Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	ARM® Cortex®-M4F
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	CANbus, EBI/EMI, I²C, IrDA, LINbus, MMC/SD, QSPI, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM
Number of I/O	51
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	1.71V ~ 3.63V
Data Converters	A/D 24x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-VQFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsame51j18a-mu

1. Configuration Summary

Table 1-1. SAM E53/E54 Family Features with Ethernet

Device	Program Memory (KB)	Data Memory (KB)	Pins	Packages	Peripherals																			Analog					Security				
					Ethernet Controller	CAN-FD	SERCOM	TC/Compare	TCC (24-bit/16-bit)	I2S	USB	QSPI	SDHC	DMA Channels	PCC (data size)	CCL	Position Decoder	RTC	WDT	Frequency Measurement	Event System (Channels)	External Interrupt Lines	I/O Pins	ADC (Channels ADC0/ADC1)	Analog Comparators (Channels)	DAC (Channels)	PTC (Mutual/Self-capacitance Channels)	Temperature Sensor	AES	TRNG	Public Key Cryptography (PUKCC)	Integrity Check Monitor	Tamper Pins
SAME53N20	1024	256	100	TQFP	Y	N	8	8/2	2/3	Y	Y	Y	2	32	14	4	Y	Y	Y	Y	32	16	81	16/12	4	2	256/32	Y	Y	Y	Y	Y	5
SAME53N19	512	192																															
SAME53J20	1024	256	64	TQFP, QFN			6	6/2					1		10								51	16/8									3
SAME53J19	512	192																															
SAME53J18	256	128																															
SAME54P20	1024	256	128 120	TQFP TFBGA		Y	8	8/2					2		14								99	16/16									5
SAME54P19	512	192	128 120	TQFP TFBGA																													
SAME54N20	1024	256	100	TQFP																			81	16/12									
SAME54N19	512	192																															

Table 1-2. SAM D51/E51 Family Features without Ethernet

Device	Program Memory (KB)	Data Memory (KB)	Pins	Packages	Peripherals																	Analog					Security				
					CAN-FD	SERCOM	TC/Compare	TCC (24-bit/16-bit)	I2S	USB	QSPI	SDHC	DMA Channels	PCC (data size)	CCL	Position Decoder	RTC	Frequency Measurement	Event System (Channels)	External Interrupt Lines	I/O Pins	ADC (Channels ADC0/ADC1)	Analog Comparators (Channels)	DAC (Channels)	PTC (Mutual/Self-capacitance Channels)	Temperature Sensor	AES	TRNG	Public Key Cryptography (PUKCC)	Integrity Check Monitor	Tamper Pins
SAMD51P20	1024	256	128 120	TQFP TFBGA	N	8	8/2	2/3	Y	Y	Y	2	32	14	4	Y	Y	Y	32	16	99	16/16	4	2	256/32	Y	Y	Y	Y	Y	5
SAMD51P19	512	192																															
SAMD51N20	1024	256	100	TQFP																	81	16/12									
SAMD51N19	512	192																															
SAMD51J20	1024	256	64	TQFP, QFN, WLCSP		6	6/2					1		10							51	16/8									3
SAMD51J19	512	192																															
SAMD51J18	256	128	64	TQFP, QFN																													
SAMD51G19	512	192	48	QFN			4/2	2/1	N			1		10							37	16/4			121/22						2
SAMD51G18	256	128																															
SAME51N20	1024	256	100	TQFP	Y	8	8/2	2/3	Y			2		14							81	16/12			256/32						5
SAME51N19	512	192																															
SAME51J20	1024	256	64	TQFP, QFN		6	6/2					1		10							51	16/8									3

Incrementation for the source address of a block transfer is enabled by writing the Source Address Incrementation Enable bit in the Block Transfer Control register (BTCTRL.SRCINC=1). The step size of the incrementation is configurable and can be chosen by writing the Step Selection bit in the Block Transfer Control register (BTCTRL.STEPSEL=1) and writing the desired step size in the Address Increment Step Size bit group in the Block Transfer Control register (BTCTRL.STEPSIZE). If BTCTRL.STEPSEL=0, the step size for the source incrementation will be the size of one beat.

When source address incrementation is configured (BTCTRL.SRCINC=1), SRCADDR is calculated as follows:

If BTCTRL.STEPSEL=1:

$$\mathrm{SRCADDR} = \mathrm{SRCADDR}_{START} + BTCNT \cdot \left(BEATSIZE + 1\right) \cdot 2^{\mathrm{STEPSIZE}}$$

