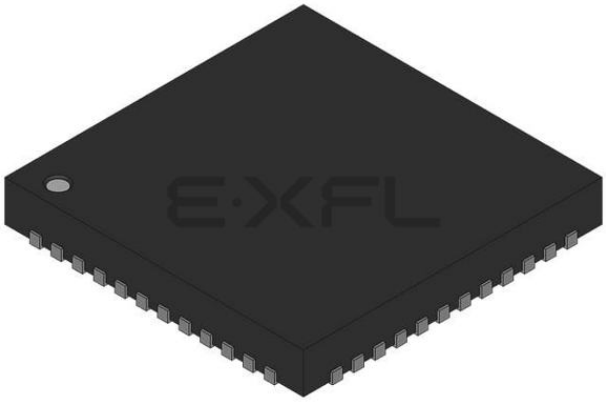


Welcome to E-XFL.COM

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	Obsolete
Core Processor	ARM® Cortex®-M3
Core Size	32-Bit Single-Core
Speed	48MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	34
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	24K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-VFQFN Exposed Pad
Supplier Device Package	48-QFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/atsam3n4aa-mu

3. Signal Description

Table 3-1 gives details on the signal name classified by peripheral.

Table 3-1. Signal Description List

Signal Name	Function	Type	Active Level	Voltage Reference	Comments
Power Supplies					
VDDIO	Peripherals I/O Lines Power Supply	Power			1.62V to 3.6V
VDDIN	Voltage Regulator, ADC and DAC Power Supply	Power			1.8V to 3.6V(3)
VDDOUT	Voltage Regulator Output	Power			1.8V Output
VDDPLL	Oscillator and PLL Power Supply	Power			1.65 V to 1.95V
VDDCORE	Power the core, the embedded memories and the peripherals	Power			1.65V to 1.95V Connected externally to VDDOUT
GND	Ground	Ground			
Clocks, Oscillators and PLLs					
XIN	Main Oscillator Input	Input		VDDIO	Reset State: - PIO Input - Internal Pull-up disabled(4) - Schmitt Trigger enabled(1)
XOUT	Main Oscillator Output	Output			
XIN32	Slow Clock Oscillator Input	Input			
XOUT32	Slow Clock Oscillator Output	Output			
PCK0–PCK2	Programmable Clock Output	Output			Reset State: - PIO Input - Internal Pull-up enabled - Schmitt Trigger enabled(1)
ICE and JTAG					
TCK/SWCLK	Test Clock/Serial Wire Clock	Input		VDDIO	Reset State: - SWJ-DP Mode - Internal pull-up disabled(1) - Schmitt Trigger enabled(1)
TDI	Test Data In	Input			
TDO/TRACESWO	Test Data Out/Trace Asynchronous Data Out	Output			
TMS/SWDIO	Test Mode Select /Serial Wire Input/Output	Input / I/O			
JTAGSEL	JTAG Selection	Input	High		Permanent Internal pull-down
Flash Memory					
ERASE	Flash and NVM Configuration Bits Erase Command	Input	High	VDDIO	Reset State: - Erase Input - Internal pull-down enabled - Schmitt Trigger enabled(1)
Reset/Test					
NRST	Microcontroller Reset	I/O	Low	VDDIO	Permanent Internal pull-up
TST	Test Mode Select	Input		VDDIO	Permanent Internal pull-down

10.3.3 PIO Controller C Multiplexing

Table 10-4. Multiplexing on PIO Controller C (PIOC)

I/O Line	Peripheral A	Peripheral B	Peripheral C	Extra Function	System Function	Comments
PC0						100-pin version
PC1						100-pin version
PC2						100-pin version
PC3						100-pin version
PC4		NPCS1				100-pin version
PC5						100-pin version
PC6						100-pin version
PC7		NPCS2				100-pin version
PC8		PWM0				100-pin version
PC9		PWM1				100-pin version
PC10		PWM2				100-pin version
PC11		PWM3				100-pin version
PC12				AD12 (1)		100-pin version
PC13				AD10(1)		100-pin version
PC14		PCK2				100-pin version
PC15				AD11(1)		100-pin version
PC16		PCK0				100-pin version
PC17		PCK1				100-pin version
PC18		PWM0				100-pin version
PC19		PWM1				100-pin version
PC20		PWM2				100-pin version
PC21		PWM3				100-pin version
PC22		PWM0				100-pin version
PC23		TIOA3				100-pin version
PC24		TIOB3				100-pin version
PC25		TCLK3				100-pin version
PC26		TIOA4				100-pin version
PC27		TIOB4				100-pin version
PC28		TCLK4				100-pin version
PC29		TIOA5		AD13(1)		100-pin version
PC30		TIOB5		AD14(1)		100-pin version
PC31		TCLK5		AD15(1)		100-pin version

