

Welcome to [E-XFL.COM](https://www.e-xfl.com)

### Understanding [Embedded - Microprocessors](#)

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

### Applications of [Embedded - Microprocessors](#)

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

#### Details

Product Status	Obsolete
Core Processor	ARM® Cortex®-A5
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	600MHz
Co-Processors/DSP	Multimedia; NEON™ SIMD
RAM Controllers	LPDDR, LPDDR2, DDR2
Graphics Acceleration	Yes
Display & Interface Controllers	LCD, Touchscreen
Ethernet	10/100Mbps (1)
SATA	-
USB	USB 2.0 (3)
Voltage - I/O	1.2V, 1.8V, 3.3V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	AES, SHA, TDES, TRNG
Package / Case	361-TFBGA
Supplier Device Package	361-TFBGA (16x16)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/atsama5d44a-cur">https://www.e-xfl.com/product-detail/microchip-technology/atsama5d44a-cur</a>

# SAMA5D4 SERIES

## 11.4.1 Boot Sequence Controller Configuration Register

**Name:**BSC\_CR

**Access:**Read/Write

**Factory Value:** 0x0000\_0000

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

### BOOT: Boot Media Sequence

This value is defined in section “Standard Boot Strategies”. It is only written if WPKEY carries the valid value.

### WPKEY: Write Protection Key (Write-only)

Value	Name	Description
0x6683	PASSWD	Writing any other value in this field aborts the write operation of the BOOT field. Always reads as 0.

## 13.5.18 L2CC Invalidate Way Register

**Name:**L2CC\_IWR

**Address:**0x00A0077C

**Access:**Read/Write

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
WAY7	WAY6	WAY5	WAY4	WAY3	WAY2	WAY1	WAY0

**WAYx: Invalidate Way Number x**

0: The corresponding way is totally invalidated.

1: Invalidates the way. This bit is read as '1' as long as invalidation of the way is in progress.

**28.6.29 PIO Input Filter Slow Clock Status Register****Name:**PIO\_IFSCSR**Address:**0xFC06A088 (PIOA), 0xFC06B088 (PIOB), 0xFC06C088 (PIOC), 0xFC068088 (PIOD), 0xFC06D088 (PIOE)**Access:**Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

**P0–P31: Glitch or Debouncing Filter Selection Status**0: The glitch filter is able to filter glitches with a duration  $< t_{\text{peripheral clock}}/2$ .1: The debouncing filter is able to filter pulses with a duration  $< t_{\text{div\_slck}}/2$ .

# SAMA5D4 SERIES

## 29.7.15 MPDDRC OCMS KEY2 Register

Name:MPDDRC\_OCMS\_KEY2

Access:Write once

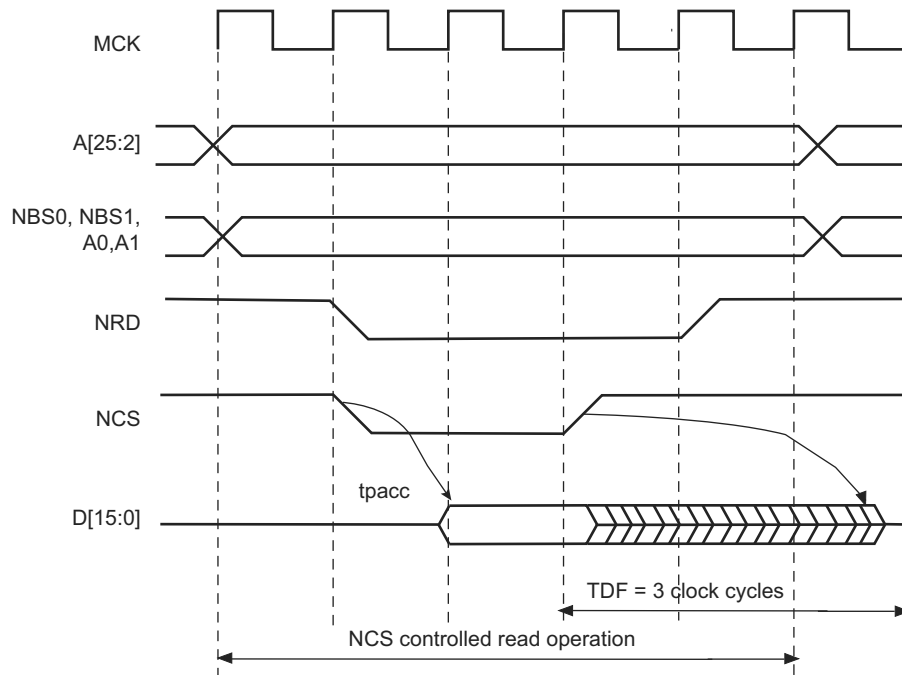
31	30	29	28	27	26	25	24
KEY2							
23	22	21	20	19	18	17	16
KEY2							
15	14	13	12	11	10	9	8
KEY2							
7	6	5	4	3	2	1	0
KEY2							