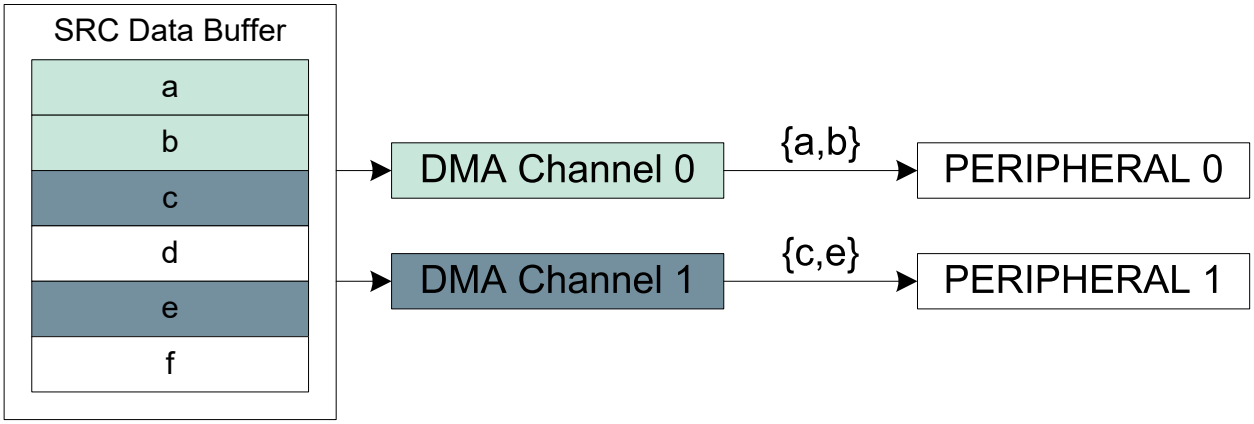
If BTCTRL.STEPSEL=0:

$$\mathrm{SRCADDR} = \mathrm{SRCADDR}_{START} + BTCNT \cdot \left(BEATSIZE + 1\right)$$

- $\mathrm{SRCADDR}_{\mathrm{START}}$ is the source address of the first beat transfer in the block transfer
- BTCNT is the initial number of beats remaining in the block transfer
- BEATSIZE is the configured number of bytes in a beat
- STEPSIZE is the configured number of beats for each incrementation

The following figure shows an example where DMA channel 0 is configured to increment the source address by one beat after each beat transfer (BTCTRL.SRCINC=1), and DMA channel 1 is configured to increment the source address by two beats (BTCTRL.SRCINC=1, BTCTRL.STEPSEL=1, and BTCTRL.STEPSIZE=0x1). As the destination address for both channels are peripherals, destination incrementation is disabled (BTCTRL.DSTINC=0).

Figure 22-8. Source Address Increment



Incrementation for the destination address of a block transfer is enabled by setting the Destination Address Incrementation Enable bit in the Block Transfer Control register (BTCTRL.DSTINC=1). The step size of the incrementation is configurable by clearing BTCTRL.STEPSEL=0 and writing BTCTRL.STEPSIZE to the desired step size. If BTCTRL.STEPSEL=1, the step size for the destination incrementation will be the size of one beat.

When the destination address incrementation is configured (BTCTRL.DSTINC=1), DSTADDR must be set and calculated as follows:

24.9.50 GMAC Greater Than 1518 Byte Frames Transmitted Register

Name: GTBFT1518
Offset: 0x130
Reset: 0x00000000
Property: Read-only

Bit	31	30	29	28	27	26	25	24
	NFTX[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	NFTX[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NFTX[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	NFTX[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – NFTX[31:0] Greater than 1518 Byte Frames Transmitted without Error
This register counts the number of 1518 or above byte frames successfully transmitted without error i.e., no underrun and not too many retries.

Bit 4 – RAMPPPDSU Interrupt Flag for RAMPPPDSU:
This flag is set when an access error is detected by the RAMPPPDSU AHB slave, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' has no effect.

Writing a '1' to this bit will clear the RAMPPPDSU interrupt flag.

Bit 3 – RAMCM4S Interrupt Flag for RAMCM4S
This flag is set when an access error is detected by the RAMCM4S AHB slave, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' has no effect.

Writing a '1' to this bit will clear the RAMCM4S interrupt flag.

Bit 2 – NVMCTRL2 Interrupt Flag for NVMCTRL2
This flag is set when an access error is detected by the NVMCTRL2 AHB slave, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' has no effect.

Writing a '1' to this bit will clear the NVMCTRL2 interrupt flag.

Bit 1 – NVMCTRL1 Interrupt Flag for NVMCTRL1
This flag is set when an access error is detected by the NVMCTRL1 AHB slave, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' has no effect.

Writing a '1' to this bit will clear the NVMCTRL1 interrupt flag.

Bit 0 – NVMCTRL0 Interrupt Flag for NVMCTRL0
This flag is set when an access error is detected by the NVMCTRL0 AHB slave, and will generate an interrupt request if INTENCLR/SET.ERR is '1'.

Writing a '0' has no effect.

Writing a '1' to this bit will clear the NVMCTRL0 interrupt flag.

© 2018 Microchip Technology Inc.

