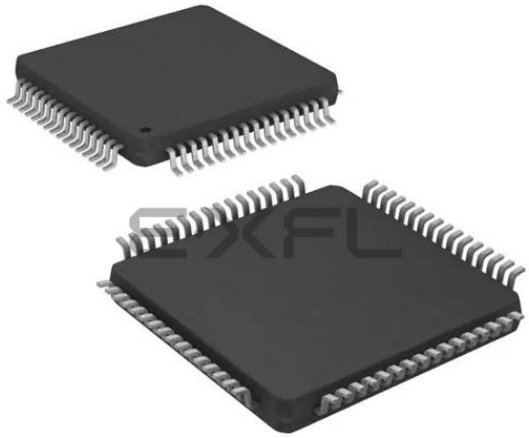


Welcome to E-XFL.COM

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	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	192K 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-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsame51j19a-aut

minimum packet buffer memory size should be chosen to satisfy the maximum frame to be transmitted in the application.

In full store and forward mode, once the complete transmit frame is written into the packet buffer memory, a trigger is sent across to the MAC transmitter, which will then begin reading the frame from the packet buffer memory. Since the whole frame is present and stable in the packet buffer memory an underflow of the transmitter is not possible. The frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be retransmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In Partial Store and Forward mode, a trigger is sent across to the MAC transmitter as soon as sufficient packet data is available, which will then begin fetching the frame from the packet buffer memory. If, after this point, the MAC transmitter is able to fetch data from the packet buffer faster than the AHB DMA can fill it, an underflow of the transmitter is possible. In this case, the transmission is terminated early, and the packet buffer is completely flushed. Transmission can only be restarted by writing a '1' to the Transmit Start bit in the Network Control register (NCR.TSTART).

In half duplex mode, the frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be retransmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In full duplex mode, the frame is removed from the packet buffer on the fly.

Other than underflow, the only MAC related errors that can occur are due to collisions during half duplex transmissions. When a collision occurs the frame still exists in the packet buffer memory so can be retried directly from there. After sixteen failed transmit attempts, the frame will be flushed from the packet buffer.

24.6.3.8 Receive Packet Buffer

The receive packet buffer stores frames from the MAC receiver along with their status and statistics. Frames with errors are flushed from the packet buffer memory, while good frames are pushed onto the DMA AHB interface.

The receiver packet buffer monitors the FIFO write interface from the MAC receiver and translates the FIFO pushes into packet buffer writes. At the end of the received frame the status and statistics are buffered so that the information can be used when the frame is read out. When programmed in full store and forward mode and the frame has an error, the frame data is immediately flushed from the packet buffer memory allowing subsequent frames to utilize the freed up space. The status and statistics for bad frames are still used to update the GMAC registers.

To accommodate the status and statistics associated with each frame, three words per packet (or two if configured in 64-bit datapath mode) are reserved at the end of the packet data. If the packet is bad and requires to be dropped, the status and statistics are the only information held on that packet.

The receiver packet buffer will also detect a full condition so that an overflow condition can be detected. If this occurs, subsequent packets are dropped and an RX overflow interrupt is raised.

For full store and forward, the DMA only begins packet fetches once the status and statistics for a frame are available. If the frame has a bad status due to a frame error, the status and statistics are passed on to the GMAC registers. If the frame has a good status, the information is used to read the frame from the packet buffer memory and burst onto the AHB using the DMA buffer management protocol. Once the last frame data has been transferred to the packet buffer, the status and statistics are updated to the GMAC registers.

128 to 255 Byte Frames Transmitted Register	1024 to 1518 Byte Frames Received Register
256 to 511 Byte Frames Transmitted Register	1519 to Maximum Byte Frames Received Register
512 to 1023 Byte Frames Transmitted Register	Undersize Frames Received Register
1024 to 1518 Byte Frames Transmitted Register	Oversize Frames Received Register
Greater Than 1518 Byte Frames Transmitted Register	Jabbers Received Register
Transmit Underruns Register	Frame Check Sequence Errors Register
Single Collision Frames Register	Length Field Frame Errors Register
Multiple Collision Frames Register	Receive Symbol Errors Register
Excessive Collisions Register	Alignment Errors Register
Late Collisions Register	Receive Resource Errors Register
Deferred Transmission Frames Register	Receive Overrun Register
Carrier Sense Errors Register	IP Header Checksum Errors Register
Octets Received Low Register	TCP Checksum Errors Register
Octets Received High Register	UDP Checksum Errors Register
Frames Received Register	

These registers reset to zero on a read and stick at all ones when they count to their maximum value. They should be read frequently enough to prevent loss of data.

The receive statistics registers are only incremented when the receive enable bit (RXEN) is set in the Network Control register.