This register can only be written if the WPEN bit is cleared in the MPDDRC Write Protection Mode Register.

### KEY2: Off-chip Memory Scrambling (OCMS) Key Part 2

When Off-chip Memory Scrambling is enabled, the data scrambling depends on KEY1 and KEY2 values.

**Figure 30-18: TDF Period in NCS Controlled Read Operation (TDF = 3)**



### 30.13.2 TDF Optimization Enabled (TDF\_MODE = 1)

When the TDF\_MODE of the HSMC\_MODE register is set to 1 (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

Figure 30-19 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

NRD\_HOLD = 4; READ\_MODE = 1 (NRD controlled)

NWE\_SETUP = 3; WRITE\_MODE = 1 (NWE controlled)

TDF\_CYCLES = 6; TDF\_MODE = 1 (optimization enabled).

30.20.10 PMECC Spare Area Size Register

Name: HSMC\_PMECCSAREA

Address: 0xFC05C074

Access: Read/Write

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

SPARESIZE: Spare Area Size

Number of bytes in the spare area. The spare area size is equal to (SPARESIZE + 1) bytes.

# SAMA5D4 SERIES

## 30.20.11 PMECC Start Address Register

**Name:** HSMC\_PMECCSADDR

**Address:** 0xFC05C078

**Access:** Read/Write

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

### STARTADDR: ECC Area Start Address

This register is programmed with the start ECC start address. When STARTADDR is equal to 0, then the first ECC byte is located at the first byte of the spare area.



## 32.6.4.12 19 bpp Unpacked Memory Mapping with Transparency Bit, RGB 1:6:6:6

**Table 32-23: 19 bpp Unpacked Memory Mapping, Little Endian Organization**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 19 bpp	–								–								A0								R0[5:0]							

## 32.6.4.13 19 bpp Packed Memory Mapping with Transparency Bit, ARGB 1:6:6:6

**Table 32-24: 19 bpp Packed Memory Mapping, Little Endian Organization at Address 0x0, 0x1, 0x2, 0x3**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 19 bpp	G1[1:0]								B1[5:0]								–								A0							

**Table 32-25: 19 bpp Packed Memory Mapping, Little Endian Organization at Address 0x4, 0x5, 0x6, 0x7**

Mem addr	0x7								0x6								0x5								0x4							
Bit	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
Pixel 19 bpp	R2[3:0]								G2[5:0]								B2[5:0]								–							

**Table 32-26: 18 bpp Packed Memory Mapping, Little Endian Organization at Address 0x8, 0x9, 0xA, 0xB**

Mem addr	0xB								0xA								0x9								0x8							
Bit	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
Pixel 19 bpp	G4[1:0]								B4[5:0]								–								A3							

