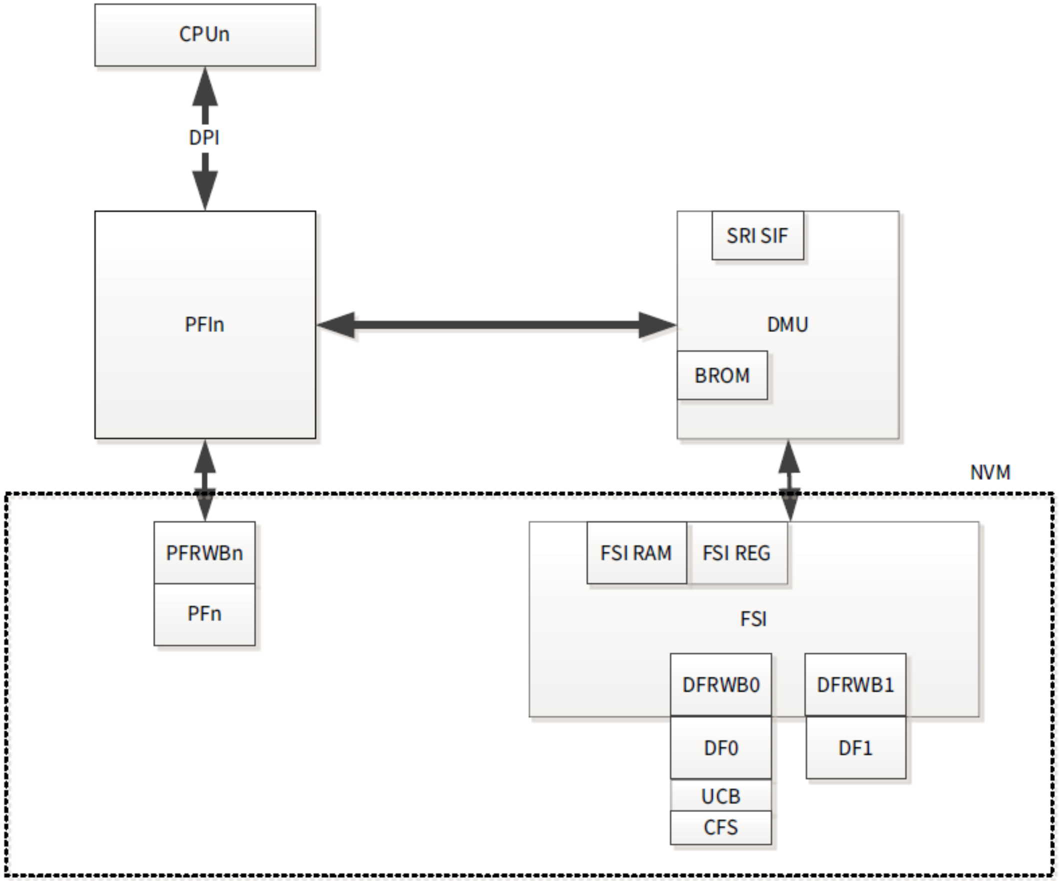
4 NVM System

5 TC1.6.2 Architecture

6 Summary

TC3xx NVM Subsystem

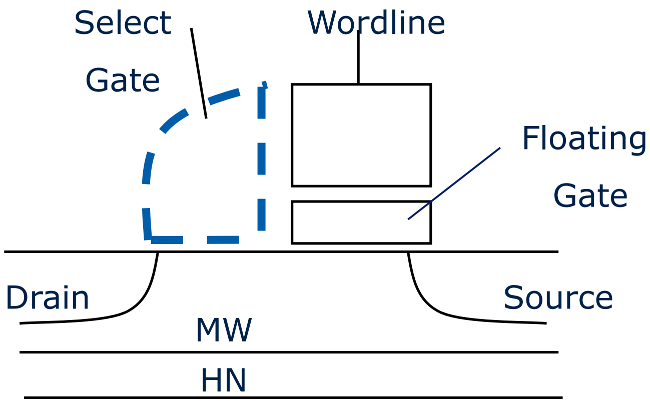
- Data Memory Unit (DMU)
  - Controls command sequences executed on all program and data flash memories.
- Flash Standard Interface (FSI):
  - Executes erase, program and verify operations on all flash memories.
- Program Flash Interface (PFI)
  - Each PFLASH bank has a unique point-to-point fast connection to a CPU.
- PFlash Read Write Buffer (PFRWB)
  - Performs the ECC correction and detection and provides the read data to the system.



NVM Major Changes from AURIX to A2G

- Dedicated performance interface for PFLASH read (PFI)
- New Flash structure and Flash sizes
- HSM PCODE sectors in PF0 increased
- Configuration Sector (CFS) moved to DFLASH
- Erase Counter moved to the respective PFLASH
- All Flash banks, CFS, UCB, Erase Counters and registers have separate system addresses
- Complement Sensing mode for DFLASH0/1 EEPROM
- New UCBs and Dual UCB concept
- Protection configuration in UCB updated to match new Flash structure
- New command sequences
- New power mode - Cranking mode
- New registers, and re-organisation of existing registers to accomodate architecture changes
- No legacy ECC for PFLASH, only safe ECC
- Requested Read feature removed
- DFLASH1 register access protection - change in definition of access term ‘H’
- Support for Software update Over the Air (SOTA)
- High and Low priority Erase Counter regions
- New functional safety features

AURIX™ - TC3xx Flash



Aurix Flash Cell

HS3P – Hot Source 3 Poly

Program Flash Parameters

- Programming Granularity: 32 Byte min 1MB/s (SOTA)
- Access time: 30ns
- Erase Time (Sector range): ~ 1sec min 1MB/s (increased from 0.5MB/s)
- Erase Granularity: **Sector 16KB**
- Data Retention: 20y for 1k w/e
- Erase suspend + resume functions
- Erase suspend time <120us
- ECC: 2bit correction, >3bit detection

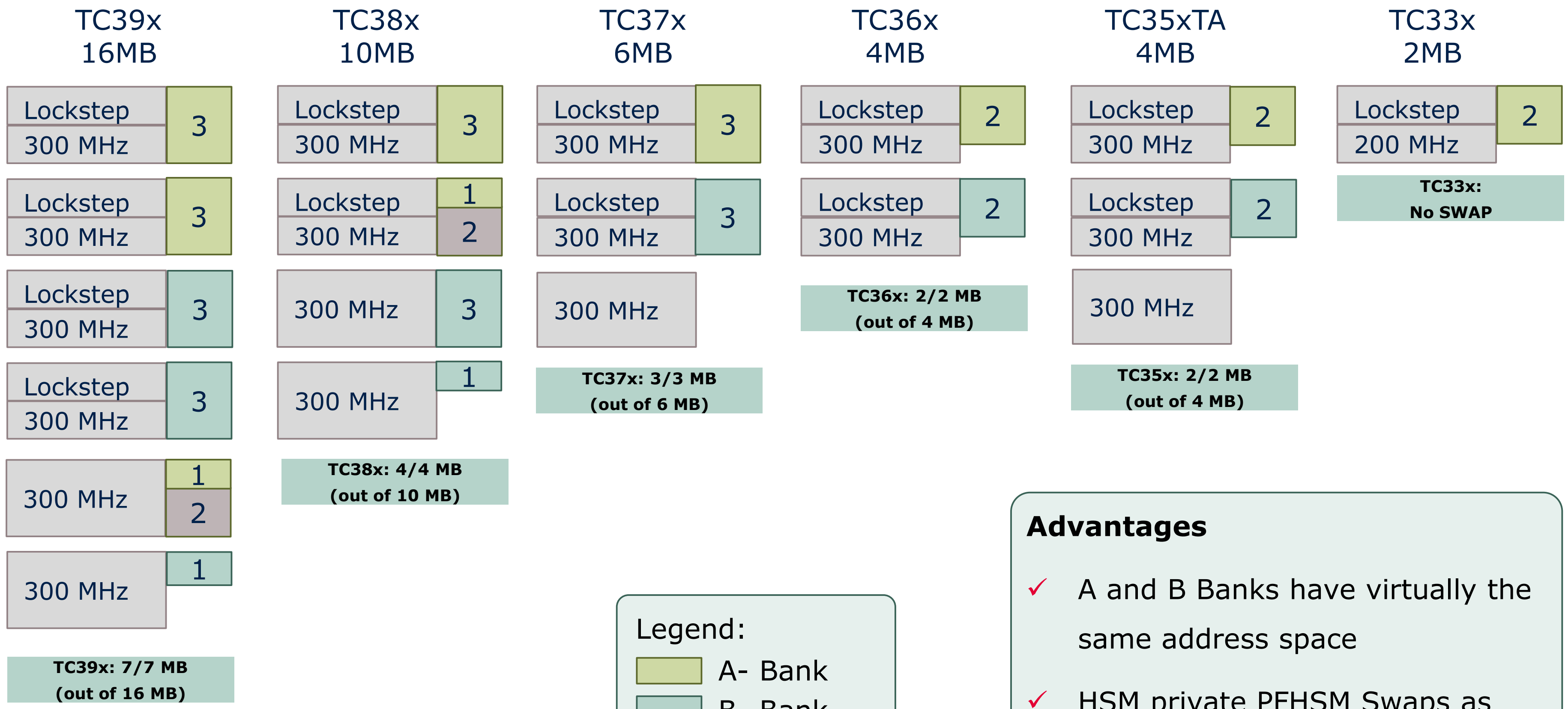
Data Flash Parameters

- Endurance = 500k (125k)
- Programming Time ~ 75us
- Programming Granularity: 8 Byte
- Access time: 100ns
- Erase Time (Sector range): ~ 1sec
- Erase Granularity: Sector 8kB
- Data Retention 10y
- Erase suspend + resume functions
- Erase suspend time <120us
- ECC: 3bit correction, >4bit detection

Subject to change

AURIX™ TC3xx A/B SWAP Capability

All TC3xx devices besides the TC33x and TC33xED have the ability to support Software updates Over The Air (SOTA).



Legend:

- 🟨 A- Bank
- 🟩 B- Bank
- 🟫 No Swap

Advantages

- ✓ A and B Banks have virtually the same address space
- ✓ HSM private PFHSM Swaps as well

PFlash Bank Differences AURIX vs. AURIX TC3xx

	AURIX	AURIX TC3xx
Max Bank Size	2MB	3MB
Max Physical Sector Size	512KB	1MB
Max Erase Size	512KB	512KB
Max Erase Timing	1 s	0.5 s
Prog Throughput	1 MByte/s*	1 MByte/s*
Wordline Size	512 Byte	1024 Byte
SenseAmp	Drain Side Current Sensing	Source Side Bitline Integration Scheme
Erase Counter	in DFlash/UCB	separate log 16kByte sector
Config Sector	logic sectors in PFlash	separate physical sector in DFlash (combined with UCB)
Logical Sectors	16 .. 256 kByte	16 kByte

* assuming using 5V supply and prog burst mode

AURIX™ TC3xx Data Flash Options
Single Ended Sensing / Complement Sensing

Two options for operating the Data FLASH: single ended sensing and complement sensing

- configured in user settings - no dynamic change!
- software driver adaptation required (MCAL will support single ended first)

	single ended sensing	**complement sensing**
endurance	125k cycles	500k cycles
Flash cells per bit	1	2

	TC39x (16MB)	**TC38x (10MB)**	**TC37x (6MB)**	**TC36x (4MB)**	**TC35x (4MB)**	**TC33x (2MB)**
Data Flash Single Ended Sensing	1024kB	512kB	256kB	128kB	128kB	128kB
Data Flash Complement Sensing	512kB	256kB	128kB	64kB	64kB	64kB

Subject to change

DataFlash Differences AURIX vs. AURIX TC3xx

	AURIX	AURIX TC3xx Single Ended	AURIX TC3xx Complement
Cell Endurance	125k p/e cycles	125k p/e cycles	500k p/e cycles (250k for HSM)
Wordline Size	512 Byte	512 Byte	256 Byte
Number of Erases* for DF0(EE) DF1(HSM)	6 x 125k 4 x 125k	6 x 125k 4 x 125k	8 x 500k 8 x 250k
Invalidation Prog with All-1	Possible but not used	Possible but not used	Limited use*
Erased DataFlash	ECC valid	ECC valid	- Arbitrary Read result - in most cases not ECC valid Blank Check - Signal available
Erase Time	~ 1s	0.5s (<1k cycles) 1.5s (< 125k cy.)	0.5s (<1k cycles) 1.5s (< 500k cy.) + adder: ~50ms / 32 kB[1]
Prog Current	s. datasheet	s. datasheet	+ adder of ~20%[2]

Notes:

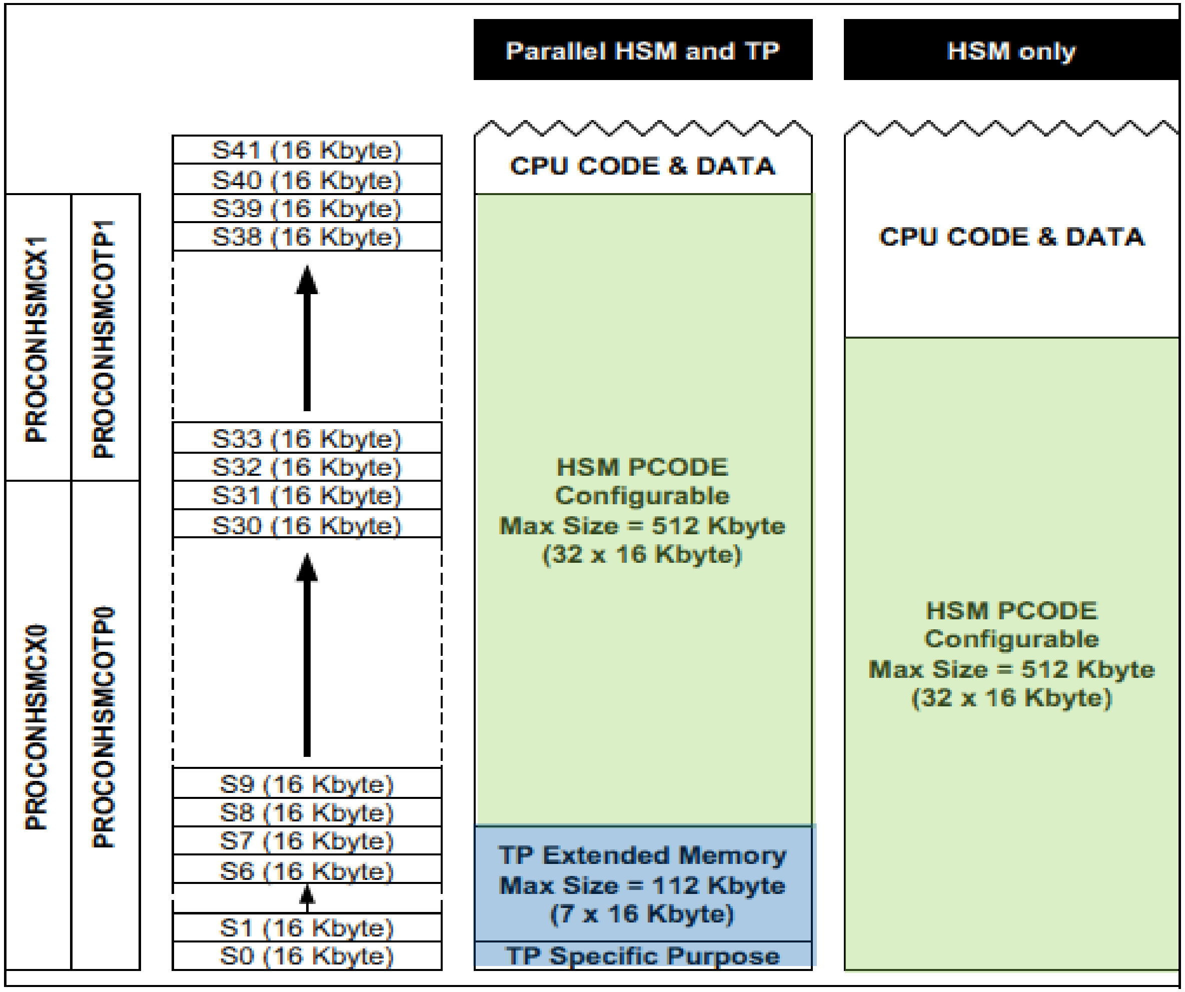
* O.K. to use for limited retention (tbd)

1: adder is due to a prog before erase which is done for complement sensing

2: adder is due to higher number of `1's to be programmed: for complement there are always the same amount of bits to be programmed (64bits w/o ECC whereas for single ended the average data would be 32 bits w/o ECC)

TC3xx PFLASH Address Map

PFLASH TP and HSM PCODE Configuration

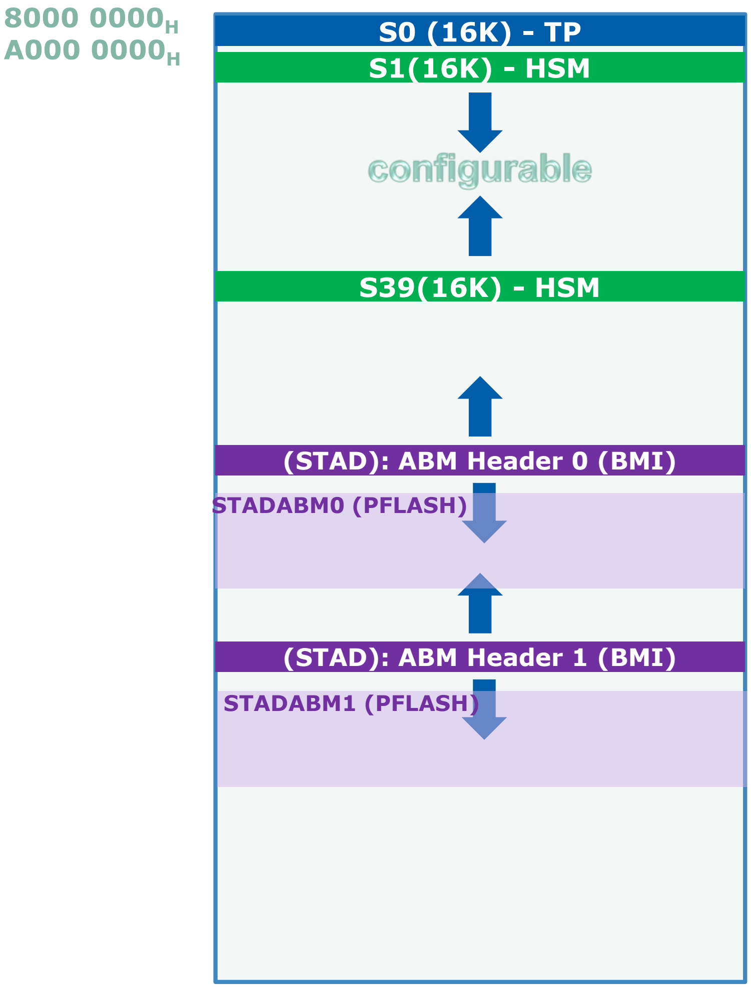


- **Configuration Options**
  - □ If parallel TP and HSM operation are required then PF0 S0 to S39 may be configured for TP and HSM PCODE as follows:
    - ¬ PF0 S0: specific TP purpose.
    - ¬ PF0 S1 to S6: TP extended memory
    - ¬ PF0 S7 to S39: HSM PCODE
    - ¬ TP, HSM PCODE and CPU address ranges should be contiguous.