Once a statistics register has been read, it is automatically cleared. When reading the Octets Transmitted and Octets Received registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

24.9.100 Received LPI Time

Name: RLPITI
Offset: 0x274
Reset: 0x00000000
Property: Read-Only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	RLPITI[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
	RLPITI[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
	RLPITI[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – RLPITI[23:0] Received LPI Time
The value of this bit field increments once every 16 AHB clock cycles when the Low Power Idle Enable bit in the Network Configuration Register (NCR.LPI) is written to '1'.

Cleared on read.

Value	Name	Description
0x11	LR	Lock Region - Locks the region containing the address location in the ADDR register until next reset.
0x12	UR	Unlock Region - Unlocks the region containing the address location in the ADDR register until next reset.
0x13	SPRM	Sets the power reduction mode.
0x14	CPRM	Clears the power reduction mode.
0x15	PBC	Page Buffer Clear - Clears the page buffer.
0x16	SSB	Set Security Bit
0x17	BKSWRST	Bank swap and system reset, if SmartEEPROM is used also reallocate its data into the opposite BANK
0x18	CELCK	Chip Erase Lock - DSU.CTRL.CE command is not available
0x19	CEULCK	Chip Erase Unlock - DSU.CTRL.CE command is available
0x1A	SBPDIS	Sets STATUS.BPDIS, Boot loader protection is discarded until CBPDIS is issued or next start-up sequence
0x1B	CBPDIS	Clears STATUS.BPDIS, Boot loader protection is not discarded
0x1C-0x2F		Reserved
0x30	ASEES0	Configure SmartEEPROM to use Sector 0
0x31	ASEES1	Configure SmartEEPROM to use Sector 1
0x32	SEERALOC	Starts SmartEEPROM sector reallocation algorithm
0x33	SEEFLUSH	Flush SmartEEPROM data when in buffered mode
0x34	LSEE	Lock access to SmartEEPROM data from any means
0x35	USEE	Unlock access to SmartEEPROM data
0x36	LSEER	Lock access to the SmartEEPROM Register Address Space (above 64KB)
0x37	USEER	Unock access to the SmartEEPROM Register Address Space (above 64KB)
0x38-0x7F		Reserved

32.9.2 Data Direction Clear

Name: DIRCLR
Offset: 0x04
Reset: 0x00000000
Property: PAC Write-Protection

This register allows the user to set one or more I/O pins as an input, without doing a read-modify-write operation. Changes in this register will also be reflected in the Data Direction (DIR), Data Direction Toggle (DIRTGL) and Data Direction Set (DIRSET) registers.

Tip: The I/O pins are assembled in pin groups (”PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

Bit	31	30	29	28	27	26	25	24
	DIRCLR[31:24]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DIRCLR[23:16]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIRCLR[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
	DIRCLR[7:0]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DIRCLR[31:0] Port Data Direction Clear
Writing a '0' to a bit has no effect.

Writing a '1' to a bit will clear the corresponding bit in the DIR register, which configures the I/O pin as an input.

Value	Description
0	The corresponding I/O pin in the PORT group will keep its configuration.
1	The corresponding I/O pin in the PORT group is configured as input.

32.9.6 Data Output Value Clear

Name: OUTCLR
Offset: 0x14
Reset: 0x00000000
Property: PAC Write-Protection

This register allows the user to set one or more output I/O pin drive levels low, without doing a read-modify-write operation. Changes in this register will also be reflected in the Data Output Value (OUT), Data Output Value Toggle (OUTTGL) and Data Output Value Set (OUTSET) registers.

Tip: The I/O pins are assembled in pin groups (”PORT groups”) with up to 32 pins. Group 0 consists of the PA pins, group 1 is for the PB pins, etc. Each pin group has its own PORT registers, with a 0x80 address spacing. For example, the register address offset for the Data Direction (DIR) register for group 0 (PA00 to PA31) is 0x00, and the register address offset for the DIR register for group 1 (PB00 to PB31) is 0x80.

Bit	31	30	29	28	27	26	25	24
	OUTCLR[31:24]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OUTCLR[23:16]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	OUTCLR[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
	OUTCLR[7:0]							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – OUTCLR[31:0] PORT Data Output Value Clear
Writing '0' to a bit has no effect.

Writing '1' to a bit will clear the corresponding bit in the OUT register. Pins configured as outputs via the Data Direction register (DIR) will be set to low output drive level. Pins configured as inputs via DIR and with pull enabled via the Pull Enable bit in the Pin Configuration register (PINCFG.PULLEN) will set the input pull direction to an internal pull-down.