## 32.6.4.14 24 bpp Unpacked Memory Mapping, RGB 8:8:8

**Table 32-27: 24 bpp Memory Mapping, Little Endian Organization**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 24 bpp	–								R0[7:0]								G0[7:0]								B0[7:0]							

## 32.6.4.15 24 bpp Packed Memory Mapping, RGB 8:8:8

**Table 32-28: 24 bpp Packed Memory Mapping, Little Endian Organization at Address 0x0, 0x1, 0x2, 0x3**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 24 bpp	B1[7:0]								R0[7:0]								G0[7:0]								B0[7:0]							

**Table 32-29: 24 bpp Packed Memory Mapping, Little Endian Organization at Address 0x4, 0x5, 0x6, 0x7**

Mem addr	0x7								0x6								0x5								0x4							
Bit	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
Pixel 24 bpp	G2[7:0]								B2[7:0]								R1[7:0]								G1[7:0]							

**Table 32-37: 16 bpp 4:2:2 interleaved Mode 3**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 16 bpp	Y1[7:0]								Cb0[7:0]								Y0[7:0]								Cr0[7:0]							

## 32.6.5.3 4:2:2 Semiplanar Mode Frame Buffer Memory Mapping

**Table 32-38: 4:2:2 Semiplanar Luminance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 16 bpp	Y3[7:0]								Y2[7:0]								Y1[7:0]								Y0[7:0]							

**Table 32-39: 4:2:2 Semiplanar Chrominance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 16 bpp	Cb2[7:0]								Cr2[7:0]								Cb0[7:0]								Cr0[7:0]							

## 32.6.5.4 4:2:2 Planar Mode Frame Buffer Memory Mapping

**Table 32-40: 4:2:2 Planar Mode Luminance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 16 bpp	Y3[7:0]								Y2[7:0]								Y1[7:0]								Y0[7:0]							

**Table 32-41: 4:2:2 Planar Mode Chrominance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 16 bpp	C3[7:0]								C2[7:0]								C1[7:0]								C0[7:0]							

## 32.6.5.5 4:2:0 Planar Mode Frame Buffer Memory Mapping

In Planar Mode, the three video components Y, Cr and Cb are split into three memory areas and stored in a raster-scan order. These three memory planes are contiguous and always aligned on a 32-bit boundary.

**Table 32-42: 4:2:0 Planar Mode Luminance Memory Mapping, Little Endian Organization for Byte 0x0, 0x1, 0x2, 0x3**

Mem addr	0x3								0x2								0x1								0x0							
Bit	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
Pixel 12 bpp	Y3[7:0]								Y2[7:0]								Y1[7:0]								Y0[7:0]							

A simple algorithm can be used by the application to send packets regardless of the number of banks associated to the endpoint.

Algorithm Description for Each Packet:

- The application waits for TXRDY flag to be cleared in the UDPHS\_EPTSTAx register before it can perform a write access to the DPR.
- The application writes one USB packet of data in the DPR through the 64 KB endpoint logical memory window.
- The application sets TXRDY flag in the UDPHS\_EPTSETSTAx register.

The application is notified that it is possible to write a new packet to the DPR by the TXRDY interrupt. This interrupt can be enabled or masked by setting the TXRDY bit in the UDPHS\_EPTCTLENB/UDPHS\_EPTCTLDIS register.

Algorithm Description to Fill Several Packets:

Using the previous algorithm, the application is interrupted for each packet. It is possible to reduce the application overhead by writing linearly several banks at the same time. The AUTO\_VALID bit in the UDPHS\_EPTCTLx must be set by writing the AUTO\_VALID bit in the UDPHS\_EPTCTLENBx register.

The auto-valid-bank mechanism allows the transfer of data (IN and OUT) without the intervention of the CPU. This means that bank validation (set TXRDY or clear the RXRDY\_TXKL bit) is done by hardware.