Bit 3 – RSTC RSTC APB Protect Enable

Value	Description
0	RSTC is not write protected
1	RSTC is write protected

Bit 2 – MCLK MCLK APB Protect Enable

Value	Description
0	MCLK is not write protected
1	MCLK is write protected

Bit 1 – PM PM APB Protect Enable

Value	Description
0	PM is not write protected
1	PM is write protected

Bit 0 – PAC PAC APB Protect Enable

Value	Description
0	PAC is not write protected
1	PAC is write protected

Table 34-3. Asynchronous Receiver Error for 16-fold Oversampling

D (Data bits+Parity)	R_{SLOW} [%]	R_{FAST} [%]	Max. total error [%]	Recommended max. Rx error [%]
5	94.12	107.69	+5.88/-7.69	±2.5
6	94.92	106.67	+5.08/-6.67	±2.0
7	95.52	105.88	+4.48/-5.88	±2.0
8	96.00	105.26	+4.00/-5.26	±2.0
9	96.39	104.76	+3.61/-4.76	±1.5
10	96.70	104.35	+3.30/-4.35	±1.5

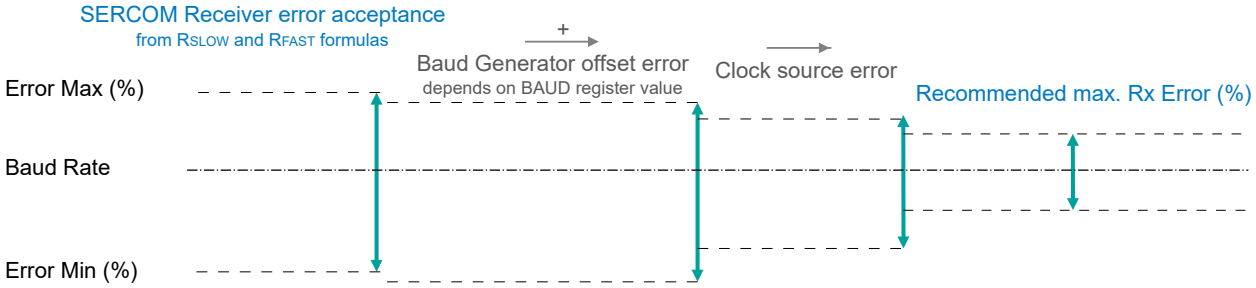
The following equations calculate the ratio of the incoming data rate and internal receiver baud rate:

$$R_{\text{SLOW}} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \quad , \quad R_{\text{FAST}} = \frac{(D+2)S}{(D+1)S+S_M}$$

- R_{SLOW} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate
- R_{FAST} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate
- D is the sum of character size and parity size (D = 5 to 10 bits)
- S is the number of samples per bit (S = 16, 8 or 3)
- S_F is the first sample number used for majority voting (S_F = 7, 3, or 2) when CTRLA.SAMPA=0.
- S_M is the middle sample number used for majority voting (S_M = 8, 4, or 2) when CTRLA.SAMPA=0.

The recommended maximum Rx Error assumes that the receiver and transmitter equally divide the maximum total error. Its connection to the SERCOM Receiver error acceptance is depicted in this figure:

Figure 34-5. USART Rx Error Calculation



The recommendation values in the table above accommodate errors of the clock source and the baud generator. The following figure gives an example for a baud rate of 3Mbps:

This bit is cleared when the corresponding interrupt flag is cleared and the next operation is given.

Writing '0' to this bit has no effect.

Writing '1' to this bit has no effect.

This bit is not write-synchronized.

Bit 6 – LOWTOUT SCL Low Time-Out
This bit is set if an SCL low time-out occurs.

Writing '1' to this bit location will clear this bit. This flag is automatically cleared when writing to the ADDR register.

Writing '0' to this bit has no effect.

This bit is not write-synchronized.

Bits 5:4 – BUSSTATE[1:0] Bus State
These bits indicate the current I^2C bus state.

When in UNKNOWN state, writing 0x1 to BUSSTATE forces the bus state into the IDLE state. The bus state cannot be forced into any other state.

Writing BUSSTATE to idle will set SYNCBUSY.SYSOP.

Value	Name	Description
0x0	UNKNOWN	The bus state is unknown to the I^2C master and will wait for a stop condition to be detected or wait to be forced into an idle state by software
0x1	IDLE	The bus state is waiting for a transaction to be initialized
0x2	OWNER	The I^2C master is the current owner of the bus
0x3	BUSY	Some other I^2C master owns the bus

Bit 2 – RXNACK Received Not Acknowledge
This bit indicates whether the last address or data packet sent was acknowledged or not.

Writing '0' to this bit has no effect.

Writing '1' to this bit has no effect.

This bit is not write-synchronized.

Value	Description
0	Slave responded with ACK.
1	Slave responded with NACK.

Bit 1 – ARBLOST Arbitration Lost
This bit is set if arbitration is lost while transmitting a high data bit or a NACK bit, or while issuing a start or repeated start condition on the bus. The Master on Bus interrupt flag (INTFLAG.MB) will be set when STATUS.ARBLOST is set.

Writing the ADDR.ADDR register will automatically clear STATUS.ARBLOST.

Writing '0' to this bit has no effect.

Writing '1' to this bit will clear it.

This bit is not write-synchronized.

38.8.3.1 Device Endpoint Configuration register n

Name: EPCFGn
Offset: 0x100 + (n x 0x20)
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
		EPTYPE1[2:0]				EPTYPE0[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bits 6:4 – EPTYPE1[2:0] Endpoint Type for IN direction
These bits contains the endpoint type for IN direction.

Upon receiving a USB reset EPCFGn.EPTYPE1 is cleared except for endpoint 0 which is unchanged.