TC3xx PFLASH Address Map - 2

ABM (Alternate Boot Mode)

› PFLASH (lower 1/2/3 MB)



ABM (Alternate Boot Mode)

- □ User start addresses shall be configurable in both ABM and internal start from Flash modes.
- □ In case of **A**lternate **B**oot **M**ode , Start address is configured in ABM.STADABMx like before.
- □ The ABM configuration need to be updated with PFLASH code update as the CRC need to be recalculated, therefore ABM header(s) are located in PFLASH (as legacy behavior like in AURIX generation)

PFLASH0

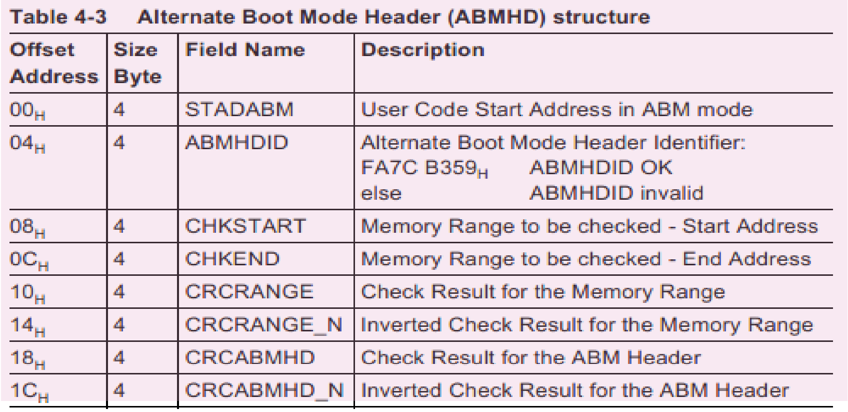


Table 4-3 Alternate Boot Mode Header (ABMHD) structure

Offset Address	Size Byte	Field Name	Description
00_H	4	STADABM	User Code Start Address in ABM mode
04_H	4	ABMHDID	Alternate Boot Mode Header Identifier: $FA7C\ B359_H$ ABMHDID OK else ABMHDID invalid
08_H	4	CHKSTART	Memory Range to be checked - Start Address
$0C_H$	4	CHKEND	Memory Range to be checked - End Address
10_H	4	CRCRANGE	Check Result for the Memory Range
14_H	4	CRCRANGE_N	Inverted Check Result for the Memory Range
18_H	4	CRCABMHD	Check Result for the ABM Header
$1C_H$	4	CRCABMHD_N	Inverted Check Result for the ABM Header

Program Flash Interface (PFI)

- Prefetch Path
  - Data Read Line Buffer (DRLB)
    - Used for SDTD and BTR2 access
  - Flash Prefetch Buffer (FPB)
    - Used for BTR4 access
- Demand Path
  - Accesses which cannot be serviced by a prefetch access are serviced by a demand access direct to the PFLASH bank.
  - Local CPU PMBI program code fetches
  - local CPU DMBI Non Safe load data constants
  - local CPU DMBI Safe load data constants
  - SRI accesses requested by remote bus masters

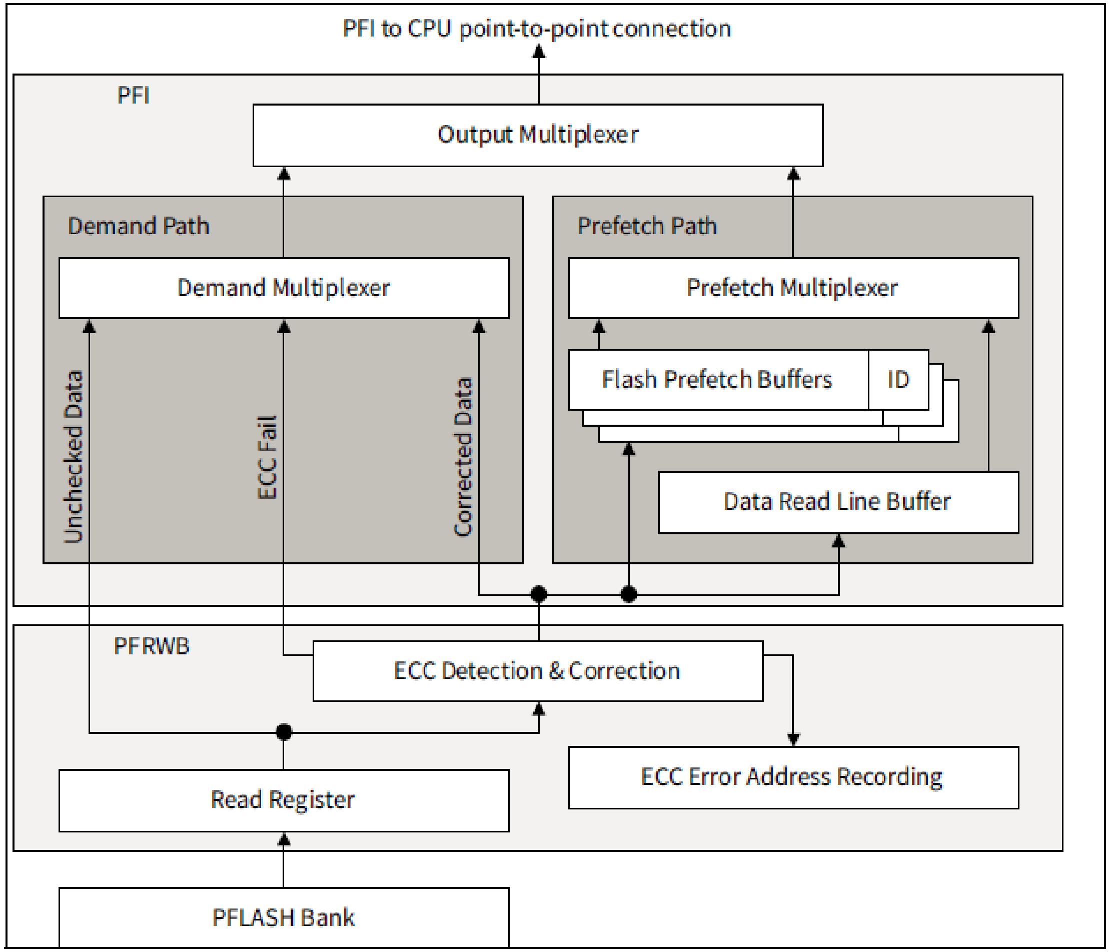
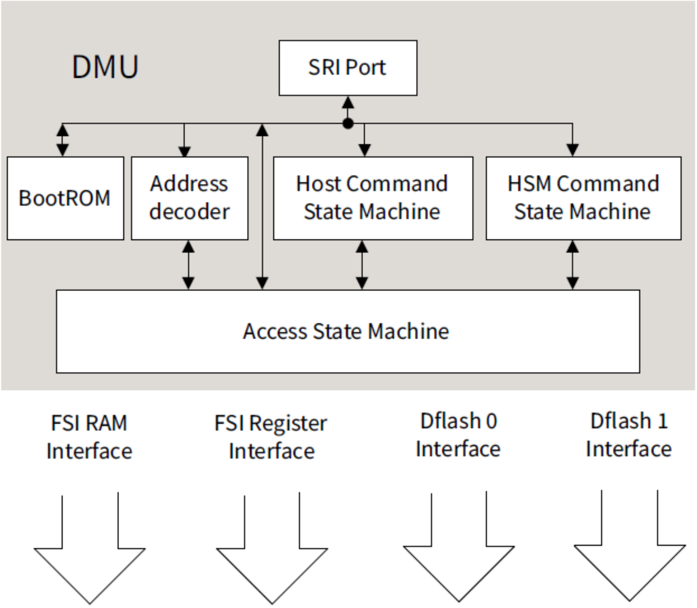
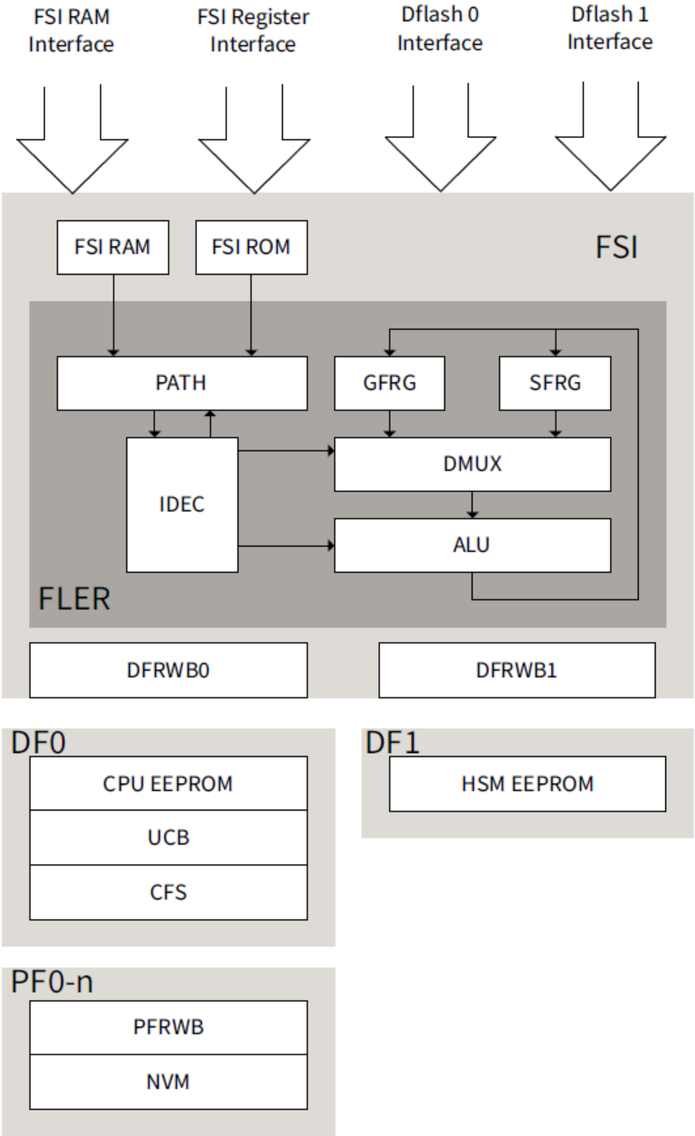


Figure 60 Block Diagram of the PFI module.

DMU





- DMU interfaces to the FSI and PFI for all flash operations, and PFlash read respectively.

DMU – FLASH Read Access Transaction Type

- **Program Flash**
  - The transaction types for local CPU (at Table 126) and remote CPU (at Table 127) read accesses to a PFLASH are as follows:

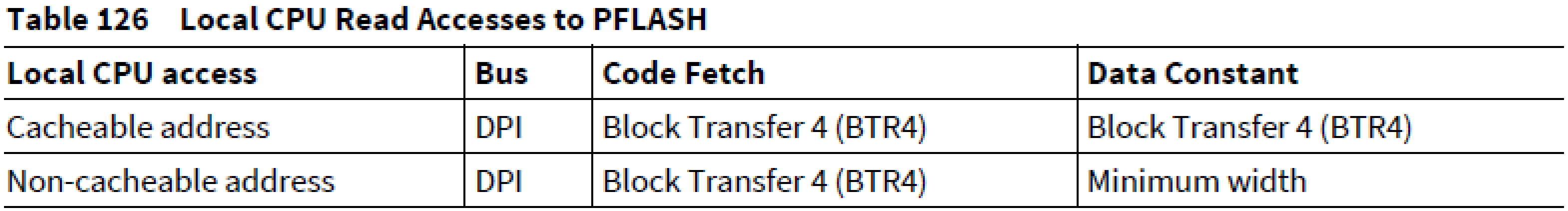


Table 126 Local CPU Read Accesses to PFLASH

Local CPU access	Bus	Code Fetch	Data Constant
Cacheable address	DPI	Block Transfer 4 (BTR4)	Block Transfer 4 (BTR4)
Non-cacheable address	DPI	Block Transfer 4 (BTR4)	Minimum width

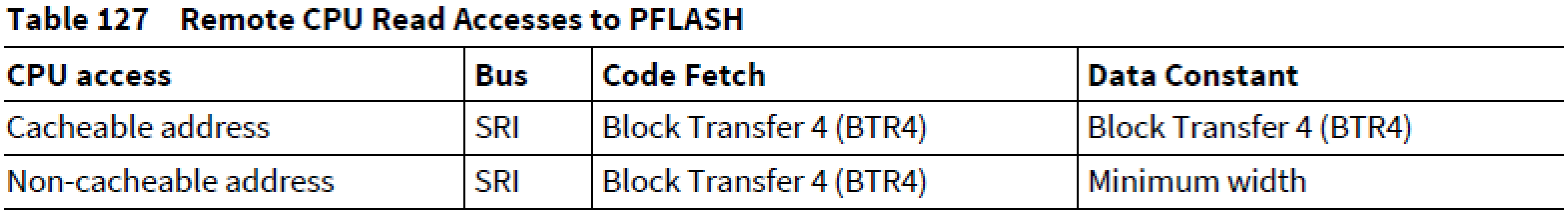


Table 127 Remote CPU Read Accesses to PFLASH

CPU access	Bus	Code Fetch	Data Constant
Cacheable address	SRI	Block Transfer 4 (BTR4)	Block Transfer 4 (BTR4)
Non-cacheable address	SRI	Block Transfer 4 (BTR4)	Minimum width

- **Data Flash**
  - Read accesses to DFLASH must be single transfers made across the SRI and are available only in the noncacheable address range.
  - A Block Transfer will result in a bus error.