- The application checks the BUSY\_BANK\_STA field in the UDPHS\_EPTSTAx register. The application must wait that at least one bank is free.
- The application writes a number of bytes inferior to the number of free DPR banks for the endpoint. Each time the application writes the last byte of a bank, the TXRDY signal is automatically set by the UDPHS.
- If the last packet is incomplete (i.e., the last byte of the bank has not been written) the application must set the TXRDY bit in the UDPHS\_EPTSETSTAx register.

The application is notified that all banks are free, so that it is possible to write another burst of packets by the BUSY\_BANK interrupt. This interrupt can be enabled or masked by setting the BUSY\_BANK flag in the UDPHS\_EPTCTLENB and UDPHS\_EPTCTLDIS registers.

This algorithm must not be used for isochronous transfer. In this case, the ping-pong mechanism does not operate.

A Zero Length Packet can be sent by setting just the TXRDY flag in the UDPHS\_EPTSETSTAx register.

### *Bulk IN or Interrupt IN: Sending a Buffer Using DMA (Device to Host)*

The UDPHS integrates a DMA host controller. This DMA controller can be used to transfer a buffer from the memory to the DPR or from the DPR to the processor memory under the UDPHS control. The DMA can be used for all transfer types except control transfer.

Example DMA configuration:

1. Program UDPHS\_DMAADDRESS x with the address of the buffer that should be transferred.
2. Enable the interrupt of the DMA in UDPHS\_IEN
3. Program UDPHS\_DMACONTROLx:
  - Size of buffer to send: size of the buffer to be sent to the host.
  - END\_B\_EN: The endpoint can validate the packet (according to the values programmed in the AUTO\_VALID and SHRT\_PCKT fields of UDPHS\_EPTCTLx.) (Refer to Section 35.7.12 “UDPHS Endpoint Control Disable Register (Isochronous Endpoint)” and Figure 35-13)
  - END\_BUFFIT: generate an interrupt when the BUFF\_COUNT in UDPHS\_DMASTATUSx reaches 0.
  - CHANN\_ENB: Run and stop at end of buffer

The auto-valid-bank mechanism allows the transfer of data (IN & OUT) without the intervention of the CPU. This means that bank validation (set TXRDY or clear the RXRDY\_TXKL bit) is done by hardware.

A transfer descriptor can be used. Instead of programming the register directly, a descriptor should be programmed and the address of this descriptor is then given to UDPHS\_DMANXTDSC to be processed after setting the LDNXT\_DSC field (Load Next Descriptor Now) in UDPHS\_DMACONTROLx register.

The structure that defines this transfer descriptor must be aligned.

Each buffer to be transferred must be described by a DMA Transfer descriptor (refer to Section 35.7.21 “UDPHS DMA Channel Transfer Descriptor”). Transfer descriptors are chained. Before executing transfer of the buffer, the UDPHS may fetch a new transfer descriptor from the memory address pointed by the UDPHS\_DMANXTDSCx register. Once the transfer is complete, the transfer status is updated in the UDPHS\_DMASTATUSx register.

To chain a new transfer descriptor with the current DMA transfer, the DMA channel must be stopped. To do so, INTDIS\_DMA and TXRDY may be set in the UDPHS\_EPTCTLENBx register. It is also possible for the application to wait for the completion of all transfers. In this case the LDNXT\_DSC bit in the last transfer descriptor UDPHS\_DMACONTROLx register must be set to 0 and the CHANN\_ENB bit set to 1.

# SAMA5D4 SERIES

---

## **EN\_CK256: Enable 256 Clock Checking**

This bit controls the End of Resume sequence of the EHCI host controller.

By default, the value of this bit is 0 and during the End of Resume sequence, the host controller waits for SE0 on the linestate before switching the PHY to High-Speed.

When set to 1, during the End of Resume sequence, the controller waits for SE0 or 256 clocks before switching the PHY to High-Speed.