Value	Description
0x0	Bank1 is disabled.
0x1	Bank1 is enabled and configured as Control IN.
0x2	Bank1 is enabled and configured as Isochronous IN.
0x3	Bank1 is enabled and configured as Bulk IN.
0x4	Bank1 is enabled and configured as Interrupt IN.
0x5	Bank1 is enabled and configured as Dual-Bank OUT (Endpoint type is the same as the one defined in EPTYPE0)
0x6-0x7	Reserved

Bits 2:0 – EPTYPE0[2:0] Endpoint Type for OUT direction
These bits contains the endpoint type for OUT direction.

Upon receiving a USB reset EPCFGn.EPTYPE0 is cleared except for endpoint 0 which is unchanged.

Value	Description
0x0	Bank0 is disabled.
0x1	Bank0 is enabled and configured as Control SETUP / Control OUT.
0x2	Bank0 is enabled and configured as Isochronous OUT.
0x3	Bank0 is enabled and configured as Bulk OUT.
0x4	Bank0 is enabled and configured as Interrupt OUT.
0x5	Bank0 is enabled and configured as Dual Bank IN (Endpoint type is the same as the one defined in EPTYPE1)
0x6-0x7	Reserved

Table 39-6. Possible Configurations for Frame Transmission

CCCR		Tx Buffer Element		Frame Transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

39.6.6.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR.TXP. If the bit is set, the CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

39.6.6.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

Value	Description
0	Work
1	Reset

41.6 Functional Description

41.6.1 Principle of Operation

Configurable Custom Logic (CCL) is a programmable logic block that can use the device port pins, internal peripherals, and the internal Event System as both input and output channels. The CCL can serve as glue logic between the device and external devices. The CCL can eliminate the need for external logic component and can also help the designer overcome challenging real-time constrains by combining core independent peripherals in clever ways to handle the most time critical parts of the application independent of the CPU.

41.6.2 Operation

41.6.2.1 Initialization

The following bits are enable-protected, meaning that they can only be written when the corresponding even LUT is disabled (LUTCTRLx.ENABLE=0):

- Sequential Selection bits in the Sequential Control x (SEQCTRLx.SEQSEL) register

The following registers are enable-protected, meaning that they can only be written when the corresponding LUT is disabled (LUTCTRLx.ENABLE=0):

- LUT Control x (LUTCTRLx) register, except the ENABLE bit

Enable-protected bits in the LUTCTRLx registers can be written at the same time as LUTCTRLx.ENABLE is written to '1', but not at the same time as LUTCTRLx.ENABLE is written to '0'.

Enable-protection is denoted by the Enable-Protected property in the register description.

41.6.2.2 Enabling, Disabling, and Resetting

The CCL is enabled by writing a '1' to the Enable bit in the Control register (CTRL.ENABLE). The CCL is disabled by writing a '0' to CTRL.ENABLE.

Each LUT is enabled by writing a '1' to the Enable bit in the LUT Control x register (LUTCTRLx.ENABLE). Each LUT is disabled by writing a '0' to LUTCTRLx.ENABLE.

The CCL is reset by writing a '1' to the Software Reset bit in the Control register (CTRL.SWRST). All registers in the CCL will be reset to their initial state, and the CCL will be disabled. Refer to 41.8.1 CTRL for details.

41.6.2.3 Lookup Table Logic

The lookup table in each LUT unit can generate any logic expression OUT as a function of three inputs (IN[2:0]), as shown in Figure 41-2. One or more inputs can be masked. The truth table for the expression is defined by TRUTH bits in LUT Control x register (LUTCTRLx.TRUTH).

Offset	Name	Bit Pos.								
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
0x2C ... 0x37	Reserved									
0x38	DATA	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							
3C	INTVECTV0	7:0	INTVECTV[7:0]							
		15:8	INTVECTV[15:8]							
		23:16	INTVECTV[23:16]							
		31:24	INTVECTV[31:24]							
40	INTVECTV1	7:0	INTVECTV[7:0]							
		15:8	INTVECTV[15:8]							
		23:16	INTVECTV[23:16]							
		31:24	INTVECTV[31:24]							
44	INTVECTV2	7:0	INTVECTV[7:0]							
		15:8	INTVECTV[15:8]							
		23:16	INTVECTV[23:16]							
		31:24	INTVECTV[31:24]							
48	INTVECTV3	7:0	INTVECTV[7:0]							
		15:8	INTVECTV[15:8]							
		23:16	INTVECTV[23:16]							
		31:24	INTVECTV[31:24]							
0x4C ... 0x5B	Reserved									
0x5C	HASHKEY0	7:0	HASHKEY[7:0]							
		15:8	HASHKEY[15:8]							
		23:16	HASHKEY[23:16]							
		31:24	HASHKEY[31:24]							
0x60	HASHKEY1	7:0	HASHKEY[7:0]							
		15:8	HASHKEY[15:8]							
		23:16	HASHKEY[23:16]							
		31:24	HASHKEY[31:24]							
0x64	HASHKEY2	7:0	HASHKEY[7:0]							
		15:8	HASHKEY[15:8]							
		23:16	HASHKEY[23:16]							
		31:24	HASHKEY[31:24]							
0x68	HASHKEY3	7:0	HASHKEY[7:0]							
		15:8	HASHKEY[15:8]							
		23:16	HASHKEY[23:16]							
		31:24	HASHKEY[31:24]							
0x6C	GHASH0	7:0	GHASH[7:0]							
		15:8	GHASH[15:8]							