Protocol T=0

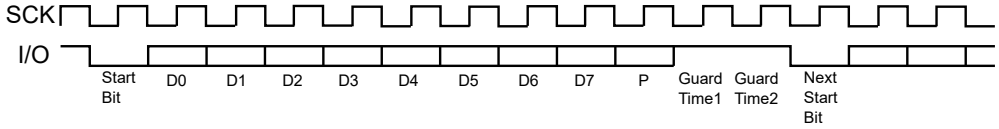
In T=0 protocol, a character is made up of:

- one start bit,
- eight data bits,
- one parity bit
- and one guard time, which lasts two bit times.

The transfer is synchronous (CTRLA.CMODE=1). The transmitter shifts out the bits and does not drive the I/O line during the guard time. Additional guard time can be added by programming the Guard Time (CTRLC.GTIME).

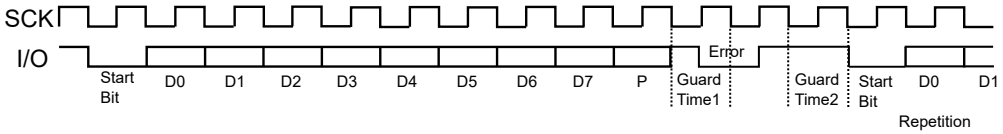
If no parity error is detected, the I/O line remains during the guard time and the transmitter can continue with the transmission of the next character, as shown in the figure below.

Figure 34-18. T=0 Protocol without Parity Error



If a parity error is detected by the receiver, it drives the I/O line to 0 during the guard time, as shown in the next figure. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time, which lasts 1 bit time.

Figure 34-19. T=0 Protocol with Parity Error



When the USART is the receiver and it detects a parity error, the parity error bit in the Status Register (STATUS.PERR) is set and the character is not written to the receive FIFO.

Receive Error Counter

The receiver also records the total number of errors (receiver parity errors and NACKs from the remote transmitter) up to a maximum of 255. This can be read in the Receive Error Count (RXERRCNT) register. RXERRCNT is automatically cleared on read.

Receive NACK Inhibit

The receiver can also be configured to inhibit error generation. This can be achieved by setting the Inhibit Not Acknowledge (CTRLC.INACK) bit. If CTRLC.INACK is 1, no error signal is driven on the I/O line even if a parity error is detected. Moreover, if CTRLC.INACK is set, the erroneous received character is stored in the receive FIFO, and the STATUS.PERR bit is set. Inhibit not acknowledge (CTRLC.INACK) takes priority over disable successive receive NACK (CTRLC.DSNACK).

Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next character. Repetition is enabled by writing the Maximum Iterations register (CTRLC.MAXITER) to a non-zero value. The USART repeats the character the number of times specified in CTRLC.MAXITER.

36. SERCOM I²C – Inter-Integrated Circuit

36.1 Overview

The inter-integrated circuit (I²C) interface is one of the available modes in the serial communication interface (SERCOM).

The I²C interface uses the SERCOM transmitter and receiver configured as shown in Figure 36-1. Labels in capital letters are registers accessible by the CPU, while lowercase labels are internal to the SERCOM.

A SERCOM instance can be configured to be either an I²C master or an I²C slave. Both master and slave have an interface containing a shift register, a transmit buffer and a receive buffer. In addition, the I²C master uses the SERCOM baud-rate generator, while the I²C slave uses the SERCOM address match logic.

Related Links

33. SERCOM – Serial Communication Interface

36.2 Features

SERCOM I²C includes the following features:

- Master or slave operation
- Can be used with DMA
- Philips I²C compatible
- SMBus™ compatible
- PMBus compatible
- Support of 100kHz and 400kHz, 1MHz and 3.4MHz I²C mode
- 32-bit Data Extension for better system bus utilization
- 4-Wire operation supported
- Physical interface includes:
  - Slew-rate limited outputs
  - Filtered inputs
- Slave operation:
  - Operation in all sleep modes
  - Wake-up on address match
  - 7-bit and 10-bit Address match in hardware for:
  - - Unique address and/or 7-bit general call address
    - Address range
    - Two unique addresses can be used with DMA

Related Links

33.2 Features

Bit	15	14	13	12	11	10	9	8
						CLKGSEL	SDCLKFSEL[9:8]	
Access						R/W	R/W	R/W
Reset						0	0	0

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

Bit 10 – CLKGSEL Clock Generator Select
Refer to CGGSEL in CCR.

Bits 9:0 – SDCLKFSEL[9:0] SDCLK Frequency Select
Refer to SDCLKFSEL in CCR.