1. To select this extra function, refer to Section 34.5.3 “Analog Inputs”.

11.11 About the instruction descriptions

The following sections give more information about using the instructions:

- “Operands” on page 77
- “Restrictions when using PC or SP” on page 77
- “Flexible second operand” on page 77
- “Shift Operations” on page 78
- “Address alignment” on page 81
- “PC-relative expressions” on page 81
- “Conditional execution” on page 81
- “Instruction width selection” on page 83.

11.11.1 Operands

An instruction operand can be an ARM register, a constant, or another instruction-specific parameter. Instructions act on the operands and often store the result in a destination register. When there is a destination register in the instruction, it is usually specified before the operands.

Operands in some instructions are flexible in that they can either be a register or a constant. See “Flexible second operand” .

11.11.2 Restrictions when using PC or SP

Many instructions have restrictions on whether you can use the *Program Counter* (PC) or *Stack Pointer* (SP) for the operands or destination register. See instruction descriptions for more information.

Bit[0] of any address you write to the PC with a BX, BLX, LDM, LDR, or POP instruction must be 1 for correct execution, because this bit indicates the required instruction set, and the Cortex-M3 processor only supports Thumb instructions.

11.11.3 Flexible second operand

Many general data processing instructions have a flexible second operand. This is shown as *Operand2* in the descriptions of the syntax of each instruction.

Operand2 can be a:

- “Constant”
- “Register with optional shift” on page 78

11.11.3.1 Constant

You specify an Operand2 constant in the form:

```
#constant
```

where `constant` can be:

- any constant that can be produced by shifting an 8-bit value left by any number of bits within a 32-bit word
- any constant of the form 0x00XY00XY
- any constant of the form 0xXY00XY00
- any constant of the form 0xXYXYXYXY.

In the constants shown above, X and Y are hexadecimal digits.

In addition, in a small number of instructions, `constant` can take a wider range of values. These are described in the individual instruction descriptions.

When an Operand2 constant is used with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to bit[31] of the constant, if the constant is greater than 255 and can be

Table 11-16 also shows the relationship between condition code suffixes and the N, Z, C, and V flags.

Table 11-16. Condition code suffixes

Suffix	Flags	Meaning
EQ	Z = 1	Equal
NE	Z = 0	Not equal
CS or HS	C = 1	Higher or same, unsigned ≥
CC or LO	C = 0	Lower, unsigned <
MI	N = 1	Negative
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned >
LS	C = 0 or Z = 1	Lower or same, unsigned ≤
GE	N = V	Greater than or equal, signed ≥
LT	N != V	Less than, signed <
GT	Z = 0 and N = V	Greater than, signed >
LE	Z = 1 and N != V	Less than or equal, signed ≤
AL	Can have any value	Always. This is the default when no suffix is specified.

11.11.7.3 Absolute value

The example below shows the use of a conditional instruction to find the absolute value of a number. R0 = ABS(R1).

```
MOVS    R0, R1          ; R0 = R1, setting flags
IT      MI              ; IT instruction for the negative condition
RSBMI   R0, R1, #0      ; If negative, R0 = -R1
```

11.11.7.4 Compare and update value

The example below shows the use of conditional instructions to update the value of R4 if the signed values R0 is greater than R1 and R2 is greater than R3.

```
CMP     R0, R1       ; Compare R0 and R1, setting flags
ITT     GT           ; IT instruction for the two GT conditions
CMPGT   R2, R3       ; If 'greater than', compare R2 and R3, setting flags
MOVGT   R4, R5       ; If still 'greater than', do R4 = R5
```

11.11.8 Instruction width selection

There are many instructions that can generate either a 16-bit encoding or a 32-bit encoding depending on the operands and destination register specified. For some of these instructions, you can force a specific instruction size by using an instruction width suffix. The .W suffix forces a 32-bit instruction encoding. The .N suffix forces a 16-bit instruction encoding.

If you specify an instruction width suffix and the assembler cannot generate an instruction encoding of the requested width, it generates an error.