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	Resulting point
nu1RandomBase	nu1	I	Crypto RAM	u2ModLength + 4	Random	Corrupted
nu1Workspace	nu1	I	Crypto RAM	3*u2ModLength + 28	–	Workspace

43.3.7.7.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields should be filled
PUKCL _GF2NEcRandomiseCoordinate(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcRandomiseCoordinate(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcRandomiseCoordinate(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_GF2NEcRandomiseCoordinate(nu1RandomBase) = <Base of the ram location where the the rng
is stored>;
PUKCL _GF2NEcRandomiseCoordinate(nu1PointBase) = <Base of the ram location of the point>;
PUKCL _GF2NEcRandomiseCoordinate(nu1Workspace) =
<Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcRandomiseCoordinate,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

43.3.7.7.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1RandomBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4} and {nu1Workspace, 3*u2ModLength + 28}

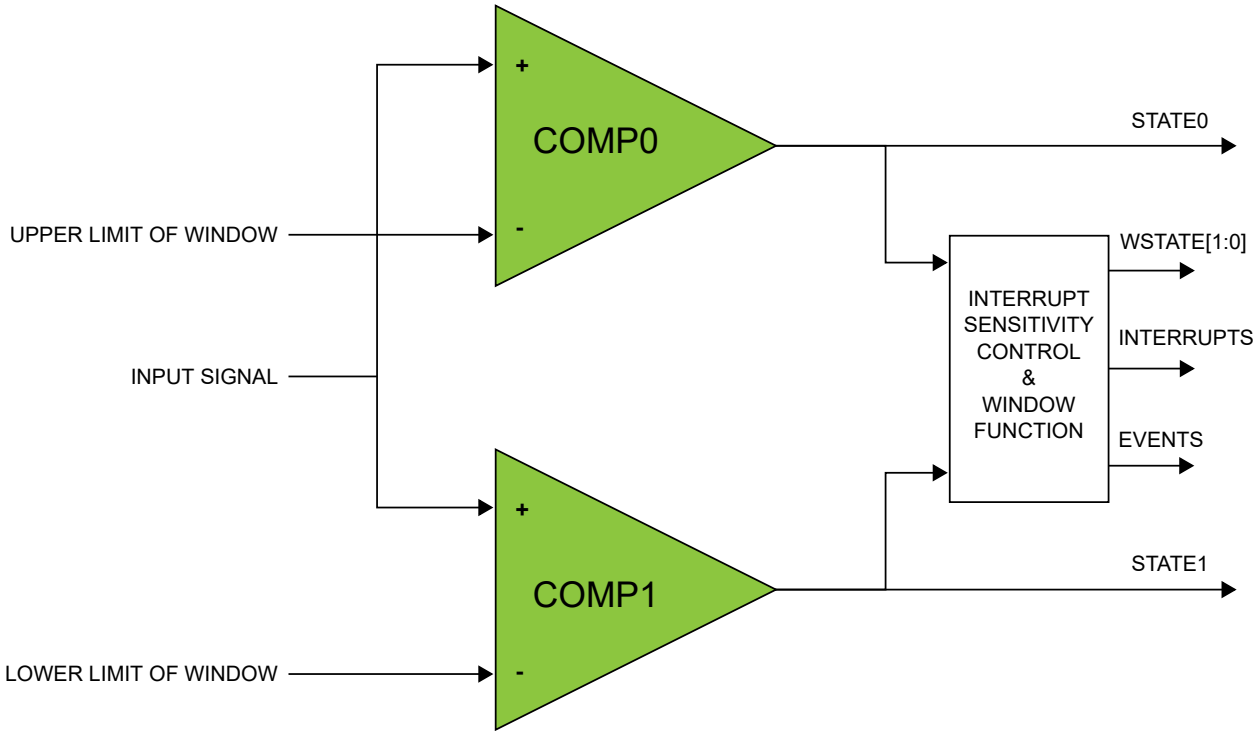
43.3.7.7.7 Status Returned Values

Table 43-105. GF2NEcRandomiseCoordinate Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

When the comparators are configured for window mode and single-shot mode, measurements are performed simultaneously on both comparators. Writing '1' to either Start Comparison bit in the Control B register (CTRLB.STARTx) will start a measurement. Likewise either peripheral event can start a measurement.