Setting this bit to 1 enables the 256-clock check. Some of the UTMI PHYs do not present SE0 on the linestate during the End of Resume sequence. For such PHYs, this bit should be set, so that the core does not wait forever for SE0.

This bit should be set only during initialization.

## 37.8.62 GMAC Late Collisions Register

**Name:**GMAC\_LC

**Address:**0xF8020144 (0), 0xFC028144 (1)

**Access:** Read-only

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

### LCOL: Late Collisions

This register counts the number of late collisions occurring after the slot time (512 bits) has expired. In 10/100 mode, late collisions are counted twice i.e., both as a collision and a late collision.

38.14.9 HSMCI Response Register

Name: HSMCI\_RSPR

Address: 0xF8000020 (0), 0xFC000020 (1)

Access: Read-only

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

RSP: Response

**Note 1:** The response register can be read by N accesses at the same HSMCI\_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

## 38.14.18 HSMCI Write Protection Mode Register

**Name:** HSMCI\_WPMR

**Address:** 0xF80000E4 (0), 0xFC0000E4 (1)

**Access:** Read/Write

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

### WPEN: Write Protect Enable

0: Disables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

1: Enables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

Refer to Section 38.13 “Register Write Protection” for the list of registers that can be write-protected.

### WPKEY: Write Protect Key

Value	Name	Description
0x4D4349	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

## 39. Serial Peripheral Interface (SPI)

### 39.1 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a Shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master” which controls the data flow, while the other devices act as “slaves” which have data shifted into and out by the master. Different CPUs can take turn being masters (multiple master protocol, contrary to single master protocol where one CPU is always the master while all of the others are always slaves). One master can simultaneously shift data into multiple slaves. However, only one slave can drive its output to write data back to the master at any given time.

A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI)—This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO)—This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK)—This control line is driven by the master and regulates the flow of the data bits. The master can transmit data at a variety of baud rates; there is one SPCK pulse for each bit that is transmitted.
- Slave Select (NSS)—This control line allows slaves to be turned on and off by hardware.

### 39.2 Embedded Characteristics

- Master or Slave Serial Peripheral Bus Interface
  - 8-bit to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delay between consecutive transfers and delay before SPI clock per chip select
  - Programmable delay between chip selects
  - Selectable mode fault detection
- Master Mode can drive SPCK up to Peripheral Clock
- Master Mode Bit Rate can be Independent of the Processor/Peripheral Clock
- Slave mode operates on SPCK, asynchronously with core and bus clock
- Four chip selects with external decoder support allow communication with up to 15 peripherals
- Communication with Serial External Devices Supported
  - Serial memories, such as DataFlash and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD controllers, CAN controllers and sensors
  - External coprocessors
- Connection to DMA Channel Capabilities, Optimizing Data Transfers
  - One channel for the receiver
  - One channel for the transmitter
- Register Write Protection