In some cases it might be necessary to specify the .W suffix, for example if the operand is the label of an instruction or literal data, as in the case of branch instructions. This is because the assembler might not automatically generate the right size encoding.

- *Rm* must not be PC and must not be SP
- if the instruction is conditional, it must be the last instruction in the IT block

- with the exception of the ADD{*cond*} PC, PC, Rm instruction, *Rn* can be PC only in ADD and SUB, and only with the additional restrictions:
  - you must not specify the S suffix
  - the second operand must be a constant in the range 0 to 4095.
  -
  - When using the PC for an addition or a subtraction, bits[1:0] of the PC are rounded to b00 before performing the calculation, making the base address for the calculation word-aligned.
  - If you want to generate the address of an instruction, you have to adjust the constant based on the value of the PC. ARM recommends that you use the ADR instruction instead of ADD or SUB with *Rn* equal to the PC, because your assembler automatically calculates the correct constant for the ADR instruction.

When *Rd* is PC in the ADD{*cond*} PC, PC, Rm instruction:

- bit[0] of the value written to the PC is ignored
- a branch occurs to the address created by forcing bit[0] of that value to 0.

11.13.1.4 Condition flags

If S is specified, these instructions update the N, Z, C and V flags according to the result.

11.13.1.5 Examples

```
ADD     R2, R1, R3
SUBS    R8, R6, #240        ; Sets the flags on the result
RSB     R4, R4, #1280       ; Subtracts contents of R4 from 1280
ADCHI   R11, R0, R3         ; Only executed if C flag set and Z
                            ; flag clear
```

11.13.1.6 Multiword arithmetic examples

11.13.1.7 64-bit addition

The example below shows two instructions that add a 64-bit integer contained in R2 and R3 to another 64-bit integer contained in R0 and R1, and place the result in R4 and R5.

```
ADDS    R4, R0, R2    ; add the least significant words
ADC     R5, R1, R3    ; add the most significant words with carry
```

11.13.1.8 96-bit subtraction

Multiword values do not have to use consecutive registers. The example below shows instructions that subtract a 96-bit integer contained in R9, R1, and R11 from another contained in R6, R2, and R8. The example stores the result in R6, R9, and R2.

```
SUBS    R6, R6, R9     ; subtract the least significant words
SBCS    R9, R2, R1     ; subtract the middle words with carry
SBC     R2, R8, R11    ; subtract the most significant words with carry
```

11.16.1 BFC and BFI

Bit Field Clear and Bit Field Insert.

11.16.1.1 Syntax

```
BFC{cond} Rd, #lsb, #width
BFI{cond} Rd, Rn, #lsb, #width
```

where:

cond — is an optional condition code, see “Conditional execution” on page 81.

Rd — is the destination register.

Rn — is the source register.

lsb — is the position of the least significant bit of the bitfield.

lsb must be in the range 0 to 31.

width — is the width of the bitfield and must be in the range 1 to 32–*lsb*.

11.16.1.2 Operation

BFC clears a bitfield in a register. It clears *width* bits in *Rd*, starting at the low bit position *lsb*. Other bits in *Rd* are unchanged.

BFI copies a bitfield into one register from another register. It replaces *width* bits in *Rd* starting at the low bit position *lsb*, with *width* bits from *Rn* starting at bit[0]. Other bits in *Rd* are unchanged.

11.16.1.3 Restrictions

Do not use SP and do not use PC.

11.16.1.4 Condition flags

These instructions do not affect the flags.

11.16.1.5 Examples

```
BFC   R4, #8, #12       ; Clear bit 8 to bit 19 (12 bits) of R4 to 0
BFI   R9, R2, #8, #12   ; Replace bit 8 to bit 19 (12 bits) of R9 with
                        ; bit 0 to bit 11 from R2
```

11.21.2 Auxiliary Control Register

The ACTLR provides disable bits for the following processor functions:

- IT folding
- write buffer use for accesses to the default memory map
- interruption of multi-cycle instructions.

See the register summary in Table 11-30 on page 157 for the ACTLR attributes. The bit assignments are:

31	30	29	28	27	26	25	24
Reserved							

23	22	21	20	19	18	17	16
Reserved							

15	14	13	12	11	10	9	8
Reserved							

7	6	5	4	3	2	1	0
Reserved					DISFOLD	DISDEFWBUF	DISMCYCINT

- **DISFOLD**

When set to 1, disables IT folding. see “About IT folding” on page 158 for more information.