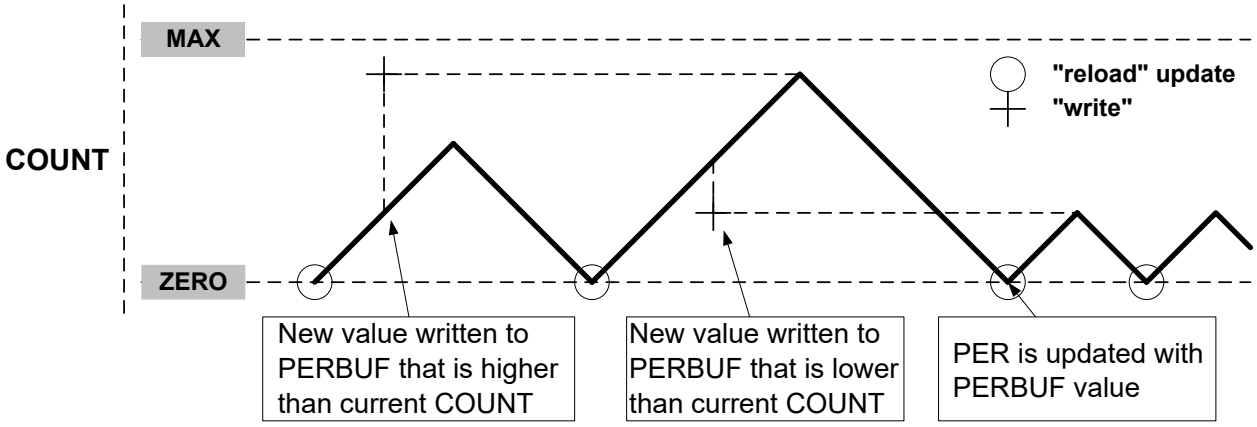
Figure 46-4. Comparators in Window Mode



46.6.5 VDD Scaler

The VDD scaler generates a reference voltage that is a fraction of the device's supply voltage, with 64 levels. One independent voltage channel is dedicated for each comparator. The scaler of a comparator is enabled when the Negative Input Mux bit field in the respective Comparator Control register (COMPCTRLx.MUXNEG) is set to 0x5 and the comparator is enabled. The voltage of each channel is selected by the Value bit field in the Scaler x registers (SCALERx.VALUE).

Figure 49-13. Changing the Period Using Buffering



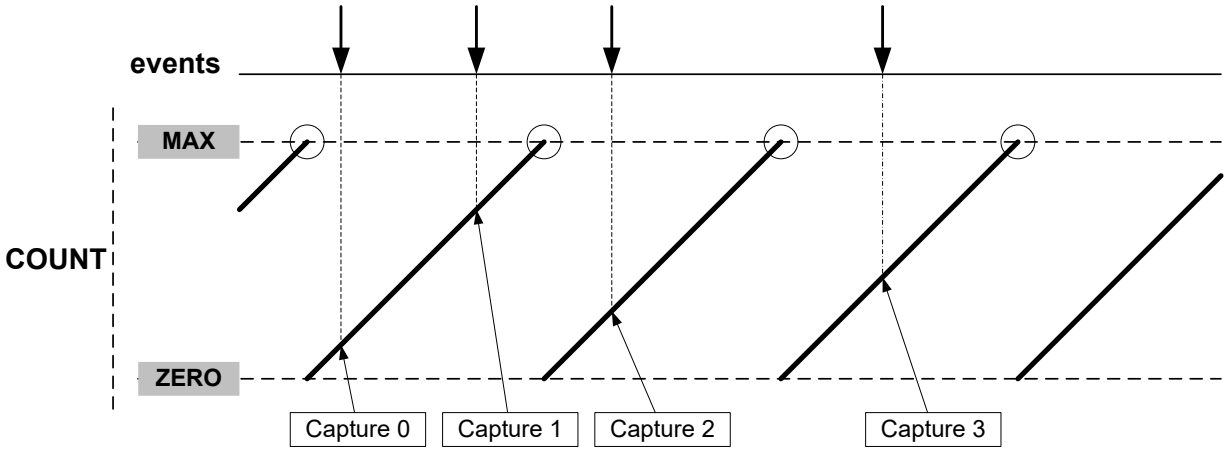
49.6.2.7 Capture Operations

To enable and use capture operations, the Match or Capture Channel x Event Input Enable bit in the Event Control register (EVCTRL.MCEIx) must be written to '1'. The capture channels to be used must also be enabled in the Capture Channel x Enable bit in the Control A register (CTRLA.CPTENx) before capturing can be performed.

Event Capture Action

The compare/capture channels can be used as input capture channels to capture events from the Event System, and give them a timestamp. The following figure shows four capture events for one capture channel.

Figure 49-14. Input Capture Timing



For input capture, the buffer register and the corresponding CCx act like a FIFO. When CCx is empty or read, any content in CCBUFx is transferred to CCx. The buffer valid flag is passed to set the CCx interrupt flag (IF) and generate the optional interrupt, event or DMA request. CCBUFx register value can't be read, all captured data must be read from CCx register.

49.8.12 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x2C
Reset: 0x00000000
Property: -

Bit	23	22	21	20	19	18	17	16
			MCx	MCx	MCx	MCx	MCx	MCx
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FAULTx	FAULTx	FAULTB	FAULTA	DFS			
Access	R/W	R/W	R/W	R/W	R/W			
Reset	0	0	0	0	0			

Bit	7	6	5	4	3	2	1	0
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 21,20,19,18,17,16 – MCx Match or Capture Channel x Interrupt Flag
This flag is set on the next CLK_TCC_COUNT cycle after a match with the compare condition or once CCx register contain a valid capture value.

Writing a '0' to one of these bits has no effect.

Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag

In Capture operation, this flag is automatically cleared when CCx register is read.

Bits 15,14 – FAULTx Non-Recoverable Fault x Interrupt Flag
This flag is set on the next CLK_TCC_COUNT cycle after a Non-Recoverable Fault x occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Non-Recoverable Fault x interrupt flag.