Figure 47-18: Synchronized Update of Update Period Value of Synchronous Channels

▼

▲

#### 47.6.5.5 Changing the Comparison Value and the Comparison Configuration

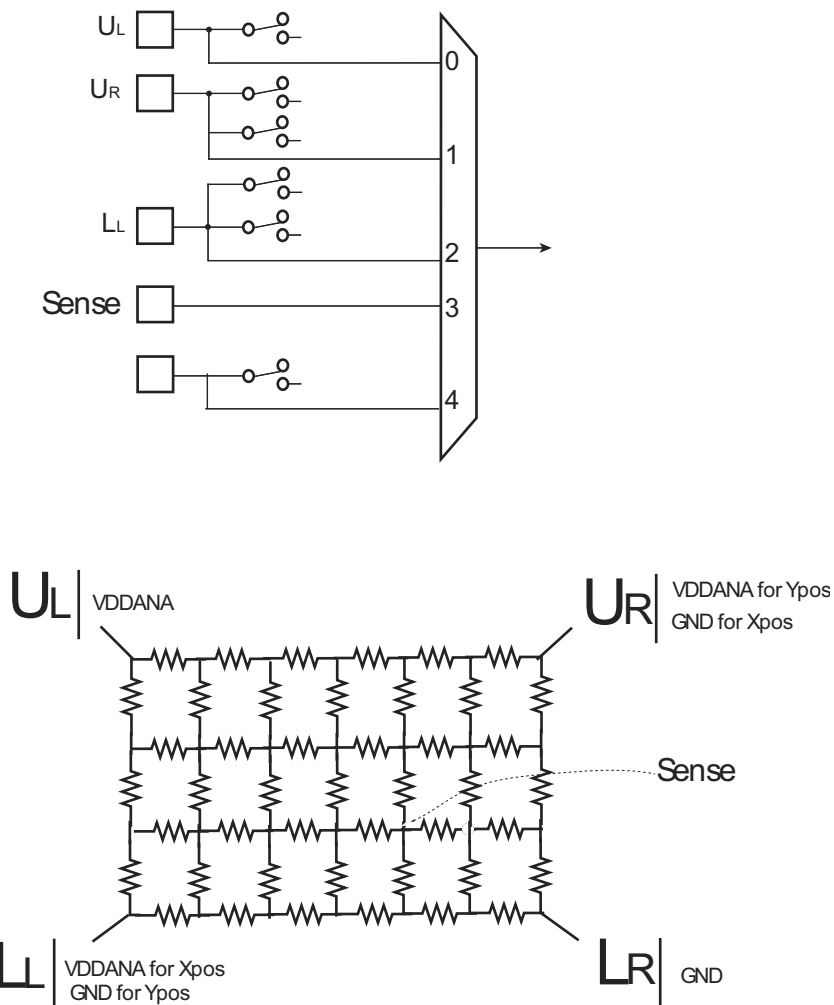
It is possible to change the comparison values and the comparison configurations while the channel 0 is enabled (refer to Section 47.6.3 "PWM Comparison Units").

To prevent unexpected comparison match, the user must use the PWM Comparison x Value Update Register (PWM\_CMPVUPDx) and the PWM Comparison x Mode Update Register (PWM\_CMPMUPDx) to change, respectively, the comparison values and the comparison configurations while the channel 0 is still enabled. These registers hold the new values until the end of the comparison update period (when CUPRCNT is equal to CUPR in PWM Comparison x Mode Register (PWM\_CMPMx) and the end of the current PWM period, then update the values for the next period.

**CAUTION:** The write of the register PWM\_CMPVUPDx must be followed by a write of the register PWM\_CMPMUPDx.

**Note:** If the update registers PWM\_CMPVUPDx and PWM\_CMPMUPDx are written several times between two updates, only the last written value are taken into account.

**Figure 48-13: Touchscreen Switches Implementation**



## 48.6.11.7 Sequence and Noise Filtering

The ADC Controller can manage ADC conversions and touchscreen measurement. On each trigger event the sequence of ADC conversions is performed as described in Section 48.6.7 “Sleep Mode and Conversion Sequencer”. The touchscreen measure frequency can be specified in number of trigger events by writing the TSFREQ parameter in ADC\_TSMR. An internal counter counts triggers up to TSFREQ, and every time it rolls out, a touchscreen sequence is appended to the classic ADC conversion sequence (see Figure 48-14).

Additionally the user can average multiple touchscreen measures by writing the TSAV parameter in ADC\_TSMR. This can be 1, 2, 4 or 8 measures performed on consecutive triggers as illustrated in Figure 48-14 below. Consequently, the TSFREQ parameter must be greater or equal to the TSAV parameter.

---

- If the message is received through a serial-to-parallel communication channel, the first received character is 0xca and it is stored at the first memory location (initial offset). The second byte, 0xfe, is stored at initial offset + 1.