- Type 3: Add a random number of clock cycles to data processing, subject to a maximum of 11/13/15 clock cycles for key sizes of 128/192/256 bits
- Type 4: Add random spurious power consumption during data processing

By default, all countermeasures are enabled. One or more of the countermeasures can be disabled by programming the Countermeasure Type field in the Control A (CTRLA.CTYPE) register. The countermeasures use random numbers generated by a deterministic random number generator embedded in AES module. The seed for the random number generator is written to the RANDSEED register. Note also that a new seed must be written after a change in the keysize. Note that enabling countermeasures reduces AES module's throughput. In short, the throughput is highest with all the countermeasures disabled. On the other hand, with all of the countermeasures enabled, the best protection is achieved but the throughput is worst.

42.6.3 Galois Counter Mode (GCM)

GCM is comprised of the AES engine in CTR mode along with a universal hash function (GHASH engine) that is defined over a binary Galois field to produce a message authentication tag. The GHASH engine processes data packets after the AES operation. GCM provides assurance of the confidentiality of data through the AES Counter mode of operation for encryption. Authenticity of the confidential data is assured through the GHASH engine. Refer to the NIST Special Publication 800-38D Recommendation for more complete information.

```
    _PUKCL_SELFTEST   PUKCL_SelfTest;
    _PUKCL_SMULT      PUKCL_Smult;
    _PUKCL_SQUARE     PUKCL_Square;
    _PUKCL_SWAP       PUKCL_Swap;

    // ECC
    _PUKCL_ZPECCADD                    PUKCL_ZpEccAdd;
    _PUKCL_ZPECCDBL                    PUKCL_ZpEccDbl;
    _PUKCL_ZPECCADDSUB                 PUKCL_ZpEccAddSub;
    _PUKCL_ZPECCMUL                    PUKCL_ZpEccMul;
    _PUKCL_ZPECDSAGENERATE             PUKCL_ZpEcDsaGenerate;
    _PUKCL_ZPECDSAVERIFY               PUKCL_ZpEcDsaVerify;
    _PUKCL_ZPECDSAQUICKVERIFY          PUKCL_ZpEcDsaQuickVerify;
    _PUKCL_ZPECCQUICKDUALMUL           PUKCL_ZpEccQuickDualMul;
    _PUKCL_ZPECCONVPROJTOAFFINE        PUKCL_ZpEcConvProjToAffine;
    _PUKCL_ZPECCONVAFFINETOPROJECTIVE PUKCL_ZpEcConvAffineToProjective;
    _PUKCL_ZPECRANDOMIZECOORDINATE     PUKCL_ZpEcRandomiseCoordinate;
    _PUKCL_ZPECPOINTISONCURVE          PUKCL_ZpEcPointIsOnCurve;

    // ECC
    _PUKCL_GF2NECCADD                    PUKCL_GF2NEccAdd;
    _PUKCL_GF2NECCDBL                    PUKCL_GF2NEccDbl;
    _PUKCL_GF2NECCMUL                    PUKCL_GF2NEccMul;
    _PUKCL_GF2NECDSAGENERATE             PUKCL_GF2NEcDsaGenerate;
    _PUKCL_GF2NECDSAVERIFY               PUKCL_GF2NEcDsaVerify;
    _PUKCL_GF2NECCONVPROJTOAFFINE        PUKCL_GF2NEcConvProjToAffine;
    _PUKCL_GF2NECCONVAFFINETOPROJECTIVE PUKCL_GF2NEcConvAffineToProjective;
    _PUKCL_GF2NECRANDOMIZECOORDINATE     PUKCL_GF2NEcRandomiseCoordinate;
    _PUKCL_GF2NECPOINTISONCURVE          PUKCL_GF2NEcPointIsOnCurve;
    } P;
} PUKCL_PARAM,
```

43.3.3.2.1 PUKCL_HEADER Structure

The PUKCL_HEADER is common for all services of the library. This header includes standard fields to indicate the requested service, sub-service, options, return status, and so on, as shown in the following tables.

Different terms used in the below description to be understood, are as follows:

- Parameter – Represents a variable used by the PUKCL. Every parameter belongs to either PUKCL_HEADER or PUKCL Service Specific Header
- Type – Indicates the data type. For details on data type, please refer to `CryptoLib_typedef_pb.h` file in the library
- Dir – Direction. Indicates whether PUKCL considers the variable as input or output. Input means that the application passes data to the PUKCL using the variable. Output means that the PUKCL uses the variable to pass data to the application.
- Location – Suggests whether the parameter need to be stored in Crypto RAM or device SRAM. The PUKCL driver has macros for placing parameters into Crypto RAM, so that the user does not have to worry about the addresses
- Data Length – If a parameter is a pointer variable, the Data Length column shows the size of the data pointed by the pointer