Bit 13 – FAULTB Recoverable Fault B Interrupt Flag
This flag is set on the next CLK_TCC_COUNT cycle after a Recoverable Fault B occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Recoverable Fault B interrupt flag.

Bit 12 – FAULTA Recoverable Fault A Interrupt Flag
This flag is set on the next CLK_TCC_COUNT cycle after a Recoverable Fault B occurs.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the Recoverable Fault B interrupt flag.

Figure 52-9. PCC Waveforms (DSIZE=4_DATA, ALWYS = 0, HALFS = 1, FRSTS = 0)

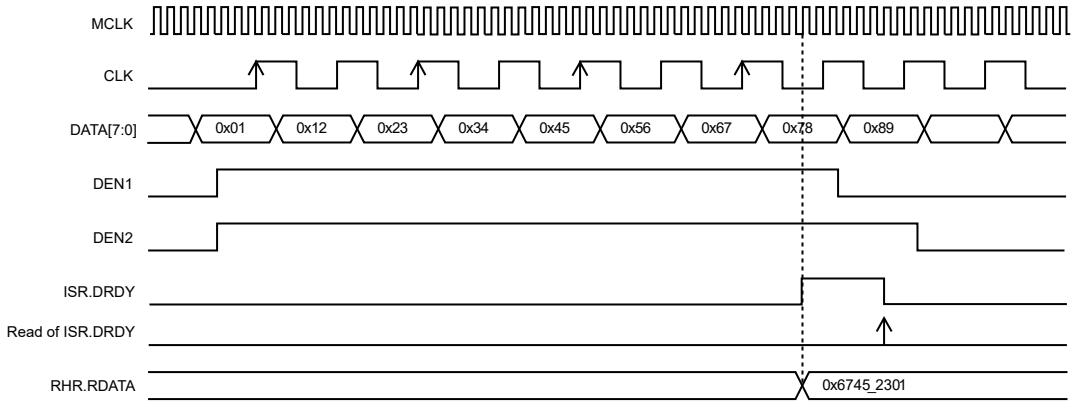


Figure 52-10. PCC Waveforms (ISIZE=10_BITS, DSIZE=2_DATA, ALWYS = 0, HALFS = 1, FRSTS = 0, SCALE = 0)

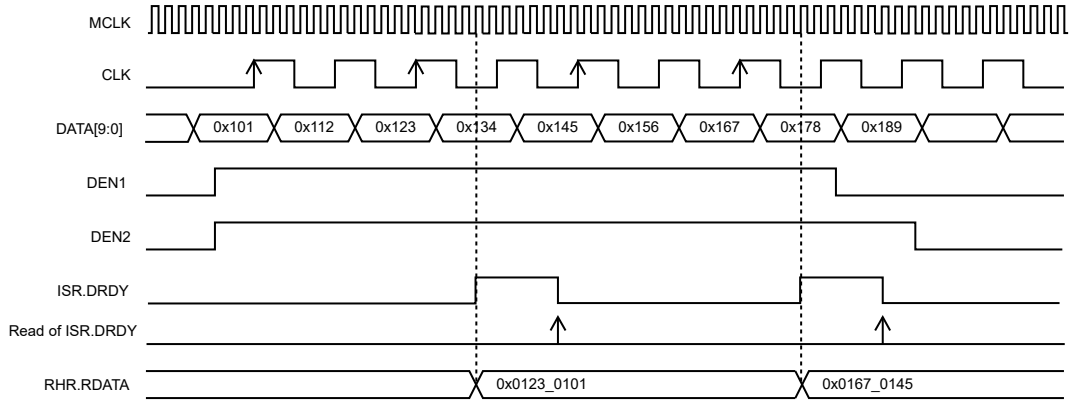
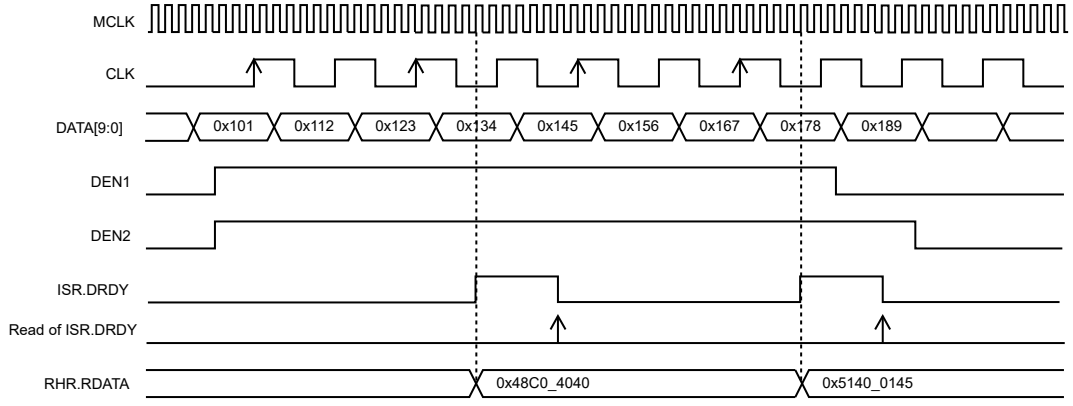