Agenda

1 Architectural Overview

2 Memory Concept

3 On-chip Bus Connectivity

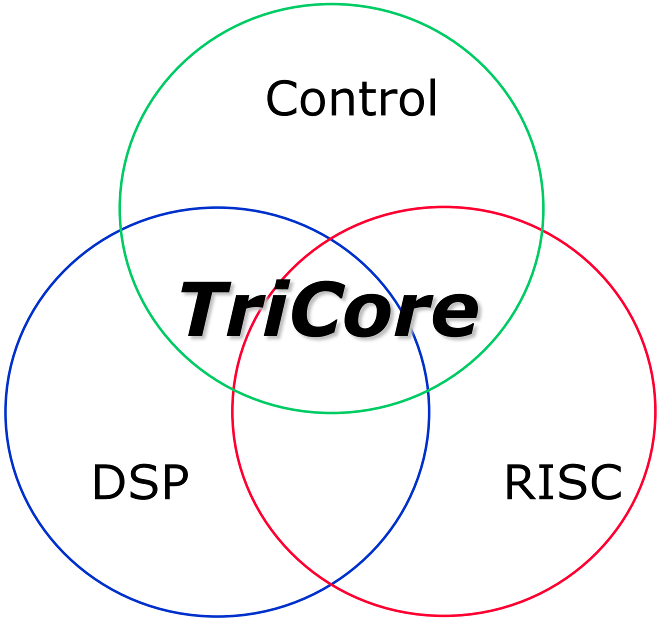
4 NVM System

5 TC1.6.2 Architecture

6 Summary

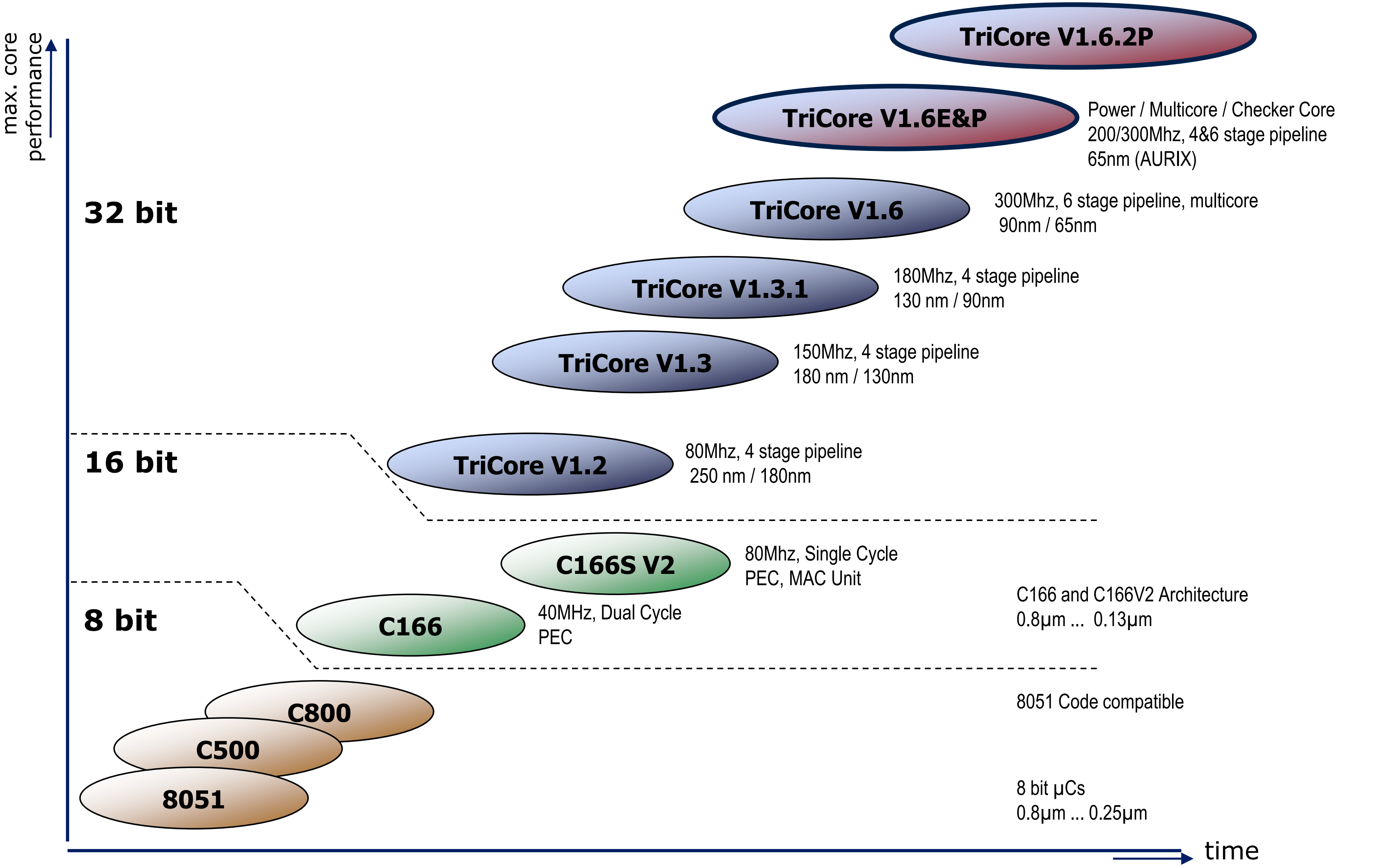
The Best of MCU, RISC, & DSP

- The TriCore processor architecture combines three powerful technologies within one processing unit.



  - Reduced Instruction Set Computing (RISC) processor architecture
  - Digital Signal Processing (DSP) operations and addressing modes
  - Real-time capability of a microcontroller with on-chip memories and peripherals

Infineon Technologies Microcontroller Core Roadmap



Power / Multicore / Checker Core
200/300Mhz, 4&6 stage pipeline
65nm (AURIX)

300Mhz, 6 stage pipeline, multicore
90nm / 65nm

180Mhz, 4 stage pipeline
130 nm / 90nm

150Mhz, 4 stage pipeline
180 nm / 130nm

80Mhz, 4 stage pipeline
250 nm / 180nm

80Mhz, Single Cycle
PEC, MAC Unit

40MHz, Dual Cycle
PEC

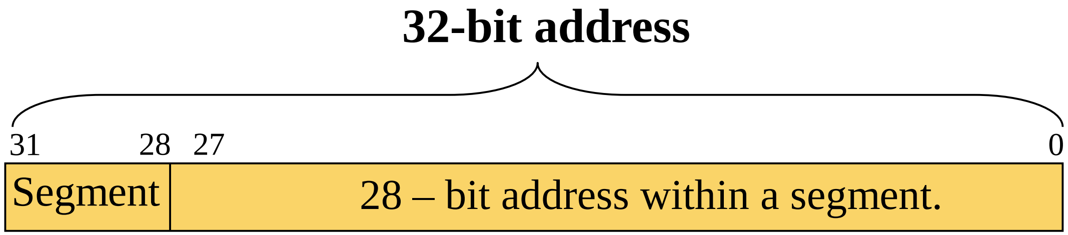
C166 and C166V2 Architecture
0.8µm ... 0.13µm

8051 Code compatible

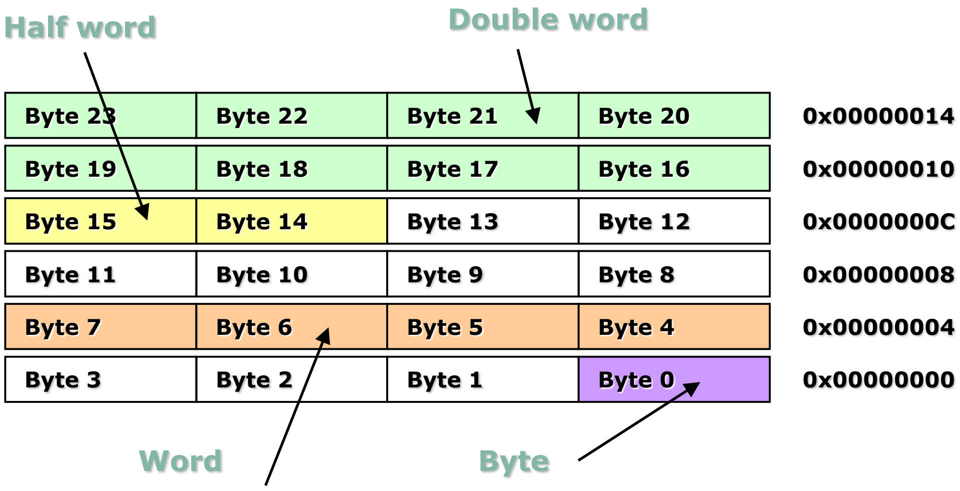
8 bit µCs
0.8µm ... 0.25µm

TC1.6.2P Architecture
Address Range

- **4Gbyte** address range (2^{32}) and divided into **16** segments of **256MB**



- **Little-endian** byte ordering for data, memory and CPU registers

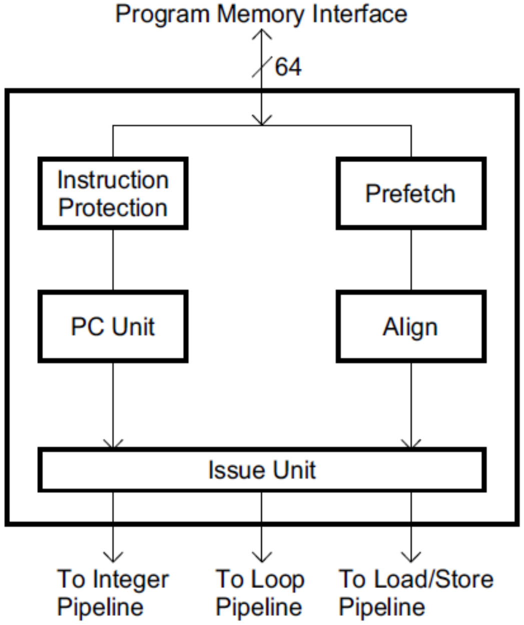


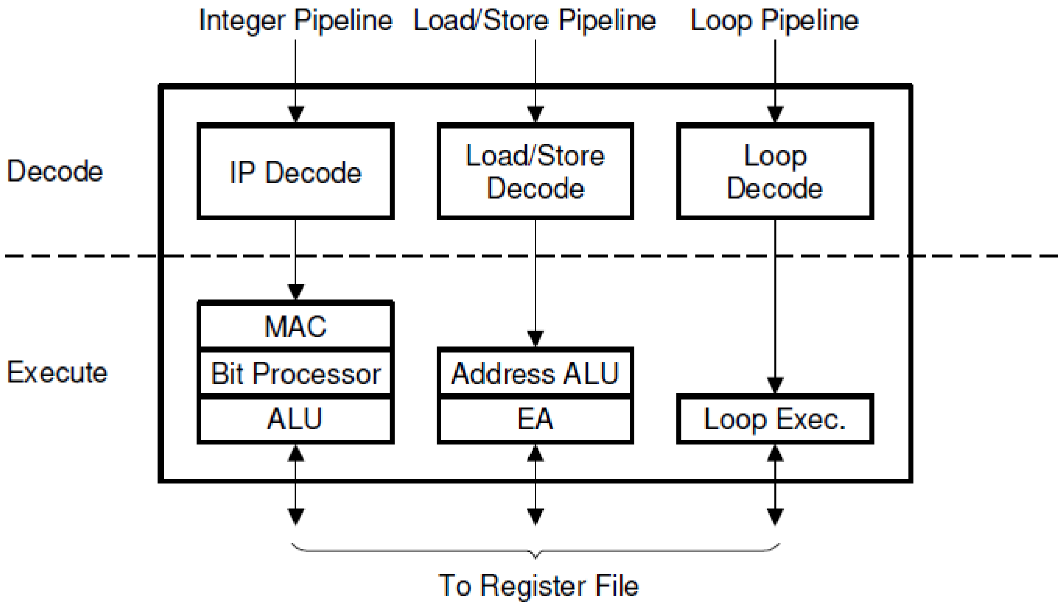
Byte 23	Byte 22	Byte 21	Byte 20	0x00000014
Byte 19	Byte 18	Byte 17	Byte 16	0x00000010
Byte 15	Byte 14	Byte 13	Byte 12	0x0000000C
Byte 11	Byte 10	Byte 9	Byte 8	0x00000008
Byte 7	Byte 6	Byte 5	Byte 4	0x00000004
Byte 3	Byte 2	Byte 1	Byte 0	0x00000000

little endian

TC1.6.2P Architecture Pipeline

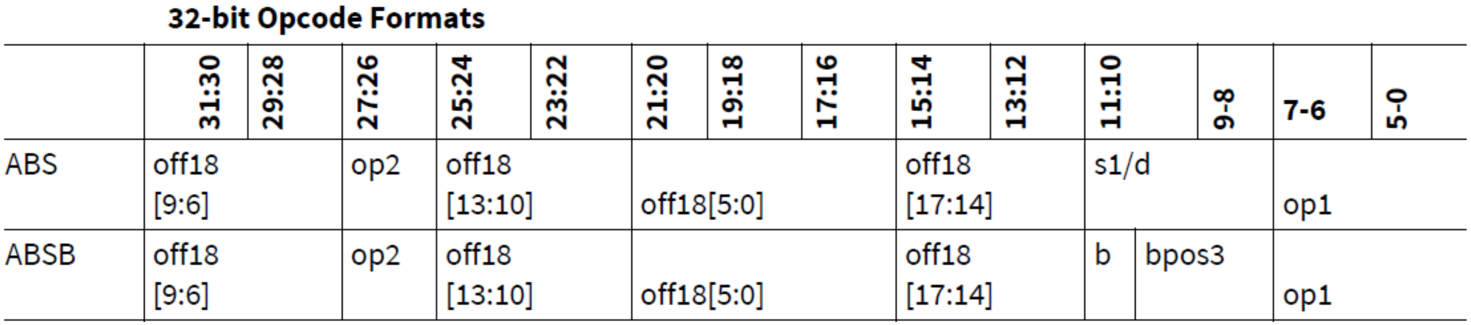
- Dual instruction issuing (in parallel into **Integer Pipeline** and **Load/Store Pipeline**)
- Third **pipeline** for **loop** instruction only (zero overhead loop)





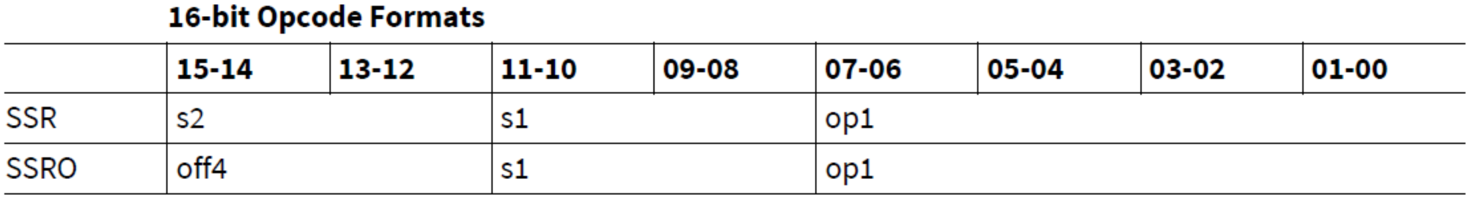
TC1.6.2P Architecture Instructions

- Support **16-bit** and **32-bit** instructions formats for reduced code size



32-bit Opcode Formats

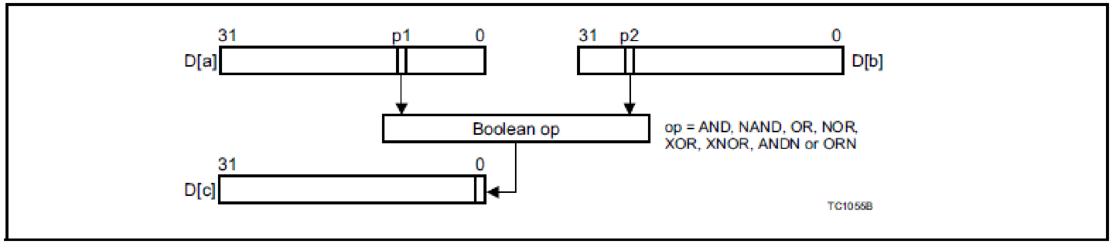
	31:30	29:28	27:26	25:24	23:22	21:20	19:18	17:16	15:14	13:12	11:10	9-8	7-6	5-0
ABS	off18 [9:6]		op2	off18 [13:10]		off18[5:0]			off18 [17:14]		s1/d		op1	
ABSB	off18 [9:6]		op2	off18 [13:10]		off18[5:0]			off18 [17:14]		b	bpos3	op1	



16-bit Opcode Formats

	15-14	13-12	11-10	09-08	07-06	05-04	03-02	01-00
SSR	s2		s1		op1			
SSRO	off4		s1		op1			

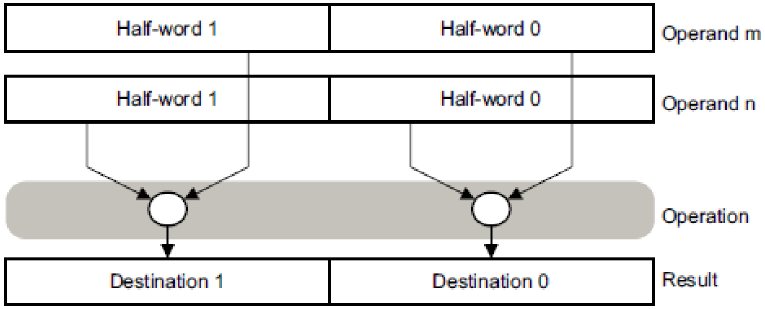
- Extensive **Bit manipulation** instructions

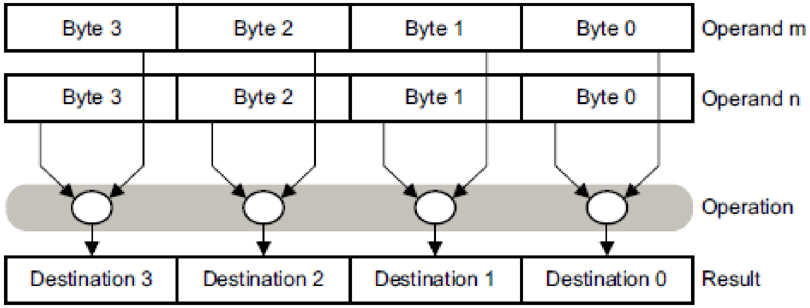


Boolean Operations

TC1.6.2P Architecture Instructions

- Multiply and Accumulate (**MAC**) instructions: Dual 16 × 16, 16 × 32, 32 × 32
- Dedicated **Integer Divide** unit
- Single Instruction Multiple Data (**SIMD**) **packed data** operations