- **DISDEFWBUF**

When set to 1, disables write buffer use during default memory map accesses. This causes all bus faults to be precise bus faults but decreases performance because any store to memory must complete before the processor can execute the next instruction.

This bit only affects write buffers implemented in the Cortex-M3 processor.

- **DISMCYCINT**

When set to 1, disables interruption of load multiple and store multiple instructions. This increases the interrupt latency of the processor because any LDM or STM must complete before the processor can stack the current state and enter the interrupt handler.

11.21.2.1 About IT folding

In some situations, the processor can start executing the first instruction in an IT block while it is still executing the IT instruction. This behavior is called IT folding, and improves performance, However, IT folding can cause jitter in looping. If a task must avoid jitter, set the DISFOLD bit to 1 before executing the task, to disable IT folding.

The DWT contains counters for the following items:

- Clock cycle (CYCCNT)
- Folded instructions
- Load Store Unit (LSU) operations
- Sleep Cycles
- CPI (all instruction cycles except for the first cycle)
- Interrupt overhead

12.5.6 ITM (Instrumentation Trace Macrocell)

The ITM is an application driven trace source that supports printf style debugging to trace Operating System (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated by three different sources with several priority levels:

- **Software trace**: Software can write directly to ITM stimulus registers. This can be done thanks to the “printf” function. For more information, refer to Section 12.5.6.1 “How to Configure the ITM”.
- **Hardware trace**: The ITM emits packets generated by the DWT.
- **Time stamping**: Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp.

12.5.6.1 How to Configure the ITM

The following example describes how to output trace data in asynchronous trace mode.

- Configure the TPIU for asynchronous trace mode (refer to Section 12.5.6.3 “5.4.3. How to Configure the TPIU”)
- Enable the write accesses into the ITM registers by writing “0xC5ACCE55” into the Lock Access Register (Address: 0xE0000FB0)
- Write 0x00010015 into the Trace Control Register:
  - Enable ITM
  - Enable Synchronization packets
  - Enable SWO behavior
  - Fix the ATB ID to 1
- Write 0x1 into the Trace Enable Register:
  - Enable the Stimulus port 0
- Write 0x1 into the Trace Privilege Register:
  - Stimulus port 0 only accessed in privileged mode (Clearing a bit in this register will result in the corresponding stimulus port being accessible in user mode.)
- Write into the Stimulus port 0 register: TPIU (Trace Port Interface Unit)

  The TPIU acts as a bridge between the on-chip trace data and the Instruction Trace Macrocell (ITM).

  The TPIU formats and transmits trace data off-chip at frequencies asynchronous to the core.

12.5.6.2 Asynchronous Mode

The TPIU is configured in asynchronous mode, trace data are output using the single TRACESWO pin. The TRACESWO signal is multiplexed with the TDO signal of the JTAG Debug Port. As a consequence, asynchronous trace mode is only available when the Serial Wire Debug mode is selected since TDO signal is used in JTAG debug mode.

Two encoding formats are available for the single pin output:

- Manchester encoded stream. This is the reset value.
- NRZ_based UART byte structure

Table 20-4. Write Handshake

Step	Programmer Action	Device Action	Data I/O
1	Sets MODE and DATA signals	Waits for NCMD low	Input
2	Clears NCMD signal	Latches MODE and DATA	Input
3	Waits for RDY low	Clears RDY signal	Input
4	Releases MODE and DATA signals	Executes command and polls NCMD high	Input
5	Sets NCMD signal	Executes command and polls NCMD high	Input
6	Waits for RDY high	Sets RDY	Input

20.2.4.2 Read Handshaking

For details on the read handshaking sequence, refer toFigure 20-5 Figure 20-6 and Table 20-5.