Figure 52-11. PCC Waveforms (ISIZE=10_BITS, DSIZE=2_DATA, ALWYS = 0, HALFS = 1, FRSTS = 0, SCALE = 1)



53.8.17 Channel x Compare Buffer Value

Name: CCBUFx
Offset: 0x30 + x*0x04 [x=0..1]
Reset: 0x00000000
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

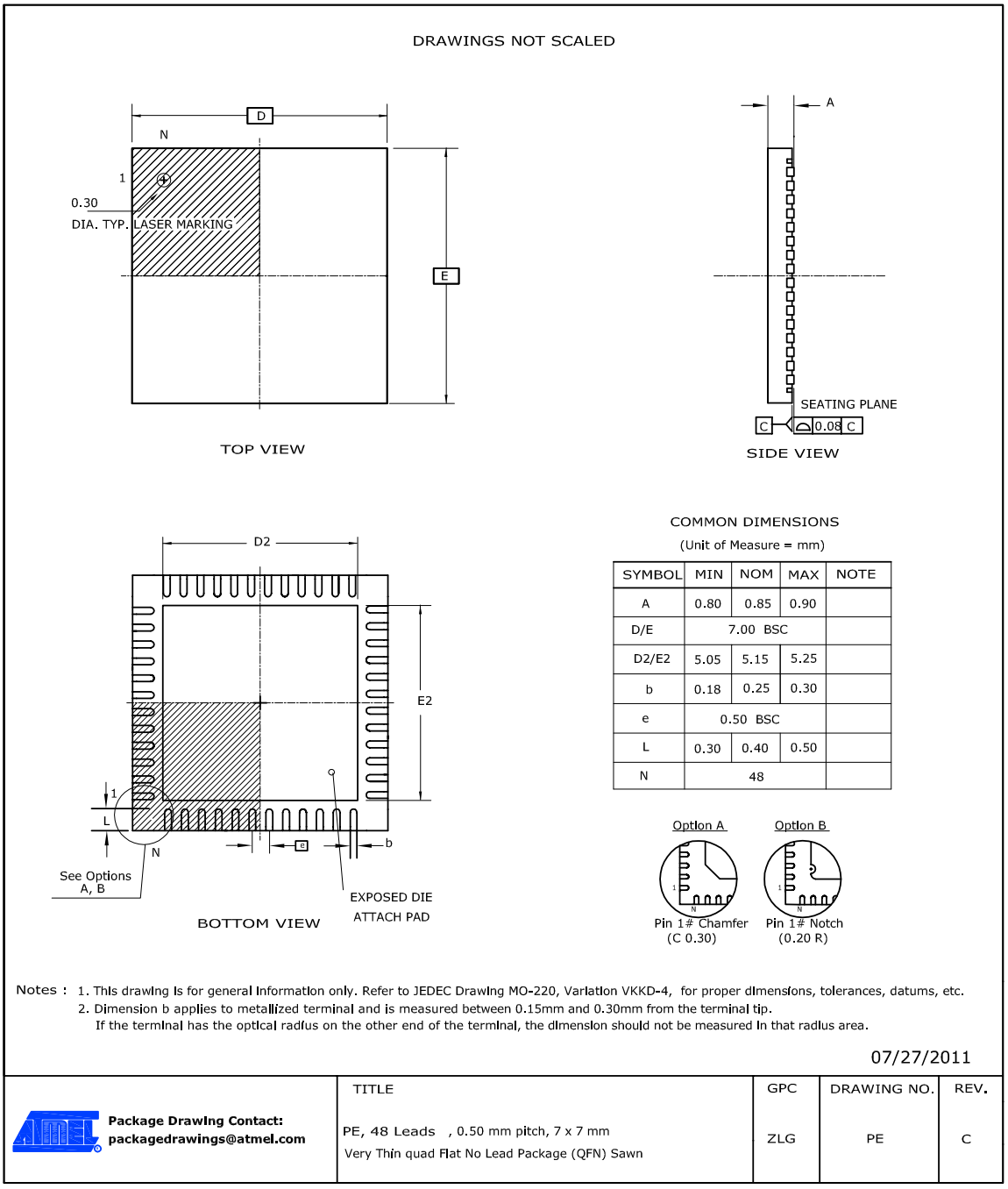
Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CCBUF[15:8]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CCBUF[7:0]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CCBUF[15:0] Channel Compare Buffer Value
These bits hold the value of the channel x compare buffer register. The register is used as buffer for the associated compare register (CCx). Accessing this register using the CPU will affect the corresponding CCBVx status bit (STATUS.CCBUFVx).

55.3.1 48 pin QFN



SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.85	0.90	
D/E		7.00 BSC		
D2/E2	5.05	5.15	5.25	
b	0.18	0.25	0.30	
e		0.50 BSC		
L	0.30	0.40	0.50	
N		48		

Note: The exposed die attach pad is not connected electrically inside the device.

Table 55-2. Device and Package Maximum Weight

140	mg

Table 55-3. Package Characteristics

Moisture Sensitivity Level	MSL3

56.10 SDHC Interface

The SD/MMC Host Controller (SDHC) is compliant with the SD Host Controller Standard specifications. There are two instances of SDHC available on this device: SDHC0 and SDHC1. The typical connection diagram is shown in the following figure.