Table 43-1. PUKCL_HEADER Structure

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u1Service	u1	I	–	–	Required service	Executed service
u1SubService	u1	I	–	–	Required sub-service	Executed sub-service
u2Option	u2	I	–	–	Required option	Executed option
Specific	PUKCL_STATUS	I/O	–	–	See Table 43-2	See Table 43-2

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8 or 64 bytes	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	–	–	Length of N	Length of N
nu1XBase	nu1	I	Crypto RAM	u2XLength or u2ModLength + 4 (see **Note 1**)	Base of X	Base of X (see **Note 2**)
u2XLength	u2	I	–	–	Length of X	Length of X
nu1YBase	nu1	I	Crypto RAM	u2YLength	Base of Y	Base of Y
u2YLength	u2	I	–	–	Length of Y	Length of Y
nu1ZBase	nu1	I	Crypto RAM	u2XLength + u2YLength	Base of Z	Base of Z untouched
nu1RBase	nu1	I	Crypto RAM	u2XLength + u2YLength	Base of R	Base of R (see **Note 3**)

Note:

1. In case of a reduction option is specified, if necessary, the area X will be extended to u2ModLength + 4 bytes.
2. If FMult is without reduction, X is untouched. If FMult is with reduction, X is filled with the final result.
3. If FMult is without reduction, R is filled with the final result. If FMult is with reduction, R is corrupted.

43.3.4.9.5 Available Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Full Multiplication operation option described in Table 43-26
- the mandatory CarryOperand option described in Table 43-27 and Table 43-28
- the facultative Modular Reduction option (see 43.3.5.1 Modular Reduction). If the Modular Reduction is not requested, this option is absent.

The u2Options number is calculated by an Inclusive OR of the options.

Some Examples in C language are:

- Operation: Full Multiply only without carry and without Modular Reduction
```
PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_FMULT_ONLY) |
SET_CARRYOPTION(CARRY_NONE);
```
- Operation: Full Multiply with addition with Specific/CarryIn addition and with Fast Modular Reduction
```
PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_FMULT_ADD) |

SET_CARRYOPTION(ADD_CARRY) |

PUKCL_REDMOD_REDUCTION |

PUKCL_REDMOD_USING_FASTRED;
```

- Setup of the Fast or Normalize functions: generation of the reduction constant
- Fast Modular Reduction
- Big Modular Reduction (using Euclide's division)
- Normalization

The service name for this operation is `RedMod`.

43.3.5.1.4 Modular Reduction Setup

This service calculates the constant Cns, computed from the modulus and used to speed up the modular reduction:

Cns = SetupConstant(N)

This service must be processed before the use of the Fast or Normalize functions. In the Setup computations, the following data must be provided:

- N the modulus (pointed by {nu1ModBase,u2ModLength +4}).
- Cns the Setup Constant Result (pointed by {nu1CnsBase,u2ModLength +12}).
- X used as a workspace (pointed by {nu1XBase,2 * u2ModLength + 8}) (include the supplementary bytes; see **Note 2** in Table 43-44
- R used as a workspace (pointed by {nu1RBase,64 or 68bytes}).
- u2ModLength is the Aligned Significant Length of the modulus and is not the byte Significant Length (see 43.3.3.4 Aligned Significant Length).

43.3.5.1.5 Fast Reductions and Normalization

These commands calculate an approximated or exact Modular Reduction, that is, the result may be greater than the modulus, but is always congruent to the true result.

Important: Before using these functions, ensure that the constant Cns has been calculated with the setup for the Modular Reduction service.

Input and Result significant values verify:

- For the Fast Modular Reduction:

$0 \leq X < N^2 \times 2^{32}$

$R = Xmod(N) + k \times N \quad with \quad 0 \leq k \leq 4$

- For the Normalize:

$XLength < (NLength + 4)bytes \;\; R$

$= Xmod(N)$

In these Fast Modular Reduction and Normalize computations, the following data have to be provided:

- X (pointed by {nu1XBase,2 * u2ModLength +8})
  - The Normalize computation accept as entry a value whose length is lower or equal to u2ModLength + 4 (that is, for example, a value yet reduced but not normalized.). The u2ModLength + 4 MSB bytes are cleared at the beginning of the computation.
  - in case of Fast RedMod computations, the value X mayverify: X < (N^2) *(2^{32}).
  - include the supplementary bytes; see **Note 3** in Table 43-45)
- R (pointed by {nu1RBase,u2Modlength +4})

46.5.1 I/O Lines

Using the AC's I/O lines requires the I/O pins to be configured. Refer to *PORT - I/O Pin Controller* for details.