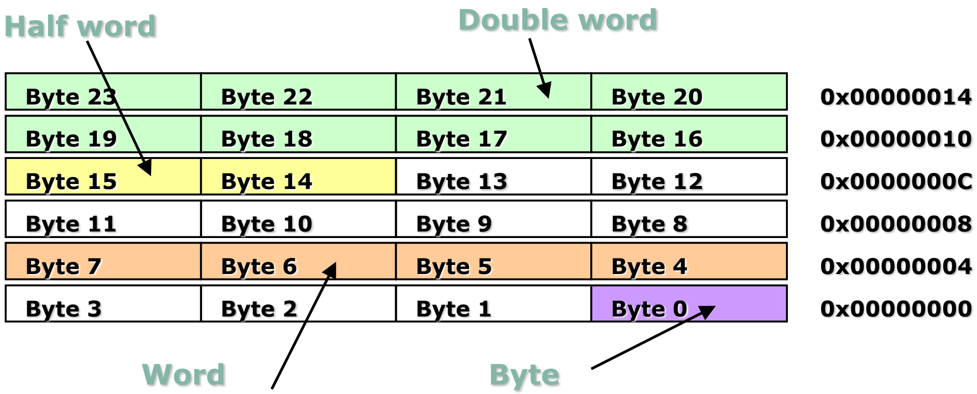


- Most instructions executed in **one cycle**
- Branch instructions in 1, 2 or 3 cycles (using dynamic branch prediction)

TC1.6.2P Architecture
Data Types

- Data types
  - Boolean,
  - Integer with saturation,
  - Bit array,
  - Signed fraction,
  - Character,
  - Double-word integers,
  - Signed integer, unsigned integer,
  - IEEE-754 single-precision floating point

- Data formats
  - Bit (1-bit),
  - Byte (8-bits),
  - Half-word (16-bits),
  - Word (32-bits),
  - Double-word (64-bits)
- Data storage

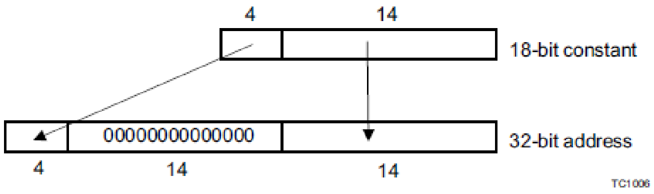


Byte 23	Byte 22	Byte 21	Byte 20	0x00000014
Byte 19	Byte 18	Byte 17	Byte 16	0x00000010
Byte 15	Byte 14	Byte 13	Byte 12	0x0000000C
Byte 11	Byte 10	Byte 9	Byte 8	0x00000008
Byte 7	Byte 6	Byte 5	Byte 4	0x00000004
Byte 3	Byte 2	Byte 1	Byte 0	0x00000000

little endian

TC1.6.2P Architecture Addressing Modes

- TriCore supports
  - Absolute addressing (data and code)
    - Memory Address = 73
  - Base + offset addressing
    - Memory Address = A3 + 6
  - Pre-increment/decrement addressing
    - A3 = A3 +6 ; Memory Address = A3
  - Post-increment/decrement addressing
    - Memory Address = A3; A3 = A3 +6
  - Circular Addressing
    - DSP Buffers, Complex
  - Bit-reverse addressing
    - FFT algorithm, Very Complex



TC1.6.2P Architecture
Core Registers (GPRs and CSFRs)

General Purpose Registers	
D0 – D15	Data Registers.
A0 – A15	Address Registers.

System Registers	
PC	Program Counter
PSW	Program Status Word
SYSCON	System Control Registers
PCXI	Previous Context Information.

Context Management	
FCX	Free CSA List Head Pointer
LCX	Free CSA List Limit Pointer

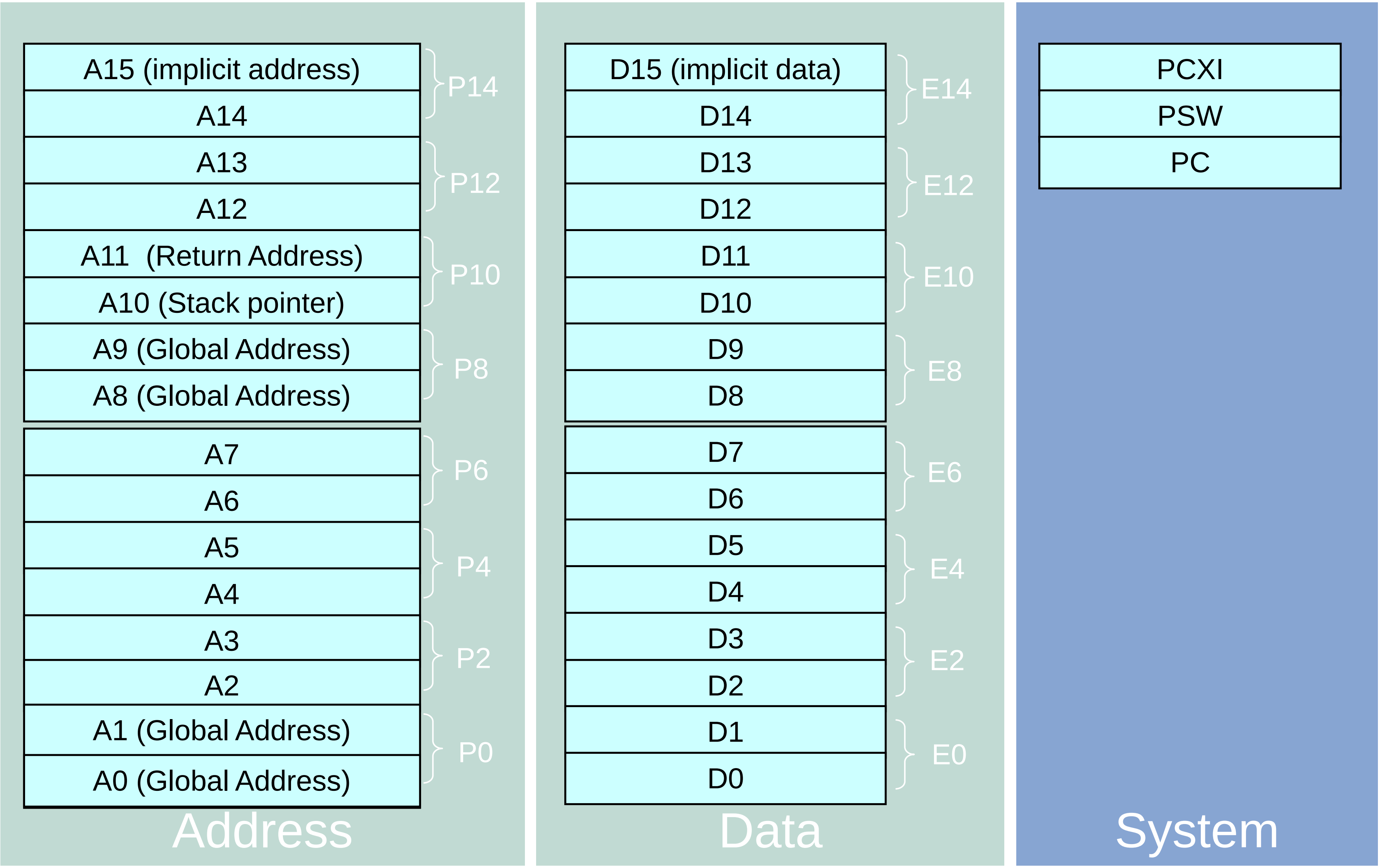
CPU Interrupt and Trap Control	
ICR	Interrupt Control Reg.
BIV	Base Address of Interrupt Vect. Table.
BTV	Base Address of Trap Vect. Table.

Memory Protection	
DPRx_0..3	Data Segment Protection Regs.
DPMx	Data Protection Mode Reg. x
CPRx_0..1	Code Segment Protection Regs.
CPMx	Code Protection Mode Reg. x

Stack management	
ISP	Interrupt Stack Pointer

Core Special Function Registers (CSFRs) can only be accessed by special read/write instructions:
→ MTCR: move to core special function register (write access, supervisor mode only)
→ MFCR: move from core special function register (read access)

TC1.6.2P Architecture
Architecture Registers



Address

A15 (implicit address)	P14
A14	
A13	P12
A12	
A11 (Return Address)	P10
A10 (Stack pointer)	
A9 (Global Address)	P8
A8 (Global Address)	
A7	P6
A6	
A5	P4
A4	
A3	P2
A2	
A1 (Global Address)	P0
A0 (Global Address)	

Data

D15 (implicit data)	E14
D14	
D13	E12
D12	
D11	E10
D10	
D9	E8
D8	
D7	E6
D6	
D5	E4
D4	
D3	E2
D2	
D1	E0
D0	

System

PCXI
PSW
PC

TC1.6.2P Architecture Context Management

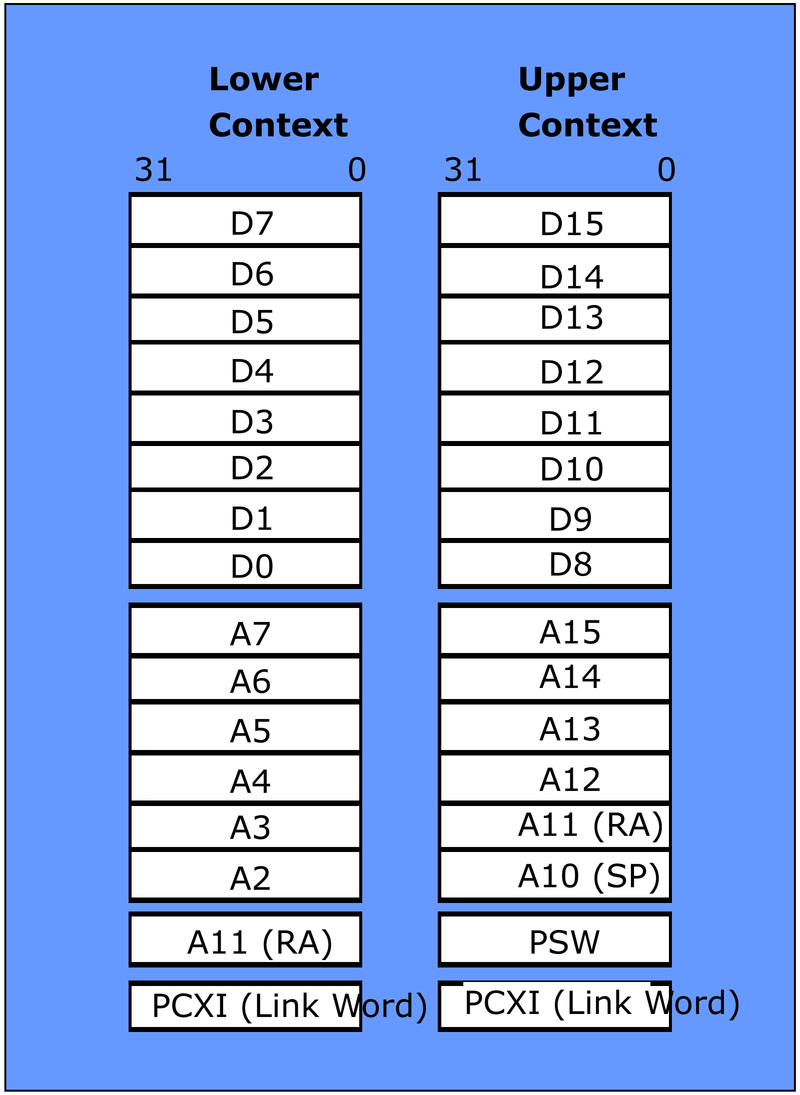
- Switching between different SW tasks needs to store the context before execute the task and restore the context after finishing the tasks.

- Context is everything the µC needs to know in order to start (or re-start) a task.
  - The upper context (task specific), stored automatically.
  - The lower context (for parameter passing), stored with instruction.

- Fast Context Store and Restore
  - **Hardware managed** context switching
  - **Wide memory interface** for fast context switch
  - **Automatic** context **save-on-entry** and **restore-on-exit** for subroutine, interrupt, trap

TC1.6.2P Architecture
Context Save Areas (CSA)

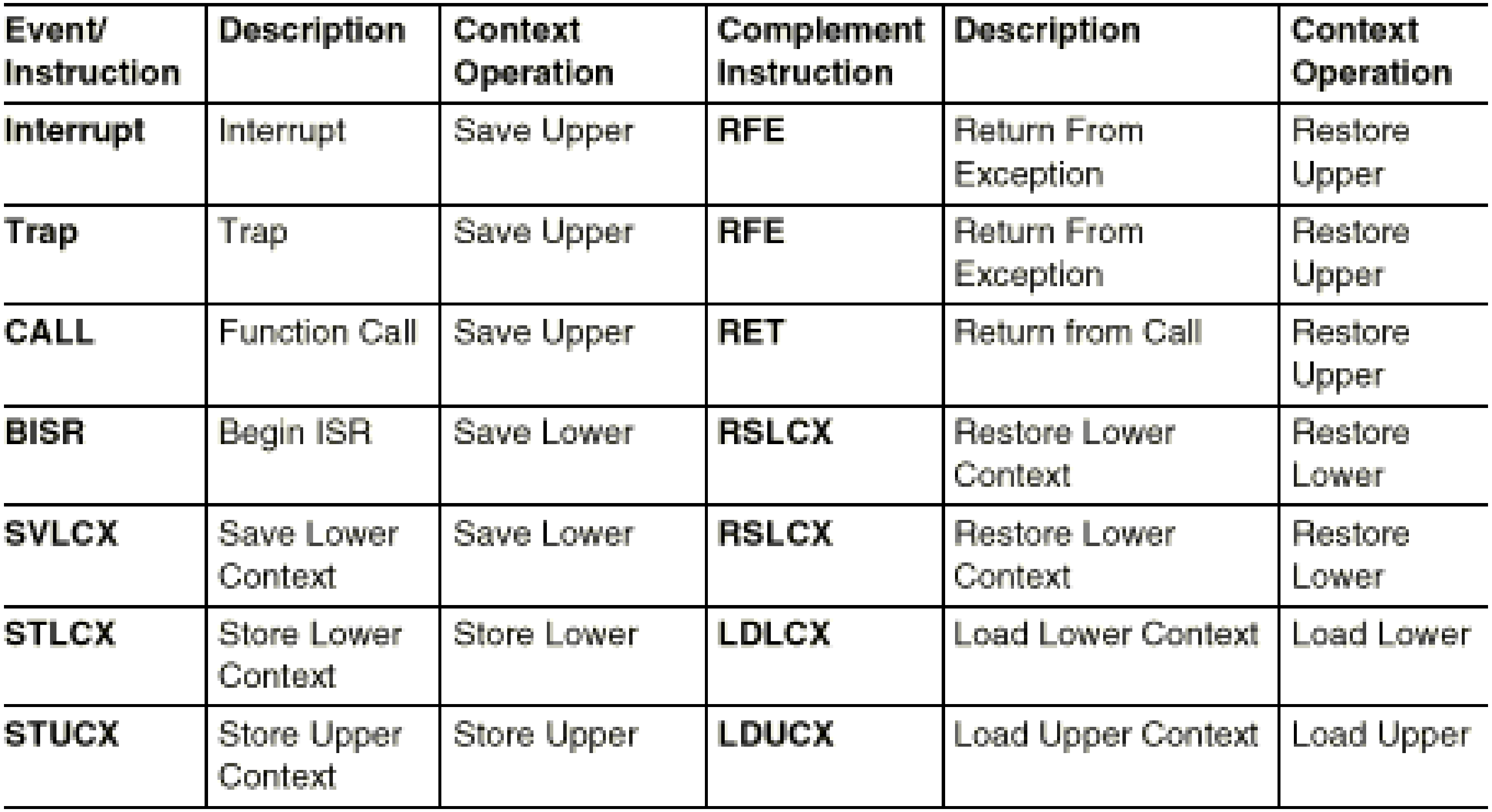
- Contexts are saved in Context Save Area (CSAs):
  - 16 words of memory storage, that can hold exactly 1 upper or 1 lower context
  - 16-word aligned
  - CSAs form a distributed stack, implemented as a linked list
  - Unused CSAs are linked together on a free context list
  - CSAs of tasks currently executing are linked together on a previous context list
  - Hardware handles allocation & freeing of CSAs as needed



Lower Context	**Upper Context**
31 0	31 0
D7	D15
D6	D14
D5	D13
D4	D12
D3	D11
D2	D10
D1	D9
D0	D8
A7	A15
A6	A14
A5	A13
A4	A12
A3	A11 (RA)
A2	A10 (SP)
A11 (RA)	PSW
PCXI (Link Word)	PCXI (Link Word)

TC1.6.2P Architecture
Context-Related Events and Instructions

TriCore architecture saves or restores context when one of the events / instructions below occurs