When the SHA\_IODATARx registers are read, the hash result is organized in little-endian format, allowing system memory storage in the same format as the message.

For this example, the 512-bit message is:

and the expected SHA-256 result is:

If the message has not already been stored in the system memory, the first step is to convert the input message to little-endian before writing to the SHA IDATARx registers. This would result in a write of:

The data in the output message digest registers, SHA\_IODATARx, contain SHA\_IODATAR0 = 0xbf1678ba... SHA\_IODATAR7 = 0xad1500f2 which is the little-endian format of 0xba7816bf....., 0xf20015ad.

When the output message is read, the user can convert back to big-endian for a resulting message value of:

### 52.4.6 Security Features

Several kinds of unspecified register accesses can occur:

- The URAD bit and the URAT field can only be reset by the SWRST bit in the SHA\_CR.

## 54.6 Integrity Check Monitor (ICM) User Interface

**Table 54-9: Register Mapping**

Offset	Register	Name	Access	Reset
0x00	Configuration Register	ICM_CFG	Read/Write	0x0
0x04	Control Register	ICM_CTRL	Write-only	–
0x08	Status Register	ICM_SR	Read-only	–
0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	ICM_IER	Write-only	–
0x14	Interrupt Disable Register	ICM_IDR	Write-only	–
0x18	Interrupt Mask Register	ICM_IMR	Read-only	0x0
0x1C	Interrupt Status Register	ICM_ISR	Read-only	0x0
0x20	Undefined Access Status Register	ICM_UASR	Read-only	0x0
0x24–0x2C	Reserved	–	–	–
0x30	Region Descriptor Area Start Address Register	ICM_DSCR	Read/Write	0x0
0x34	Region Hash Area Start Address Register	ICM_HASH	Read/Write	0x0
0x38	User Initial Hash Value 0 Register	ICM_UIHVAL0	Write-only	–
...	...	...	...	...
0x54	User Initial Hash Value 7	ICM_UIHVAL7	Write-only	–
0x78–0xE8	Reserved	–	–	–
0xEC–0xFC	Reserved	–	–	–

# SAMA5D4 SERIES

**Table 56-46: ISI Timings with Peripheral Supply 3.3V**

Symbol	Parameter	Min	Max	Unit
ISI <sub>1</sub>	DATA/VSYNC/HSYNC setup time	3.2	—	ns
ISI <sub>2</sub>	DATA/VSYNC/HSYNC hold time	0.7	—	ns
ISI <sub>3</sub>	PIXCLK frequency	—	75	MHz

## 56.18 MCI Timings

The High Speed MultiMedia Card Interface (HSMCI) supports the MultiMedia Card (MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification, and CE-ATA V1.1.

## 56.19 Ethernet MAC (GMAC) Timings

### 56.19.1 Timing Conditions

Timings assuming a capacitance load on data and clock are given in Table 56-47.

**Table 56-47: Capacitance Load on Data, Clock Pads**

Supply	Corner	
	Max	Min
3.3V	20 pF	0 pF

### 56.19.2 Timing Constraints

**Table 56-48: Ethernet MAC Signals Relative to GMDC**

Symbol	Parameter	Min	Max	Unit
EMAC <sub>1</sub>	Setup for GMDIO from GMDC rising	10	—	ns
EMAC <sub>2</sub>	Hold for GMDIO from GMDC rising	10	—	ns
EMAC <sub>3</sub>	GMDIO toggling from GMDC rising	0 <sup>(1)</sup>	300 <sup>(1)</sup>	ns

**Note 1:** For Ethernet MAC output signals, minimum and maximum access time are defined. The minimum access time is the time between the GMDC rising edge and the signal change. The maximum access timing is the time between the GMDC rising edge and the signal stabilizes. Figure 56-23 illustrates minimum and maximum accesses for EMAC<sub>3</sub>.

**Figure 56-23: Minimum and Maximum Access Time of Ethernet MAC Output Signals**