Table 46-1. I/O Lines

Instance	Signal	I/O Line	Peripheral Function
AC0	AIN0	PAxx	A
AC0	AIN1	PAxx	A
AC0	AIN2	PAxx	A
AC0	AIN3	PAxx	A
AC0	CMP0	PAxx	A
AC0	CMP1	PAxx	A

Related Links

32. PORT - I/O Pin Controller

46.5.2 Power Management

The AC will continue to operate in any sleep mode where the selected source clock is running. The AC's interrupts can be used to wake up the device from sleep modes. Events connected to the event system can trigger other operations in the system without exiting sleep modes.

46.5.3 Clocks

The AC bus clock (CLK_AC_APB) can be enabled and disabled in the Main Clock module, MCLK (see *MCLK - Main Clock*, and the default state of CLK_AC_APB can be found in *Peripheral Clock Masking*.

A generic clock (GCLK_AC) is required to clock the AC. This clock must be configured and enabled in the generic clock controller before using the AC. Refer to the Generic Clock Controller chapter for details.

This generic clock is asynchronous to the bus clock (CLK_AC_APB). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. Refer to Synchronization for further details.

Related Links

15.6.2.6 Peripheral Clock Masking

15. MCLK – Main Clock

46.5.4 DMA

Not applicable.

46.5.5 Interrupts

The interrupt request lines are connected to the interrupt controller. Using the AC interrupts requires the interrupt controller to be configured first. Refer to *Nested Vector Interrupt Controller* for details.

Related Links

10.2 Nested Vector Interrupt Controller

46.5.6 Events

The events are connected to the Event System. Refer to *EVSYS – Event System* for details on how to configure the Event System.

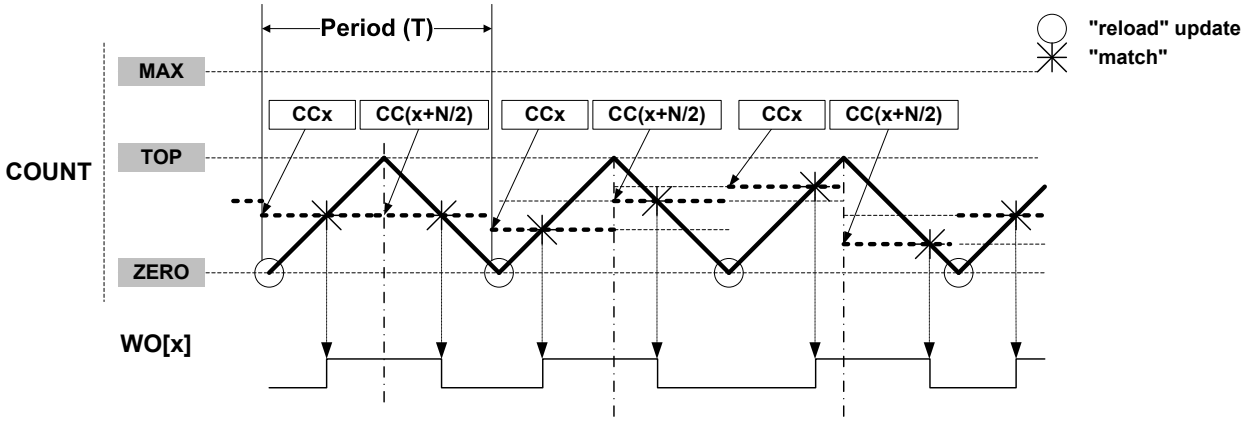
Bit 0 – DIR Counter Direction
This bit is used to change the direction of the counter.

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

Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

Figure 49-8. Dual-Slope Critical Pulse Width Modulation (N=CC_NUM)



49.6.2.5.8 Output Polarity

The polarity (WAVE.POLx) is available in all waveform output generation. In single-slope and dual-slope PWM operation, it is possible to invert the pulse edge alignment individually on start or end of a PWM cycle for each compare channels. The table below shows the waveform output set/clear conditions, depending on the settings of timer/counter, direction, and polarity.

Table 49-3. Waveform Generation Set/Clear Conditions

Waveform Generation operation	DIR	POLx	Waveform Generation Output Update	
			Set	Clear
Single-Slope PWM	0	0	Timer/counter matches TOP	Timer/counter matches CCx
		1	Timer/counter matches CC	Timer/counter matches TOP
	1	0	Timer/counter matches CC	Timer/counter matches ZERO
		1	Timer/counter matches ZERO	Timer/counter matches CC
Dual-Slope PWM	x	0	Timer/counter matches CC when counting up	Timer/counter matches CC when counting down
		1	Timer/counter matches CC when counting down	Timer/counter matches CC when counting up

In Normal and Match Frequency, the WAVE.POLx value represents the initial state of the waveform output.