Figure 20-5. SAM3NxB/C (64/100 pins) Parallel Programming Timing, Read Sequence

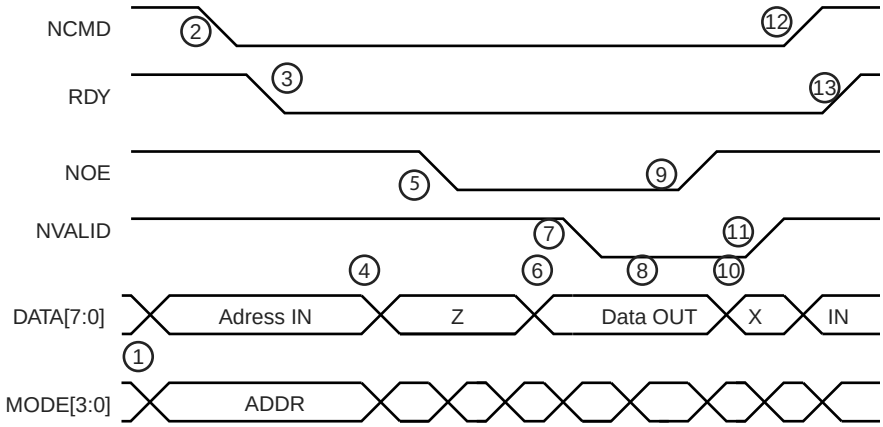
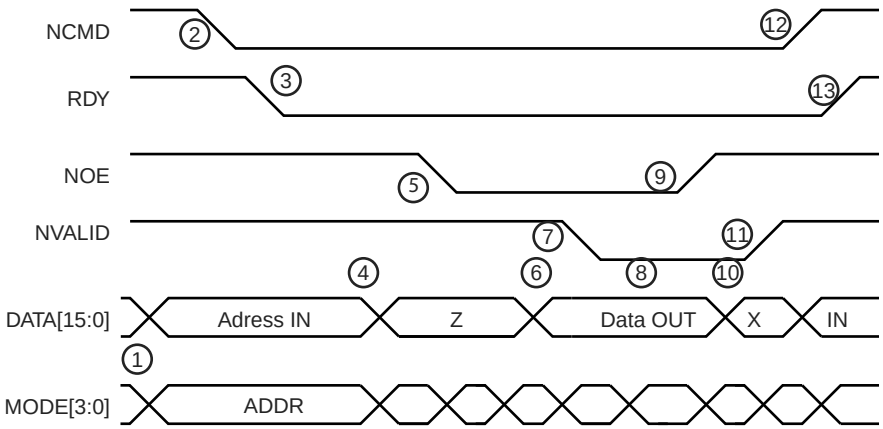


Figure 20-6. SAM3NxA (48 pins) Parallel Programming Timing, Read Sequence



22.8.6 Write Protect Status Register

Name: MATRIX_WPSR

Address: 0x400E03E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

For more details on MATRIX_WPSR, refer to Section 22.7 “Write Protect Registers” on page 303.

- **WPVS: Write Protect Violation Status**

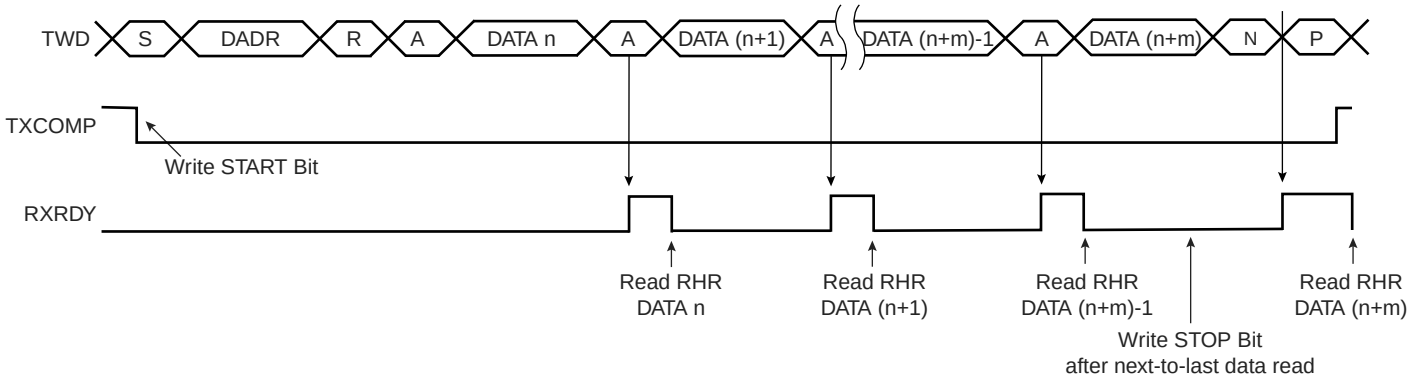
0: No Write Protect Violation has occurred since the last write of MATRIX_WPMR.

1: At least one Write Protect Violation has occurred since the last write of MATRIX_WPMR.