Event/ Instruction	Description	Context Operation	Complement Instruction	Description	Context Operation
Interrupt	Interrupt	Save Upper	**RFE**	Return From Exception	Restore Upper
Trap	Trap	Save Upper	**RFE**	Return From Exception	Restore Upper
CALL	Function Call	Save Upper	**RET**	Return from Call	Restore Upper
BISR	Begin ISR	Save Lower	**RSLCX**	Restore Lower Context	Restore Lower
SVLCX	Save Lower Context	Save Lower	**RSLCX**	Restore Lower Context	Restore Lower
STLCX	Store Lower Context	Store Lower	**LDLCX**	Load Lower Context	Load Lower
STUCX	Store Upper Context	Store Upper	**LDUCX**	Load Upper Context	Load Upper

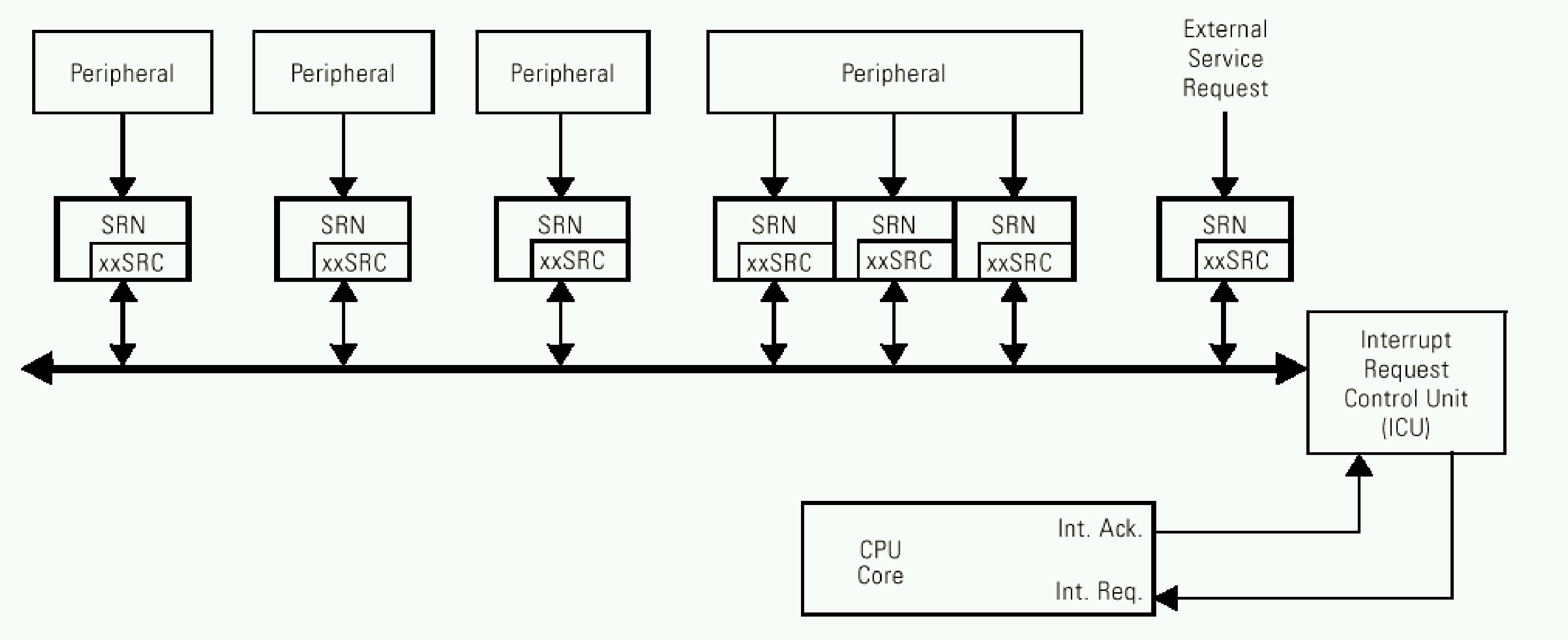
TC1.6.2P Architecture
Context Save Area (CSA) Placement

- The Context Save Areas (CSA) placed by user SW.
  - CSA placement in DSPR (maximum performance)
  - CSA placement in DLMU
  - CSA placement in Cached External Memory
- CSA is generally initialized at startup.

- 3 point core registers
  - FCX - Free CSA List Head Pointer
  - PCX - Previous Context Pointer (in register PCXI)
  - LCX - Free CSA List Limit Pointer

TC1.6.2P Architecture
Interrupt System Overview

- Independent Interrupt Systems per Interrupt Service Provider (CPUx, DMA)
- Centralized Arbitration System
- 255 Prioritization Levels per Interrupt Service Provider

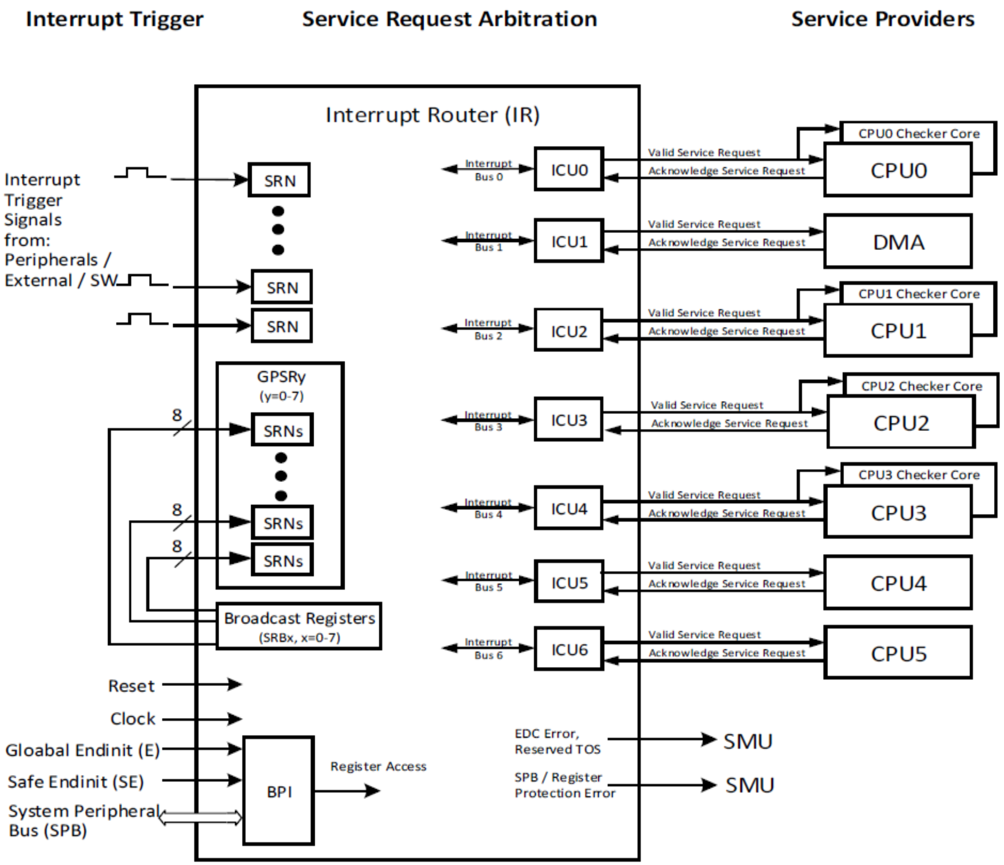


TC1.6.2P Architecture
Interrupt System Overview

- **Service Requests :** Interrupt Requests.
- **SRN** : Service Request Node
  - Each interrupt request source must connect to a SRN.
  - An interrupting device can have more than one SRN.
  - Each peripheral interrupt with a dedicated SRN.
  - Each SRN with a programmable 8-bit priority vector
- **ICU** : Interrupt Control Unit
  - Handles the priority arbitration and the communication with the CPU / DMA module (Service Provider)
- **SRC** : Service Request Control Register
  - Each SRN contains a SRC and the necessary logic to communicate with the requesting source and the interrupt arbitration bus

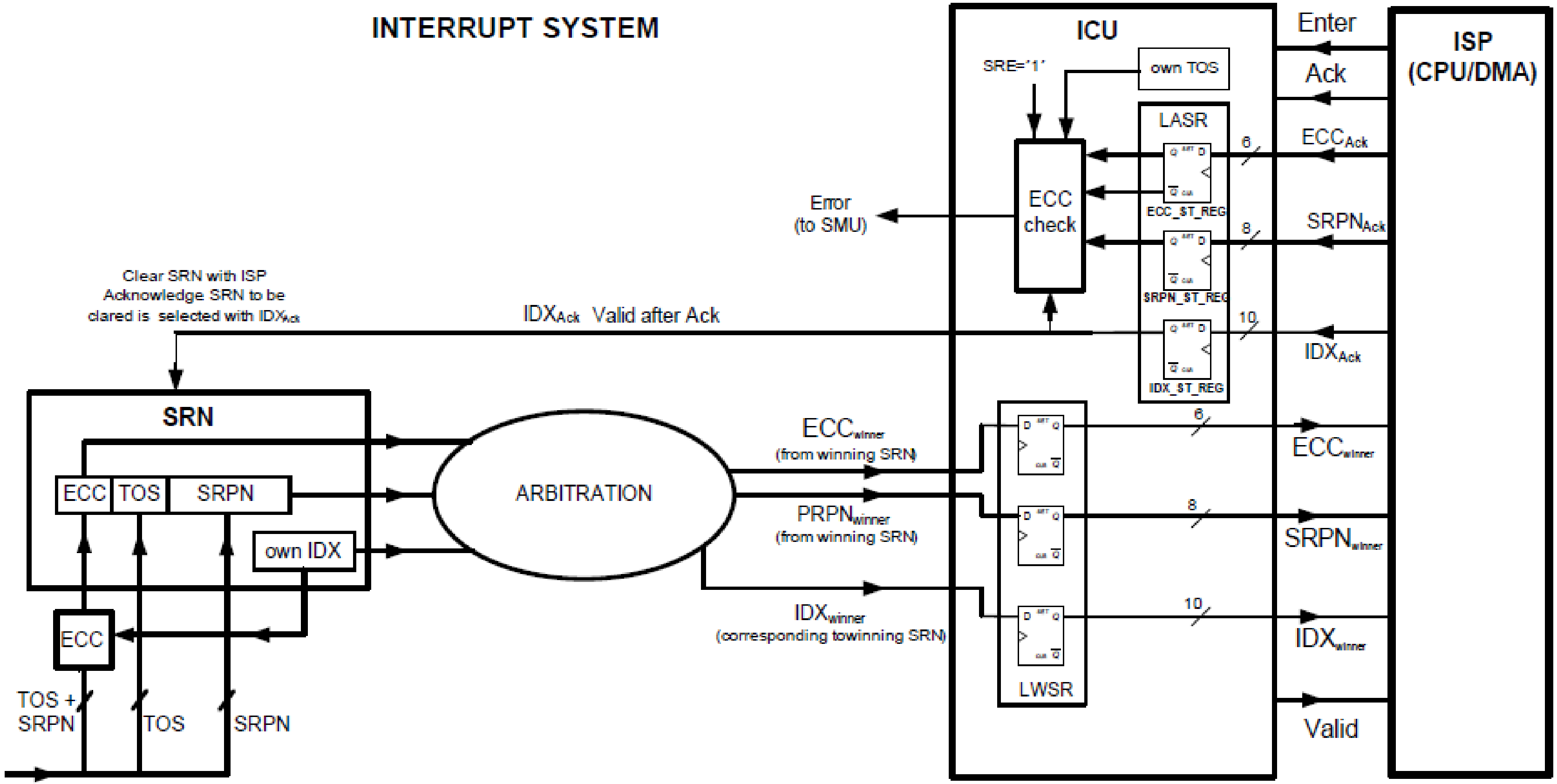
TC1.6.2P Architecture
Interrupt Router (IR) Features

- Interrupt System with support of up to **1024** service **requests**
- Support of up to **255** service request **priority** levels per ICU / Service Provider
- Support of up to **8** ICUs / Service Providers
- **Low latency arbitration** - three / four clocks from receipt of an service request to sending it to the service provider
- **8** General Purpose Service Requests (**GPSR**) per CPU that can be used as **Software Interrupts**
- Service Request Broadcast Registers (**SRB**) to signal General Purpose Service Requests (Software Interrupts) simultaneously to multiple Service Providers
- **Eight SRNs** (Int_SCUSRC[7:0]) are reserved to handle external interrupts. Controlled in the External Request Unit (ERU) (SCU chapter).



Abbreviations:
SRN: Service Request Node
ICU: Interrupt Control Unit
TOS: Type of Service
BPI: Bus Peripheral Interface
GPSR: General Purpose Service Request Group (8 x SRN / Group)

TC1.6.2P Architecture Arbitration Process



Abbreviations:

Enter – Valid winner taken by ISP
Ack – Acknowledge taken winner of by SP
ECC – Error Correction Code
IDX – SRN Index Number
ISP – Interrupt Service Provider
Own IDX – SRNs unique Index Number
Own TOS – ICUs unique TOS number
SPB – System Peripheral Bus

SRN – Service Request Node
SRN(IDX$_{ACK}$) – SRN selected by acknowledged index number
SRPN – Service Request Priority Number
TOS – Type of Service
Valid -- ICU signals valid winner to ISP
LWSR – Last Winning Service Request register
LASR – Last Acknowledged Service Request register

int_ecc

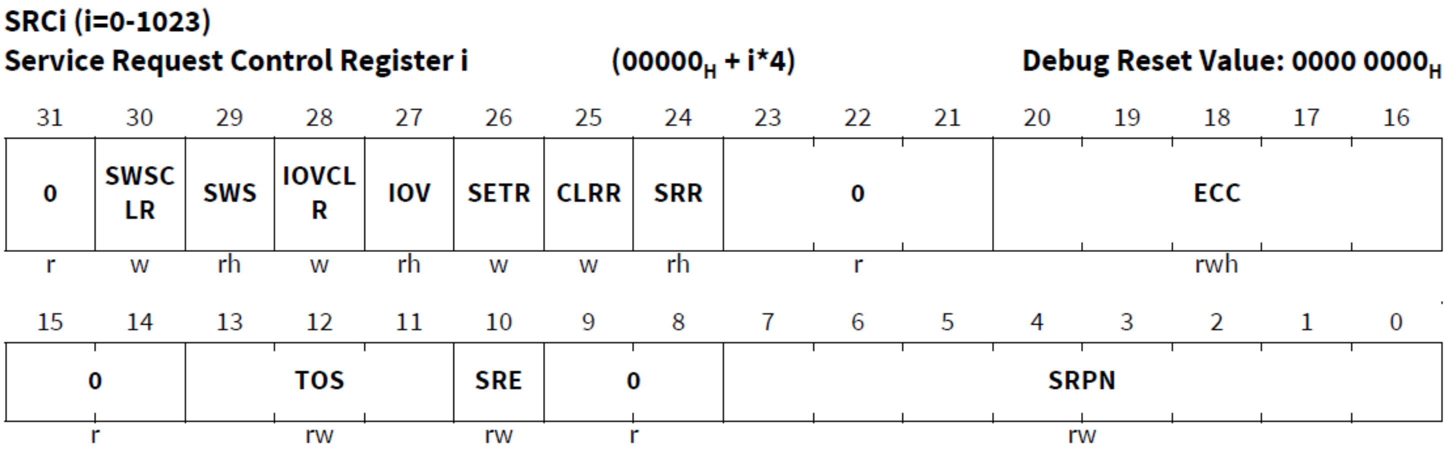
TC1.6.2P Architecture
Interrupt Protection Scheme

- ECC Protection of SRN’s SRC
  - SRC ECC check will be checked whenever the SRN with an pending service request was accepted by the selected (TOS) service provider as next service request to be processed. ECC is only used for error detection. Detected errors are reported to the SMU.

- Access protection of SRCx[31:0]
  - Introduced in A2G via an On Chip Bus Master TAG-ID protection. Violation will send alarm to SMU.
  - SRC[31:16] is write protected by ACCEN_TOSx (x = SRC.TOS)
  - SRC[15:0] is write protected by ACCEN_CONFIG

- Protection of access to the reserved TOS
  - On detection of a service request signaled to a Reserved TOS, an alarm is signaled to the SMU.