49.6.2.6 Double Buffering

The Pattern (PATT), Period (PER) and Compare Channels (CCx) registers are all double buffered. Each buffer register has a buffer valid (PATTBUFV, PERBUFV and CCBUFVx) bit in the STATUS register, which indicates that the buffer register contains a valid value that can be copied into the corresponding register. As long as the respective buffer valid status flag (PATTBUFV, PERBUFV or CCBUFVx) are set to '1', the related SYNCBUSY bits are set (SYNCBUSY.PATT, SYNCBUSY.PER or SYNCBUSY.CCx), a write to the respective PATT/PATTBUF, PER/PERBUF or CCx/CCBUFx registers will generate a PAC error, and read access to the respective PATT, PER or CCx register is invalid.

When the buffer valid flag bit in the STATUS register is '1' and the Lock Update bit in the CTRLB register is set to '0', (writing CTRLBCLR.LUPD to '1'), double buffering is enabled: the data from buffer registers

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

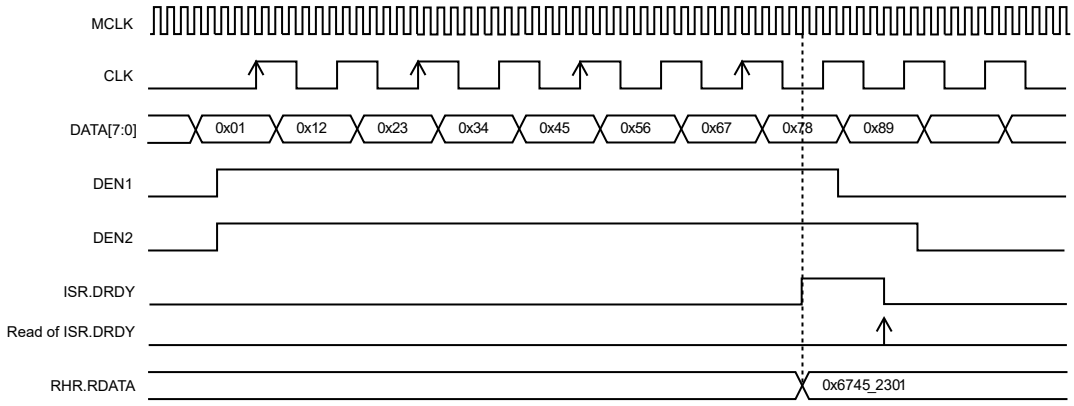


Figure 52-10. PCC Waveforms (ISIZE=10_BITS, DSIZE=2_DATA, ALWYS = 0, HALFS = 1, FRSTS = 0, SCALE = 0)

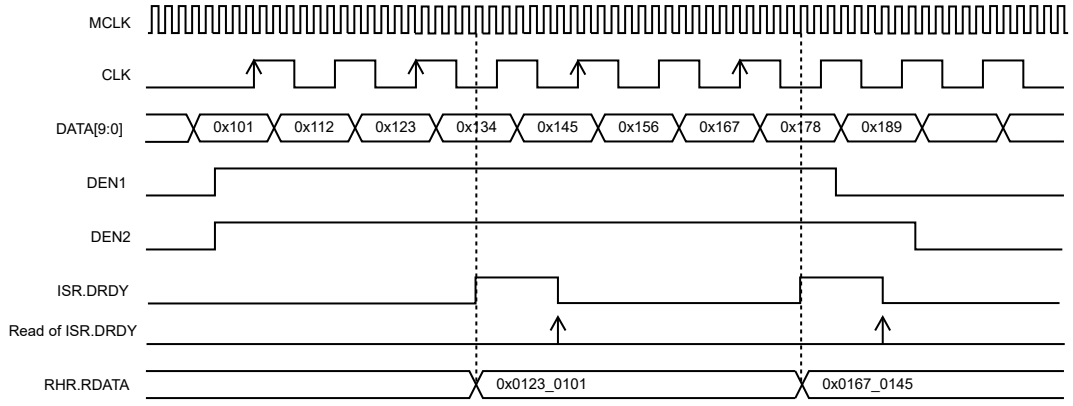
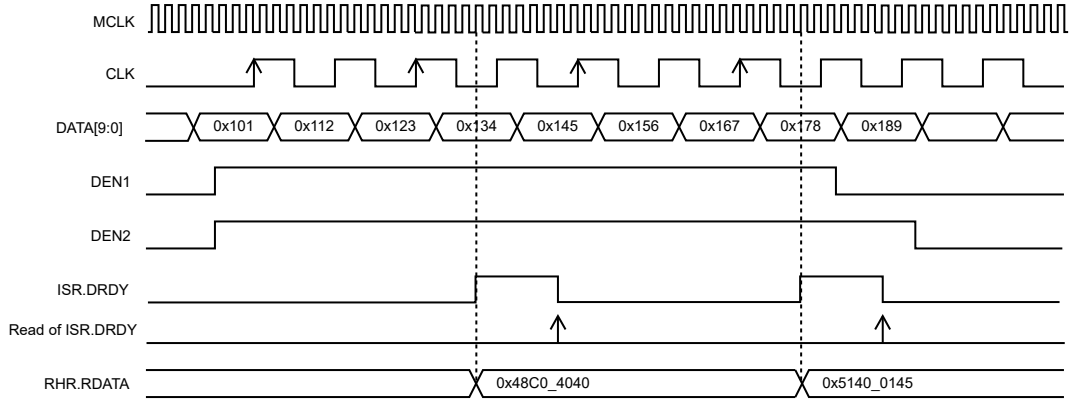