- **WPVSRC: Write Protect Violation Source**

Should be written at value 0x4D4154 (“MAT” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Figure 29-10. Master Read with Multiple Data Bytes



RXRDY is used as Receive Ready for the PDC receive channel.

29.8.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

29.8.6.1 7-bit Slave Addressing

When Addressing 7-bit slave devices, the internal address bytes are used to perform random address (read or write) accesses to reach one or more data bytes, within a memory page location in a serial memory, for example. When performing read operations with an internal address, the TWI performs a write operation to set the internal address into the slave device, and then switch to Master Receiver mode. Note that the second start condition (after sending the IADR) is sometimes called “repeated start” (Sr) in I2C fully-compatible devices. See Figure 29-12. See Figure 29-11 and Figure 29-13 for Master Write operation with internal address.

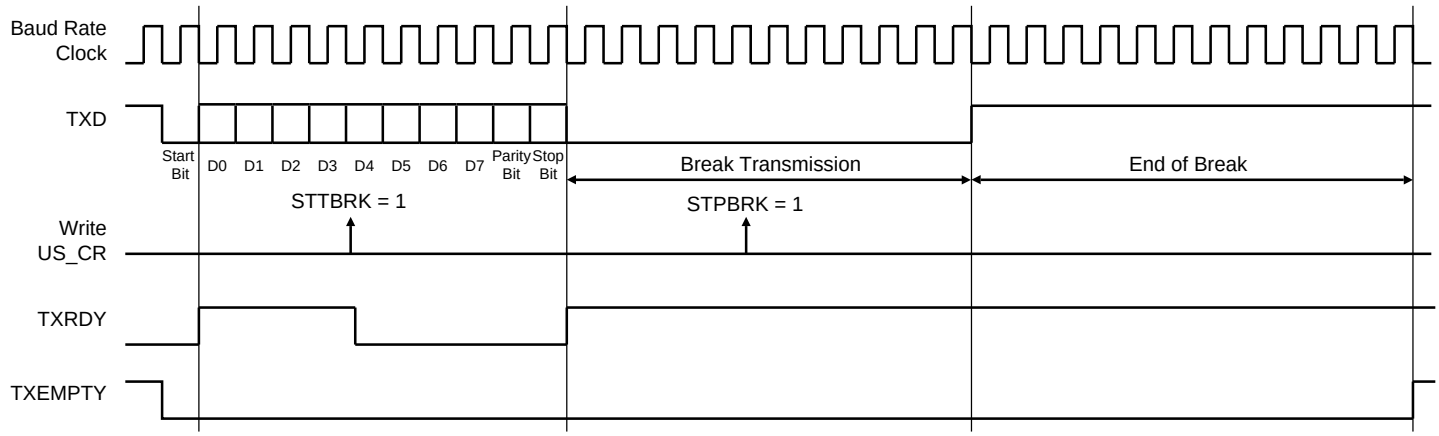
The three internal address bytes are configurable through the Master Mode register (TWI_MMR).

If the slave device supports only a 7-bit address, i.e. no internal address, **IADRSZ** must be set to 0.

In the figures below the following abbreviations are used:

- S Start
- Sr Repeated Start
- P Stop
- W Write
- R Read
- A Acknowledge
- N Not Acknowledge
- DADR Device Address
- IADR Internal Address

Figure 31-16. Break Transmission



31.7.3.11 Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data to 0x00, but FRAME remains low.

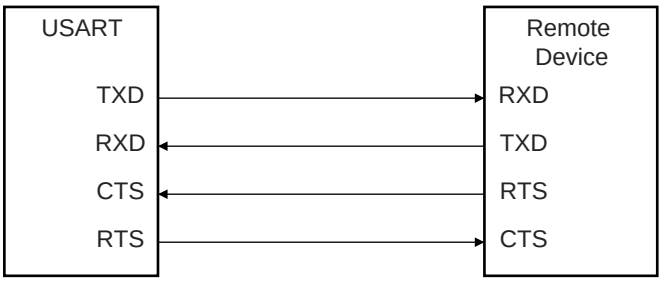
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA to 1.

An end of receive break is detected by a high level for at least 2/16 of a bit period in asynchronous operating mode or one sample at high level in synchronous operating mode. The end of break detection also asserts the RXBRK bit.

31.7.3.12 Hardware Handshaking

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 31-17.