TC1.6.2P Architecture
Service Request Control Register (SRC)



SRCi (i=0-1023)

Service Request Control Register i **(00000$_H$ + i*4)** **Debug Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**SWSCLR**	**SWS**	**IOVCLR**	**IOV**	**SETR**	**CLRR**	**SRR**	**0**			**ECC**				
r	w	rh	w	rh	w	w	rh	r			rwh				

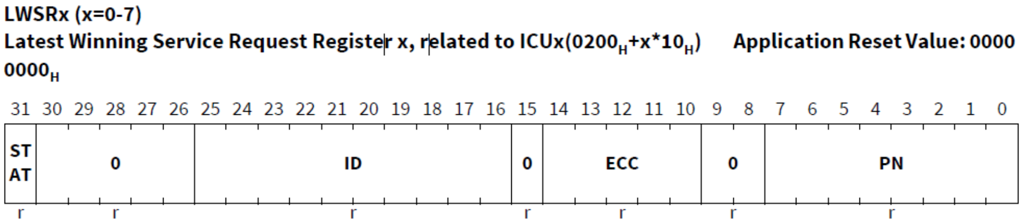
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**TOS**			**SRE**	**0**		**SRPN**							
r		rw			rw	r		rw							

- **SRPN** : Service Request Priority Number
  - Priority increasing with number
  - **CPU**: SRPN=0 not allowed
  - **DMA**: available **channel numbers**
- **SRE** : Service Request Enable / Disable
- **TOS** : Type Of Service Control
  - 0->CPU0, **1->DMA**, 2->CPU1
  - 3->CPU2, 4->CPU3, 5->CPU4, 6->CPU5

- **ECC** : Error Correction Code
  - Updated with any write to SRC[31:0]
- **SETR/CLRR** : Set/Clear Interrupt request by software
- **SRR** : Service Request Pending
- **IOV/IOVCLR** : Interrupt Overflow Bit /Software clear bit
- **SWS/SWSCLR** : Software Sticky Bit /software clear bit

TC1.6.2P Architecture
ICU Control Registers

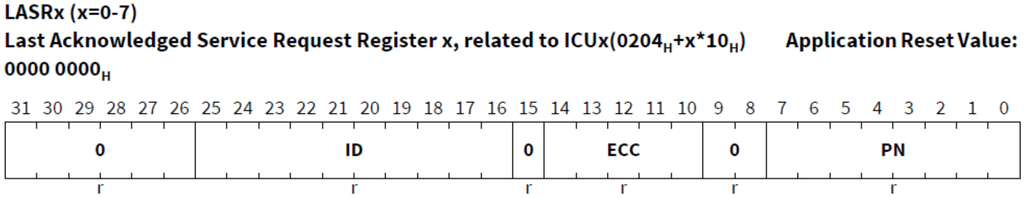
- Last Winning Service Request (**LWSR**)



LWSRx (x=0-7)
Latest Winning Service Request Register x, related to ICUx(0200_H+x*10_H) Application Reset Value: 0000 0000_H

31	30 29 28 27 26	25 24 23 22 21 20 19 18 17 16	15	14 13 12 11 10	9 8	7 6 5 4 3 2 1 0
STAT	0	ID	0	ECC	0	PN
r	r	r	r	r	r	r

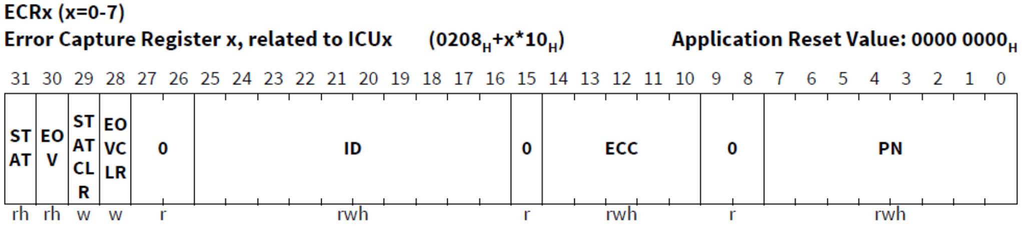
- Last Acknowledged Service Request (**LASR**)



LASRx (x=0-7)
Last Acknowledged Service Request Register x, related to ICUx(0204_H+x*10_H) Application Reset Value: 0000 0000_H

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16	15	14 13 12 11 10	9 8	7 6 5 4 3 2 1 0
0	ID	0	ECC	0	PN
r	r	r	r	r	r

- Error Captures Register (**ECR**)
  ECC error was detected for an Acknowledged Service Request

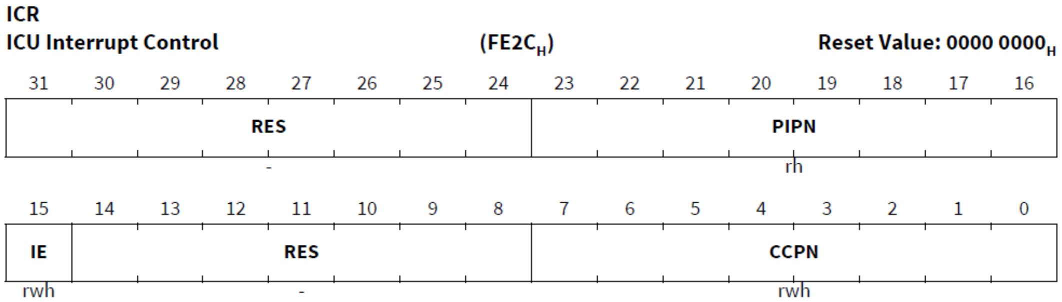


ECRx (x=0-7)
Error Capture Register x, related to ICUx (0208_H+x*10_H) Application Reset Value: 0000 0000_H

31	30	29	28	27 26	25 24 23 22 21 20 19 18 17 16	15	14 13 12 11 10	9 8	7 6 5 4 3 2 1 0
STAT	EOV	STATCLR	EOVCLR	0	ID	0	ECC	0	PN
rh	rh	w	w	r	rwh	r	rwh	r	rwh

TC1.6.2P Architecture
Two CSFRs support Interrupt handling

- **The ICU Interrupt Control Register (ICR)**
  - the Current CPU Priority Number (**CCPN**),
  - the global Interrupt enable/disable bit (**IE**),
  - the current Pending Interrupt Priority Number (**PIPN**).

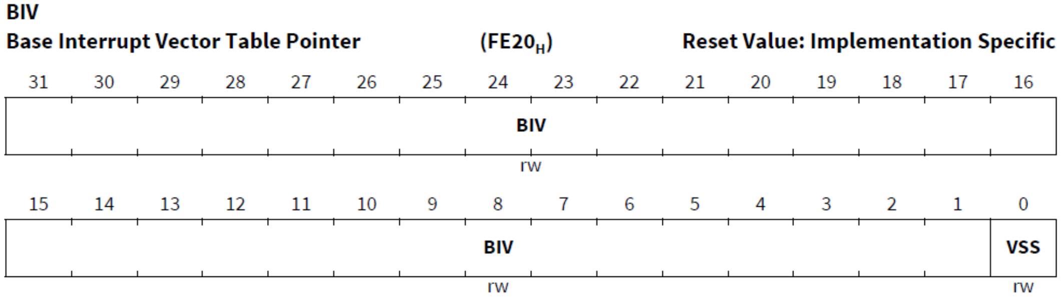


ICR
ICU Interrupt Control **(FE2C$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES								**PIPN**							
-								rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IE	**RES**							**CCPN**							
rwh	-							rwh							

- **Base of Interrupt Vector Table register (BIV)**
  - Base Address of Interrupt Vector Table **(BIV)**
  - Vector Spacing Select **(VSS)**



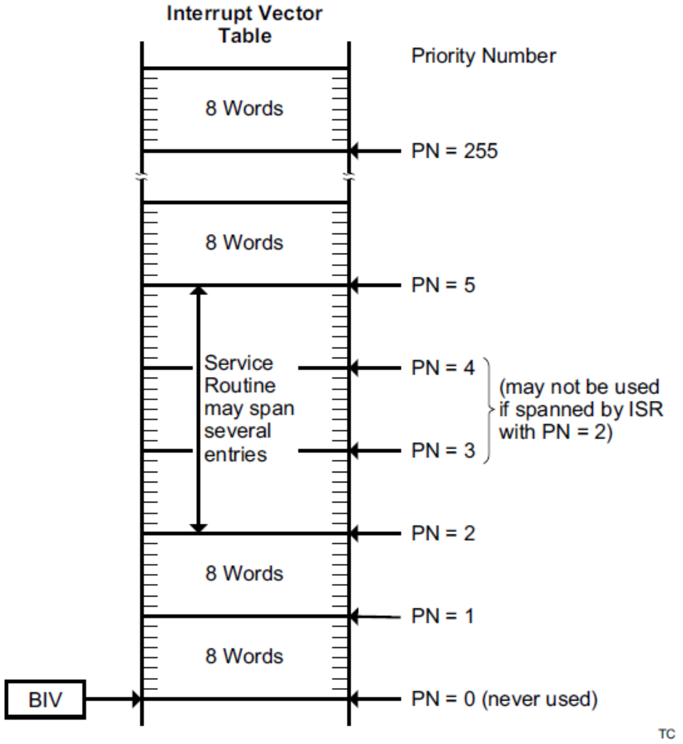
BIV
Base Interrupt Vector Table Pointer **(FE20$_H$)** **Reset Value: Implementation Specific**

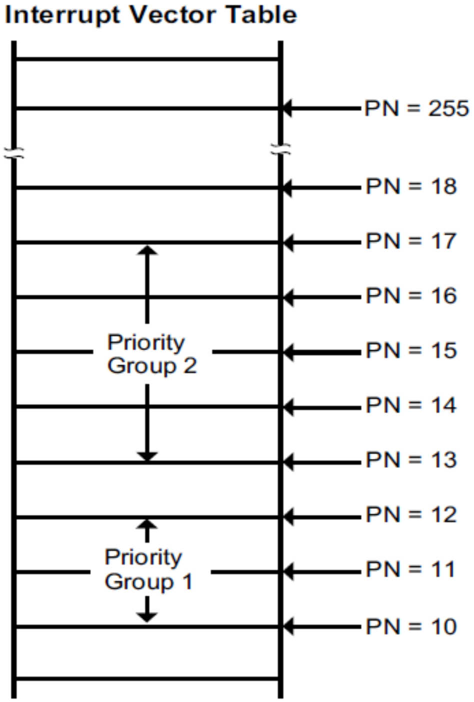
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BIV															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIV															**VSS**
rw															rw

TC1.6.2P Architecture
CPU Interrupt Vector Table

- BIV register can be modified using the MTCR instruction during the initialization phase of the system. (BIV is ENDINIT protected)
- Vector table are available with either **32 byte** to **8 byte** spacing between vectors. (selected by the BIV.VSS bit)
- Easy to span Interrupt Service Routines (ISRs) across vector entry locations.
- Interrupt priority groups for those cannot interrupt each other’s service routine.





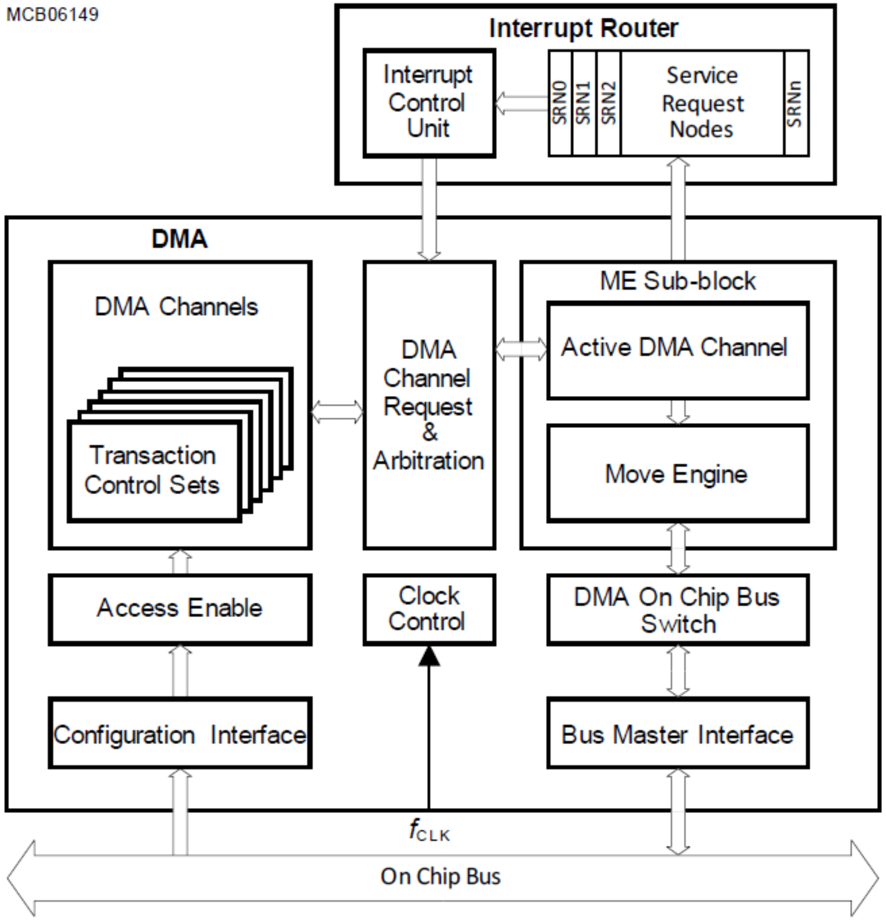
TC1.6.2P Architecture
Interrupt Entry and Exit

- **Entering an Interrupt Service Routine (ISR)**
  - The upper context of the current task is saved.
  - The Return Address A[11] is updated with the current PC.
  - The interrupt system is globally disabled: ICR.IE = 0.
  - A[10] Stack Pointer is set to the interrupt stack pointer (ISP) if PSW.IS = 0.
  - The I/O mode is set to Supervisor mode: PSW.IO = 10B.
  - The current Protection Register Set is set to 0: PSW.PRS = 000B.
  - The call depth limit selector is set for 64: PSW.CDC = 0000000B.
  - Call Depth Counter is enabled, PSW.CDE = 1.
  - The Current CPU Priority Number (ICR.CCPN) is saved into the Previous CPU Priority Number (PCXI.PCPN) field.
  - The Pending Interrupt Priority Number (ICR.PIPN) is saved into the Current CPU Priority Number (ICR.CCPN) field.
- **Exiting an Interrupt Service Routine (ISR)**
  - PCXI.PCPN is written to ICR.CCPN to set the CPU priority number to the value before interruption.
  - PCXI.PIE is written to ICR.IE to restore the state of this bit.

TC1.6.2P Architecture
DMA Service Requests

- **DMA channels** are associated with the Service Request Priority Number (**SRPN**) bit field programmed in the Service Request Control Register (SRC). For example:
  - DMA channel 000 equates to SRC.SRPN = 0_D programmed in IR.
  - DMA channel 001 equates to SRC.SRPN = 1_D programmed in IR.
  - DMA channel 002 equates to SRC.SPRN = 2_D programmed in IR.
  - DMA channel 003 equates to SRC.SRPN = 3_D programmed in IR.
  - DMA channel 004 equates to SRC.SPRN = 4_D programmed in IR.

- The **highest number** DMA **channel** with a pending **DMA Request wins** the DMA channel arbitration. The pending DMA request is forwarded to the highest number available ME.



TC1.6.2P Architecture
Traps

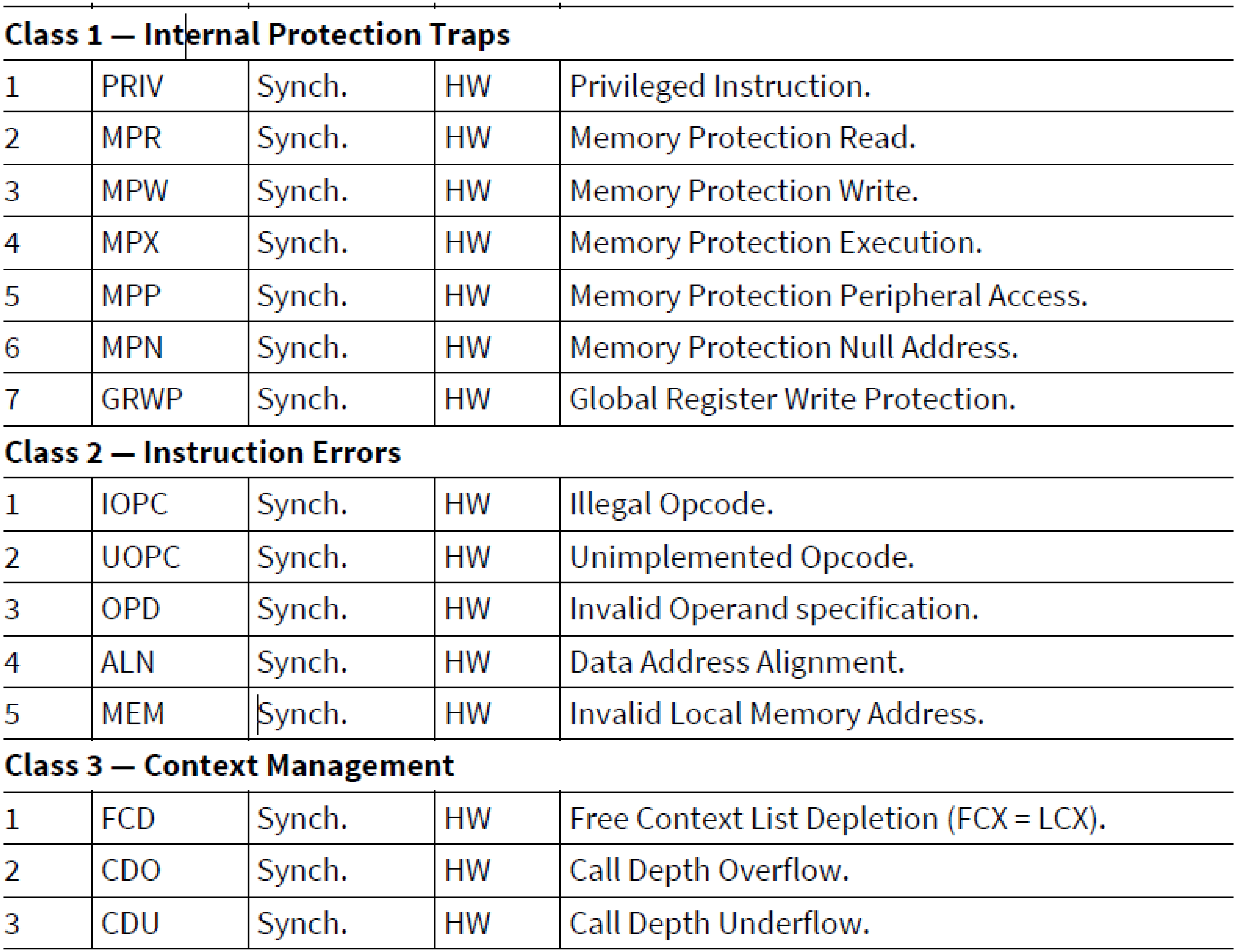
- A trap is a service routine, which typically occurs as a result of a failure of the application (Some complex systems may use traps for virtulisation or safety support):
  - Illegal access.
  - Non Maskable interrupts.
  - Memory protection violation.
  - Etc.
- The trap system is always active: it can not be disabled.
- A trap has two identification components:
  - Trap Class Number (TCN).
  - Trap Identification Number (TIN).
- The CPU aborts the instruction in progress when a Trap occurs and forces execution of the appropriate TSR from the Trap Vector Table

TC1.6.2P Architecture
Trap Types

- Synchronous
  - Associated with execution or attempted execution of specific instructions.
  - The instruction causing the trap is known precisely.
- Asynchronous
  - Similar to interrupts, in that they are associated with hardware conditions detected externally & signalled back to the core.