Figure 52-11. PCC Waveforms (ISIZE=10_BITS, DSIZE=2_DATA, ALWYS = 0, HALFS = 1, FRSTS = 0, SCALE = 1)



Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset
Writing a '0' to this bit has no effect.

Writing a '1' to this bit resets all registers in the PDEC (except DBGCTRL) to their initial state, and the PDEC will be disabled.

Writing a '1' to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.

Due to synchronization, there is a delay from writing CTRLA.SWRST until the Reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the Reset is complete.

Value	Description
0	There is no Reset operation ongoing.
1	A Reset operation is ongoing.

53.8.11 Prescaler Value

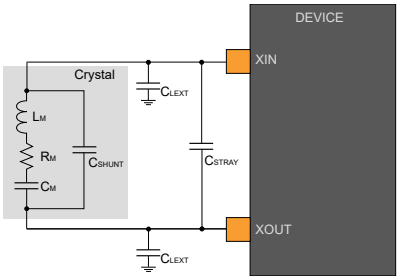
Name: PRESC
Offset: 0x14
Reset: 0x00
Property: Write-Synchronized

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

Bits 3:0 – PRESC[3:0] Prescaler Value
These bits select the GCLK prescaler factor.

Value	Name	Description
0	DIV1	No division
1	DIV2	Divide by 2
2	DIV4	Divide by 4
3	DIV8	Divide by 8
4	DIV16	Divide by 16
5	DIV32	Divide by 32
6	DIV64	Divide by 64
7	DIV128	Divide by 128
8	DIV256	Divide by 256
9	DIV512	Divide by 512
10	DIV1024	Divide by 1024

Figure 54-6. Oscillator Connection



The user must choose a crystal oscillator where the crystal load capacitance C_L is within the range given in the Table. The exact value of C_L can be found in the crystal datasheet. The capacitance of the external capacitors (C_{LEXT}) can then be computed as follows:

$C_{LEXT} = 2(C_L - C_{STRAY} - SHUNT)$,

where C_{SHUNT} is the shunt capacity of the crystal, and C_{STRAY} is the capacitance of the pins and the PCB:

$C_{STRAY} = C_{StrayDevice} + C_{StrayPCB}$, and $1/C_{StrayDevice} = 1/C_{XIN} + 1/C_{XOUT}$.

Table 54-40. Multi-Crystal Oscillator Electrical Characteristics

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
F_{OUT}	Crystal oscillator frequency		8	-	48	MHz
C_L	Crystal Load	F = 8 MHz	-	-	20	pF
		F = 16 MHz	-	-	20	
		F = 32 MHz	-	-	13	
		F = 48 MHz	-	-	13	
ESR	Crystal Equivalent Series Resistance - SF=3	F = 8 MHz, C_L=20 pF - IMULT = 0x3	-	-	181	Ω
		F = 16 MHz, C_L = 20 pF - IMULT = 0x4	-	-	180	
		F= 24 MHz, C_L = 20 pF - IMULT = 0x5	-	-	70	
		F = 48 MHz, C_L = 13 pF - IMULT = 0x6	-	-	70	
C_{XIN}	Parasitic load capacitor	-	-	6.3	-	pF
C_{XOUT}		-	-	5.9	-	
D_L	Drive Level (see **Note 1**)	ENALC = ON	-	-	100	µW
T_{START}	Startup time	F = 8 MHz, C_L = 20 pF, C_{SHUNT} = 2 pF - IMULT = 0x3	-	39700	72200	Cycles
		F = 16 MHz, C_L = 20 pF, C_{SHUNT} = 1.5 pF - IMULT = 0x4	-	37550	62000	
		F = 24 MHz, C_L = 20 pF, C_{SHUNT} = 2.5 pF - IMULT = 0x5	-	32700	68500	
		F = 48 MHz, C_L = 13 pF, C_{SHUNT} = 5 pF - IMULT = 0x6	-	18400	38500	

Note: To ensure that the crystal is not overdriven, the automatic loop control is recommended to be turned ON (ENALC = 1).