Figure 31-17. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the PDC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 31-18 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the PDC channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the PDC clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

31.8.15 USART Write Protect Mode Register

Name: US_WPMR

Addresses: 0x400240E4 (0), 0x400280E4 (1)

Access: Read-write

Reset: See Table 31-15

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 Protect if WPKEY corresponds to 0x555341 (“USA” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x555341 (“USA” in ASCII).

Protects the registers:

- “USART Mode Register” on page 567
- “USART Baud Rate Generator Register” on page 577
- “USART Receiver Time-out Register” on page 578
- “USART Transmitter Timeguard Register” on page 579
- “USART FI DI RATIO Register” on page 580
- “USART IrDA FILTER Register” on page 582

- **WPKEY: Write Protect KEY**

Should be written at value 0x555341 (“USA” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

32.6.11.1 WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 2^{16}-1. Once 2^{16}-1 has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 32-7.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 32-8.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 32-7. WAVSEL = 00 without Trigger

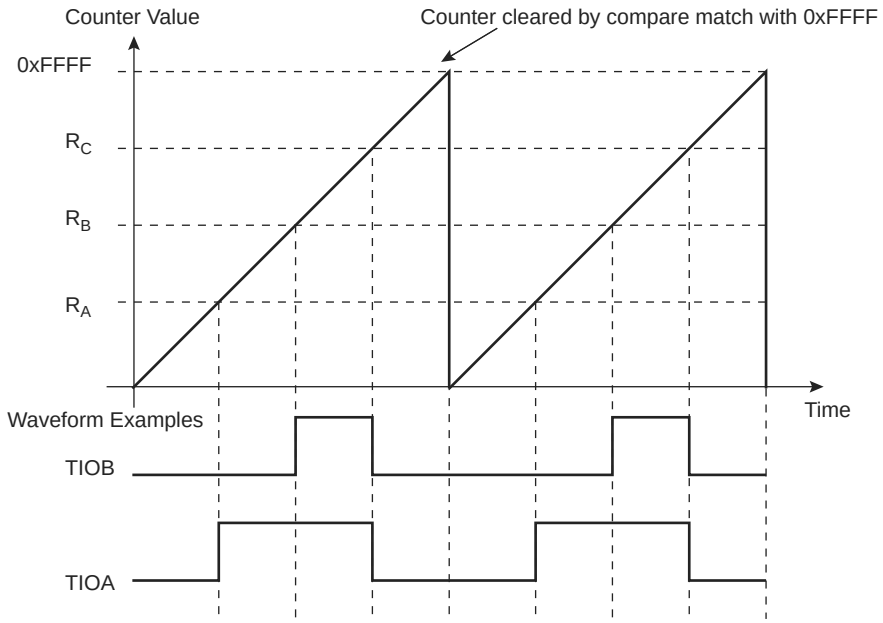
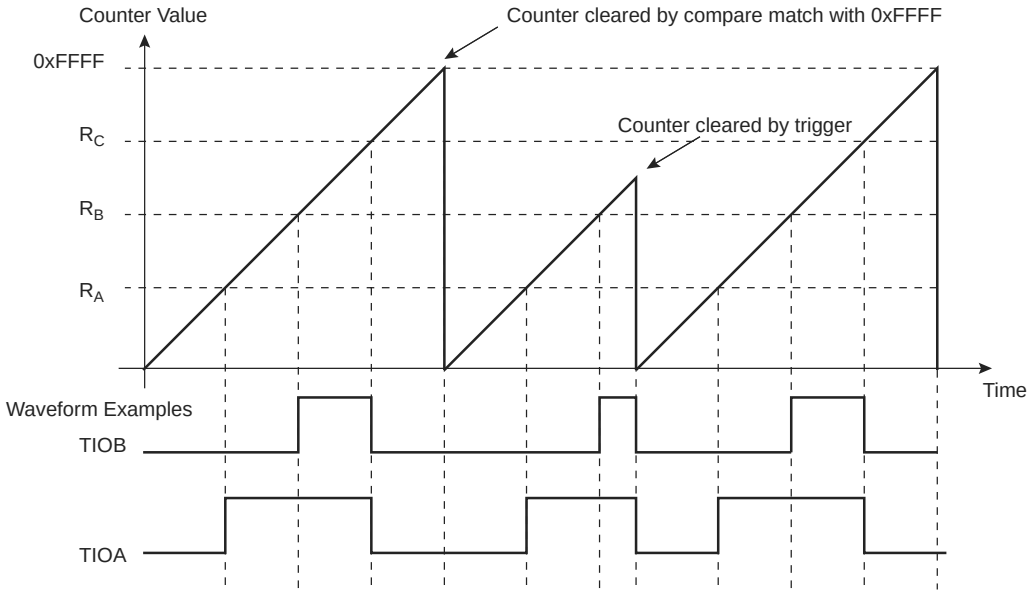


Figure 32-8. WAVSEL = 00 with Trigger



32.7.17 TC QDEC Interrupt Mask Register

Name: TC_QIMR

Address: 0x400100D0 (0), 0x400140D0 (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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: Index**

0: The interrupt on IDX input is disabled.