- Hardware
  - Generated as result of certain TriCore instructions. Examples are illegal instruction traps, memory protection traps, & data memory address misalignment traps.
- Software
  - Include system calls & assertion traps.

TC1.6.2P Architecture Supported Traps



Class 1 — Internal Protection Traps				
1	PRIV	Synch.	HW	Privileged Instruction.
2	MPR	Synch.	HW	Memory Protection Read.
3	MPW	Synch.	HW	Memory Protection Write.
4	MPX	Synch.	HW	Memory Protection Execution.
5	MPP	Synch.	HW	Memory Protection Peripheral Access.
6	MPN	Synch.	HW	Memory Protection Null Address.
7	GRWP	Synch.	HW	Global Register Write Protection.
Class 2 — Instruction Errors				
1	IOPC	Synch.	HW	Illegal Opcode.
2	UOPC	Synch.	HW	Unimplemented Opcode.
3	OPD	Synch.	HW	Invalid Operand specification.
4	ALN	Synch.	HW	Data Address Alignment.
5	MEM	Synch.	HW	Invalid Local Memory Address.
Class 3 — Context Management				
1	FCD	Synch.	HW	Free Context List Depletion (FCX = LCX).
2	CDO	Synch.	HW	Call Depth Overflow.
3	CDU	Synch.	HW	Call Depth Underflow.

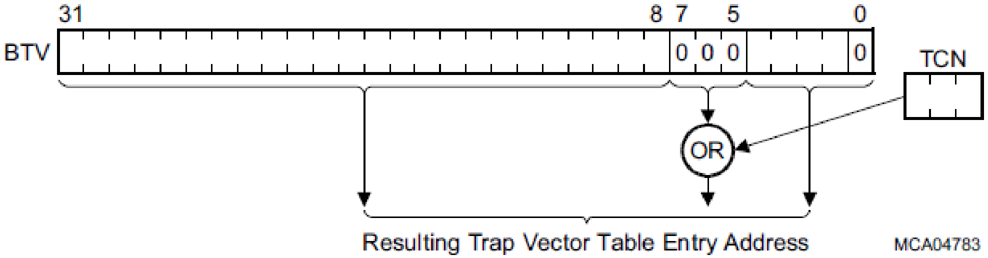
TC1.6.2P Architecture Supported Traps

TIN	Name	Synch. / Asynch.	HW / SW	Definition
4	FCU	Synch.	HW	Free Context List Underflow (FCX = 0).
5	CSU	Synch.	HW	Call Stack Underflow (PCX = 0).
6	CTYP	Synch.	HW	Context Type (PCXI.UL wrong).
7	NEST	Synch.	HW	Nesting Error: RFE with non-zero call depth.
Class 4 — System Bus and Peripheral Errors				
1	PSE	Synch.	HW	Program Fetch Synchronous Error.
2	DSE	Synch.	HW	Data Access Synchronous Error.
3	DAE	Asynch.	HW	Data Access Asynchronous Error.
4	CAE	Asynch	HW	Coprocessor Trap Asynchronous Error.
5	PIE	Synch	HW	Program Memory Integrity Error.
6	DIE	Asynch	HW	Data Memory Integrity Error.
7	TAE	Asynch	HW	Temporal Asynchronous Error
Class 5— Assertion Traps				
1	OVF	Synch.	SW	Arithmetic Overflow.
2	SOVF	Synch.	SW	Sticky Arithmetic Overflow.

Class 6 — System Call[1)]				
	SYS	Synch.	SW	System Call.
Class 7 — Non-Maskable Interrupt				
0	NMI	Asynch.	HW	Non-Maskable Interrupt.

TC1.6.2P Architecture
Entering an Trap Service Routine (TSR)

- On entry to a Trap Service Routine the following operations are performed.
  - Upper context is saved and return address A11 updated
  - The stack pointer (A10) is set to the interrupt stack pointer
  - The IO privilege mode is set to Supervisor
  - Memory protection is set to protection set 2’b00
  - The Call depth counter (CDC) is cleared and the depth limit set to 64
  - Global write permissions are DISABLED
  - Interrupts are disabled (ICR.IE = 0)
  - The previous ICR.IE value is saved in PCXI.PIE
  - **D15** is updated with the trap identification number (**TIN**)
  - The Trap Vector table is accessed to fetch the first instruction in the TSR
    - The Vector address is generated from the Trap vector Table base address (**BTV**) and the trap class number (**TCN**)



TC1.6.2P Architecture
TriCore Protection System

- TriCore protection system aims at :
  - Protecting core system functionality.
  - Protecting application against each other.
  - Providing test and debug functionality.

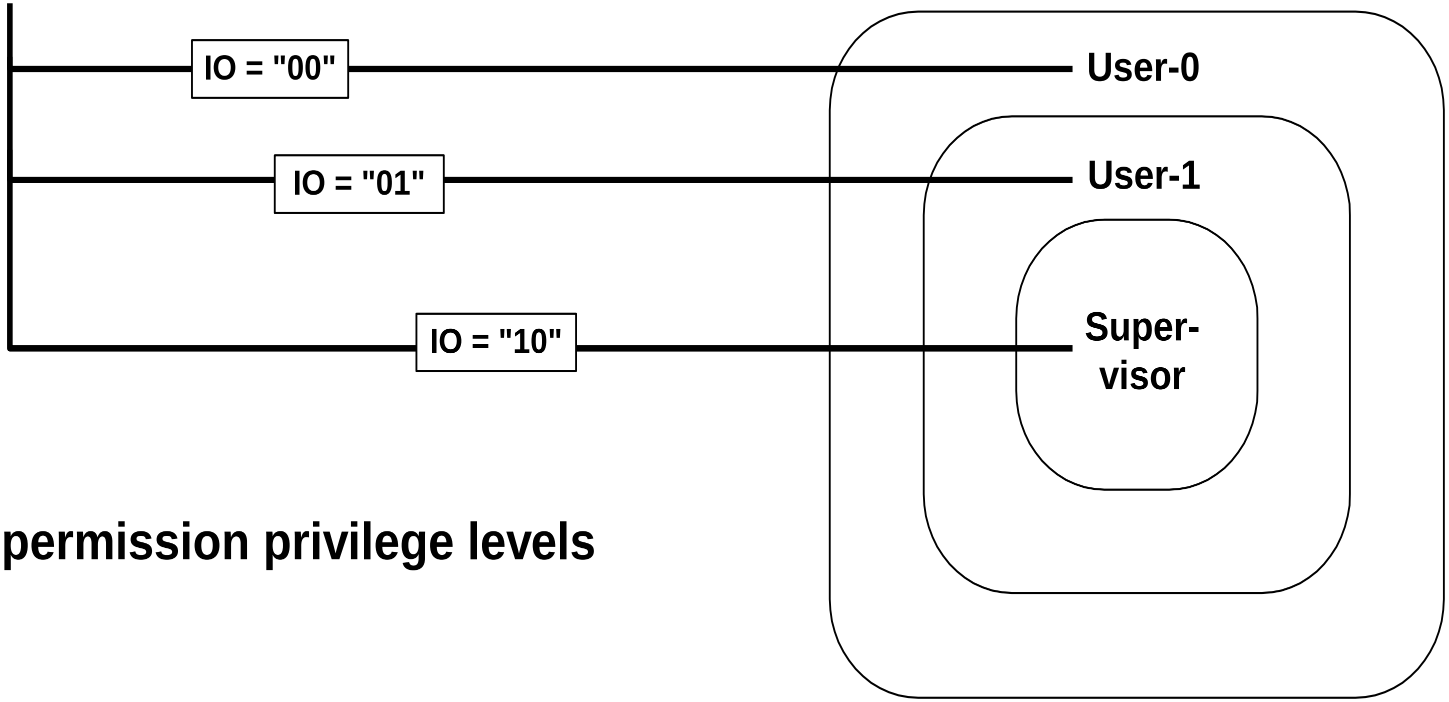
- TriCore protection system is based on:
  - Traps.
  - Permission levels.
  - Memory protection.

TC1.6.2P Architecture
Permission Privilege Levels

Program Status Word (PSW)

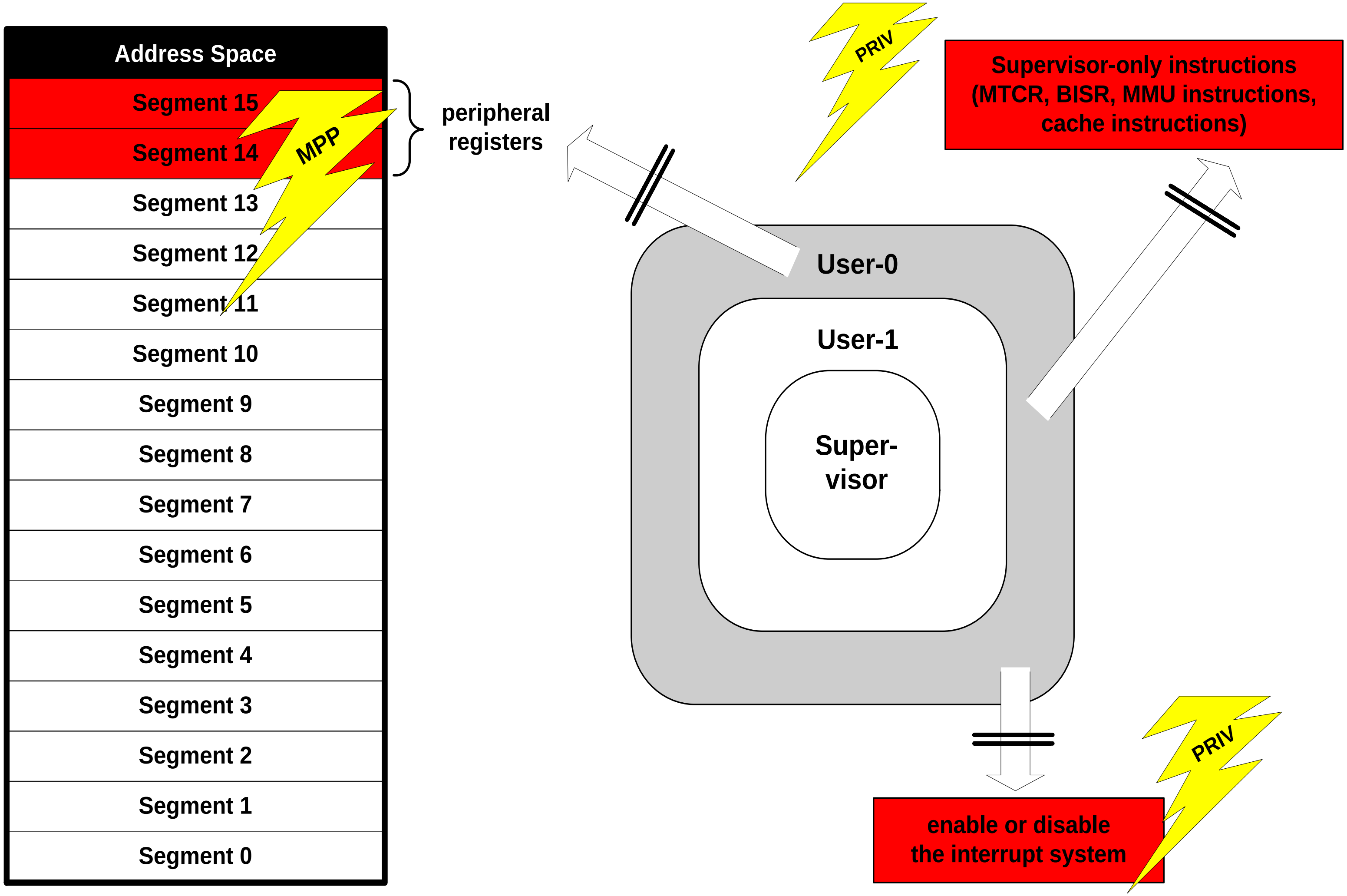
31					16
C	V	SV	AV	SAV	res

15		11 10				0
res	PRS	IO	IS	GW	CDE	CDC

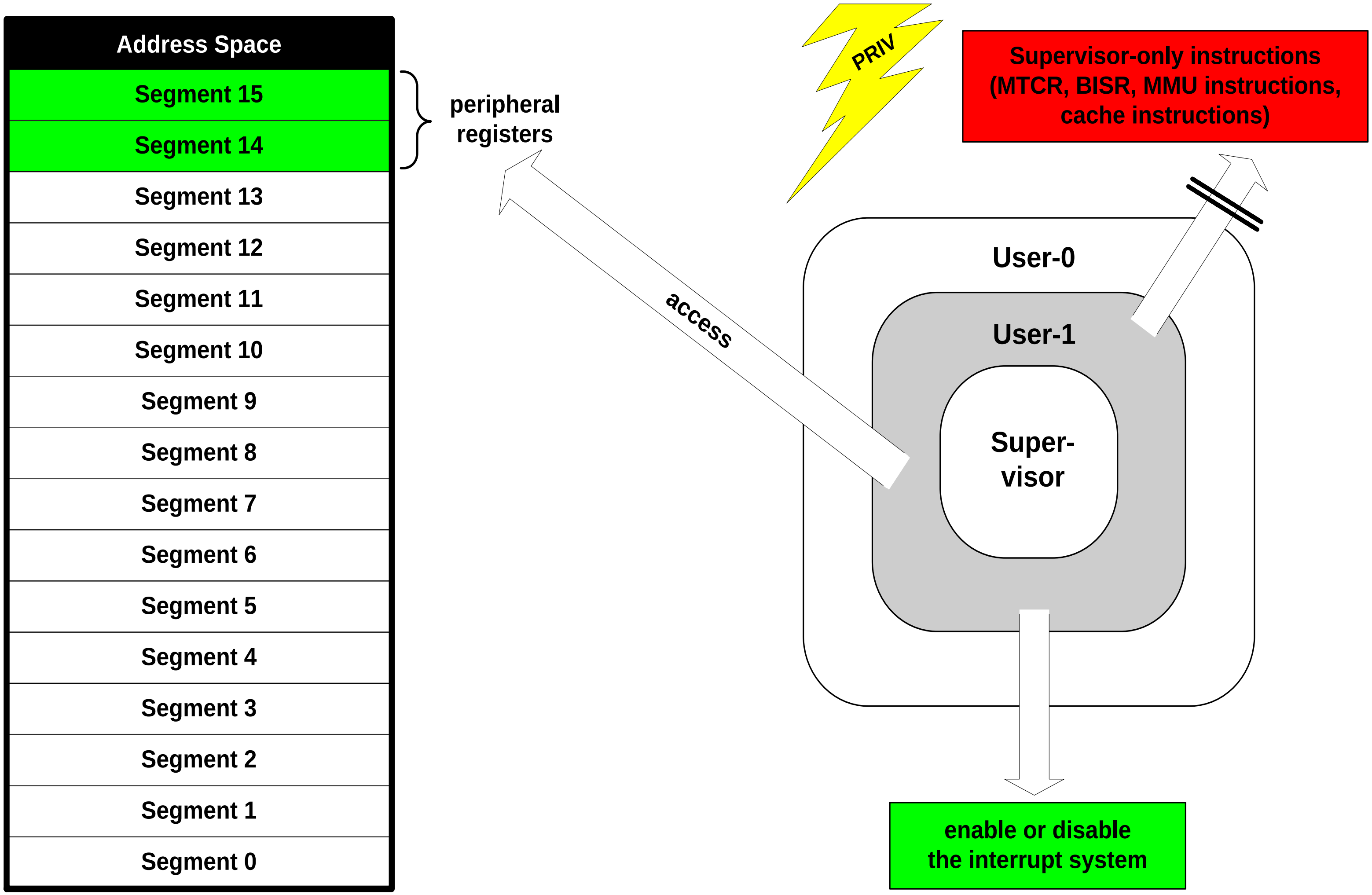


permission privilege levels

TC1.6.2P Architecture Permissions of User-0



TC1.6.2P Architecture
Permissions of User-1



Address Space

Segment 15

Segment 14

Segment 13

Segment 12

Segment 11

Segment 10

Segment 9

Segment 8

Segment 7

Segment 6

Segment 5

Segment 4

Segment 3

Segment 2

Segment 1

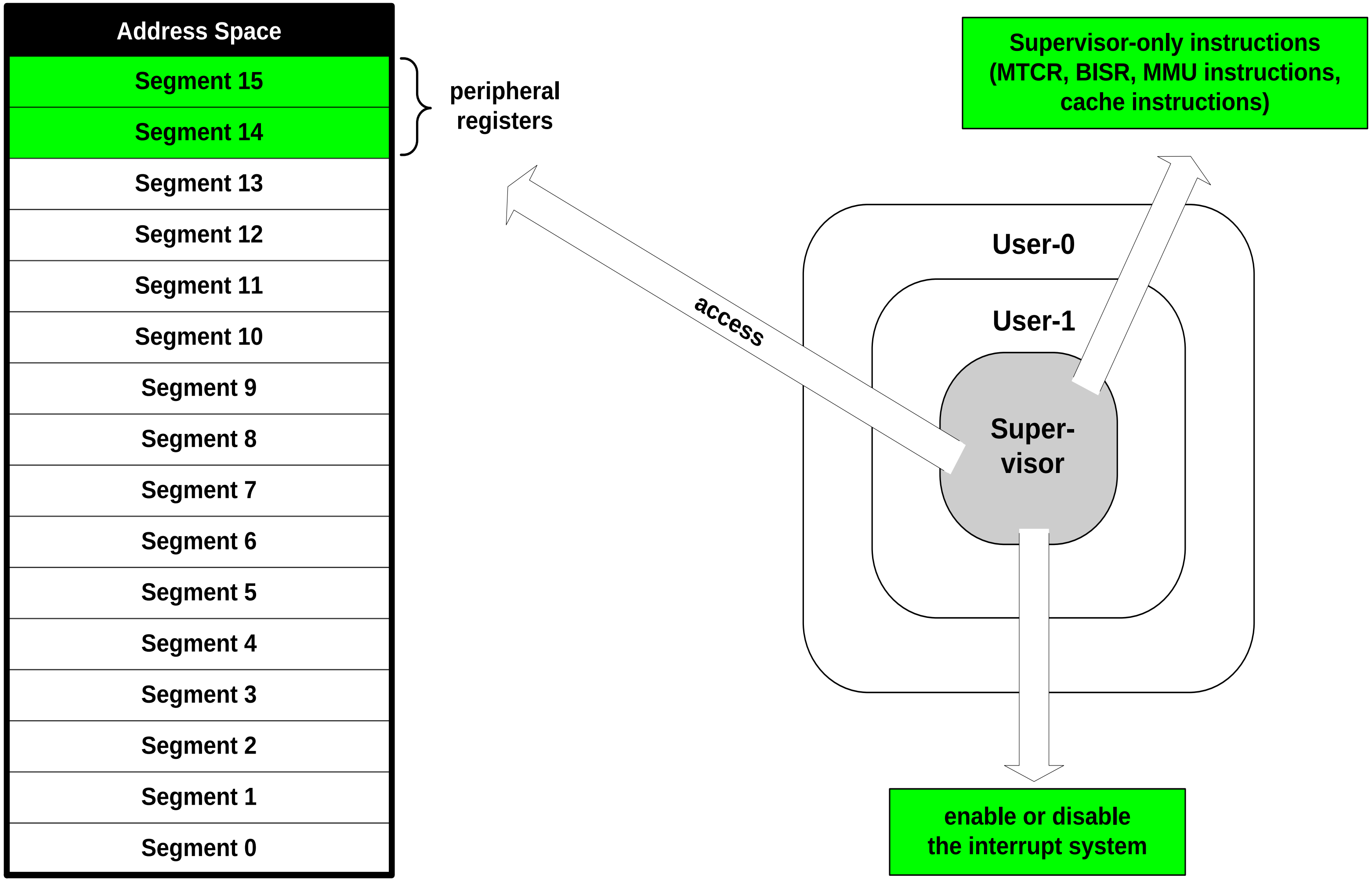
Segment 0

peripheral registers

Supervisor-only instructions (MTCR, BISR, MMU instructions, cache instructions)

enable or disable the interrupt system

TC1.6.2P Architecture
Permissions of Supervisor



Address Space

Segment 15

Segment 14

Segment 13

Segment 12

Segment 11

Segment 10

Segment 9

Segment 8

Segment 7

Segment 6

Segment 5

Segment 4

Segment 3

Segment 2

Segment 1

Segment 0

peripheral registers

access

Supervisor-only instructions (MTCR, BISR, MMU instructions, cache instructions)

User-0

User-1

Super-visor

enable or disable the interrupt system

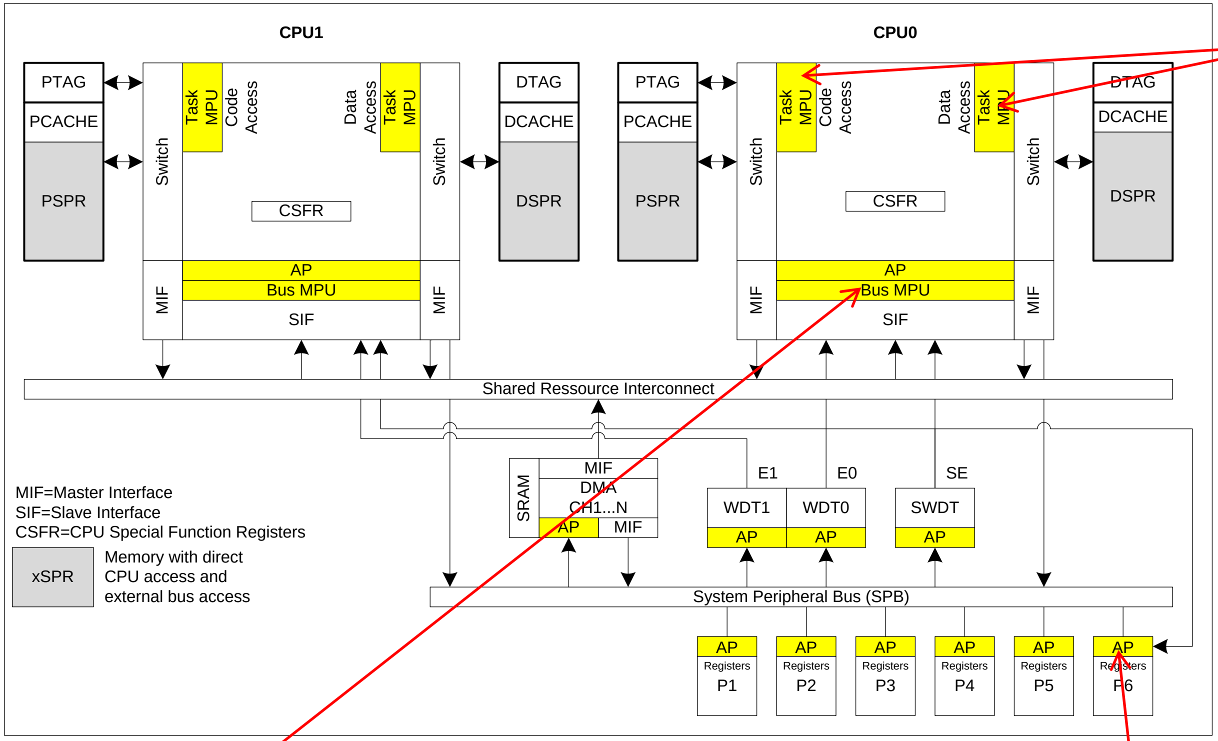
TC1.6.2P Architecture
Memory Protection and Temporal Protection

- Flexible **memory protection** system
  - 18* data memory protection ranges (read or write access right)
  - 10* code memory protection ranges (code fetch access right)
  - 6* memory protection register sets
- **Temporal protection** system allowing time bounded real time operation.
  - A dedicated exception timer *

(* delta to previous core version)

AURIX™ TC3xx Protection System

Enhanced CPU MPU: 18 data and 10 code Ranges



CPU MPU

- → Monitors **direct** access to Local Memories
- → Applies to SW Tasks
- → **18** data and **10** code ranges; organized in 6 sets; switched via PSW;
- → Dynamic re-configuration possible (typically OS)
- → Scope: whole address space

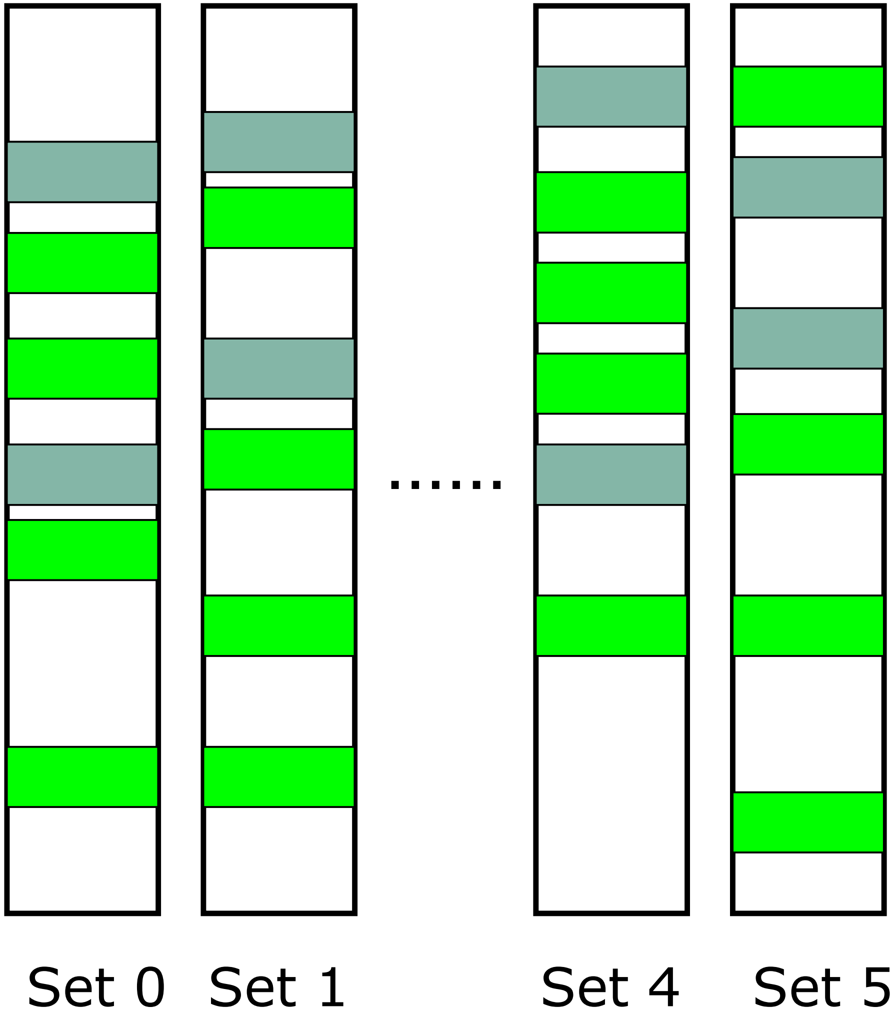
Bus MPU

- → Monitors SRAM accesses via SRI BUS
- → Static or dynamic configuration
- → Scope: local SRAM address space

Register Access Protection

- → Monitors write accesses to module registers
- → Static configuration
- → Scope: register address space

TriCore Memory Protection Overview



🟩 Data access range ▨ Fetch access range

- Access permissions defined by address ranges
- Separate ranges dedicated for data access and program code fetch
- Data access permission defined by any combination of:
  - Read Enable
  - Write Enable
- Fetch Access permission
  - Access Enable
- Fast permission switching -> Protection Sets

TriCore Memory Protection Feature Comparison Table

		TC1.3	TC1.3.1	TC1.6	TC1.6P & E	TC1.6.2P
Protection Sets		2	4	4	4	6
Data Protection Ranges	Per Set	4	4	8	16	18
	Total	8	16	16	16	18
Code Protection Ranges	Per Set	2	2	4	8	10
	Total	4	8	8	8	10
Range to Set assignment		fixed		flexible*	flexible	flexible
Ranges Shared with Debug system		yes		no		no
Range Granularity	Data	1 bytes		8 bytes		8 bytes
	Code	2 bytes				32 bytes
Miscellaneous				TC1.3x compatibility mode	Selective Peripheral Space Protection	global SRAM master tag protection

TC1.6 Temporal Protection System Problem

- In a real time system each task is assigned a defined memory resource and a defined runtime.
  - Overrunning either the assigned physical memory or the assigned runtime can lead to overall program failure.
- The Memory protection stops a program overrunning its assigned region of memory.
- How do we prevent a task overrunning its assigned runtime?
  - Particularly challenging if the task is (necessarily) allowed to disable interrupts for periods of time.

- Primary application: Temporal task protection for AutoSAR

TC1.6 Temporal Protection System Solution

- The TriCore™ Temporal Protection System is used to guard against run-time over-run. The system consists of two primary mechanisms:
  - the temporal protection timers
  - the exception timers

- The temporal protection timers system
  - Consists of three independent decrementing 32 bit counters.
    - Counter activated when writing a non-zero value
    - Counter disabled on load of zero or when core is in Idle
  - Generate a Temporal Asynchronous Exception (TAE) trap (Class-4, Tin-7) on decrement to zero.
- The exception timer system
  - provides a method of detecting the overrun of exception handlers in the system.

TC1.6 Temporal Protection System

- Solution
  - Use a counter to monitor the task runtime, Call the OS via a TRAP (rather than interrupt) if runtime is exceeded
    - Two 32-bit decrementing counters clocked by the primary clock
    - Loaded with expected runtime at task start.
    - Easily loaded/unloaded with task state.
    - TAE trap generated on decrement to zero.
    - Counter disabled on load of zero
    - Counter disabled when core is in Idle
    - Counter value and status available via CSFR space.
      - SYSCON register extended with global enable/disable for Temporal Protection
      - Control register Time-out flags, Trap flag
- Primary application: Temporal task protection for AutoSAR

Agenda

1 Architectural Overview

2 Memory Concept

3 On-chip Bus Connectivity

4 NVM System

5 TC1.6.2 Architecture

6 Summary

Summary of Functional Changes from AURIX

- New instructions (See architecture manual for details):-
  - CRC32B.W, CRC32L.W, CRC32.B (CRC32 for big endian, little endian and byte data)
  - CRCN (arbitrary width and polynomial CRC calculation)
  - SHUFFLE (Reorder bytes within word)
  - POPCNT (count number of bits set in word)
  - FTOHP, HPTOF (Half precision floating point conversion)
  - LHA (Load high bits of address value)
- Enhanced memory protection
  - Number of protection sets increased to 6 (was 4), The PSW.PRS field has been extended to reflect this.
  - Number of code protection ranges increased to 10 (was 8)
  - Number of data protection ranges increased to 18 (was 16)
- The temporal protection system is extended to provide a dedicated exception timer.

Summary of Functional Changes from AURIX

- Independent core resets implemented. (Individual cores may be independently reset as required)
- To exit boot halt the SYSCON.BHALT should be cleared (Was DBGSR.HALT)
- A portion of the LMU memory (called DLMU) is distributed between the processors to provide high performance access to global SRAM
- The PFlash memory is distributed between the processors to provide high performance access to a local PFlash bank. (LPB)
- The overlay system is extended to support additional processor cores.
- The store buffer data merge functionality is extended to merge consecutive half words into words and consecutive words into double words.
- The safety protection system has been extended to cover external read and write accesses to local DSPR/PSPR and DLMU, and to cover external read accesses to the LPB.
- The CPU_ID has changed (to 0x00C0C020 for TC39X A-Step)

TC1.6.2P Architecture Manual

- Volume 1: Core Architecture
- Volume 2: Instruction Set

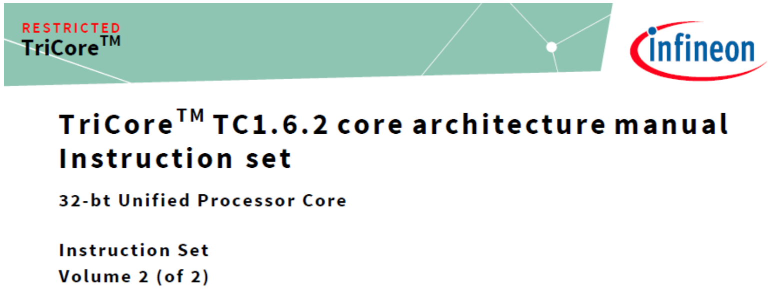
RESTRICTED

TriCore™

TriCore™ TC1.6.2 core architecture manual

32-bt microcontroller

Core architecture
Volume 1 (of 2)



RESTRICTED

TriCore™

TriCore™ TC1.6.2 core architecture manual
Instruction set

32-bt Unified Processor Core

Instruction Set
Volume 2 (of 2)

1. How to get them?
2. User need register account at www.myinfineon.com to download the “TriCore™ TC1.6.2 core architecture manual” after being authorized the access right of AURIX 2G documents.

Part of your life. Part of tomorrow.