1: The interrupt on IDX input is enabled.

- **DIRCHG: Direction Change**

0: The interrupt on rotation direction change is disabled.

1: The interrupt on rotation direction change is enabled.

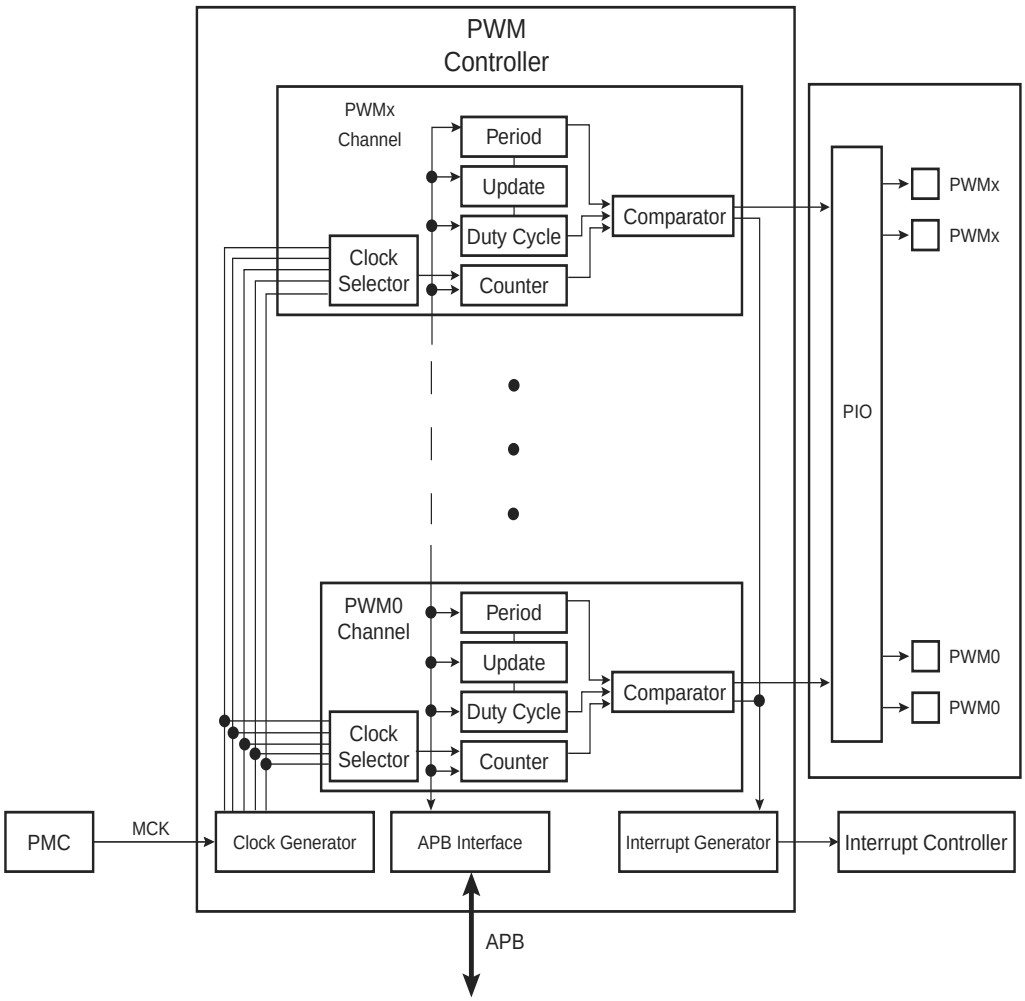
- **QERR: Quadrature Error**

0: The interrupt on quadrature error is disabled.

1: The interrupt on quadrature error is enabled.

33.3 Block Diagram

Figure 33-1. Pulse Width Modulation Controller Block Diagram



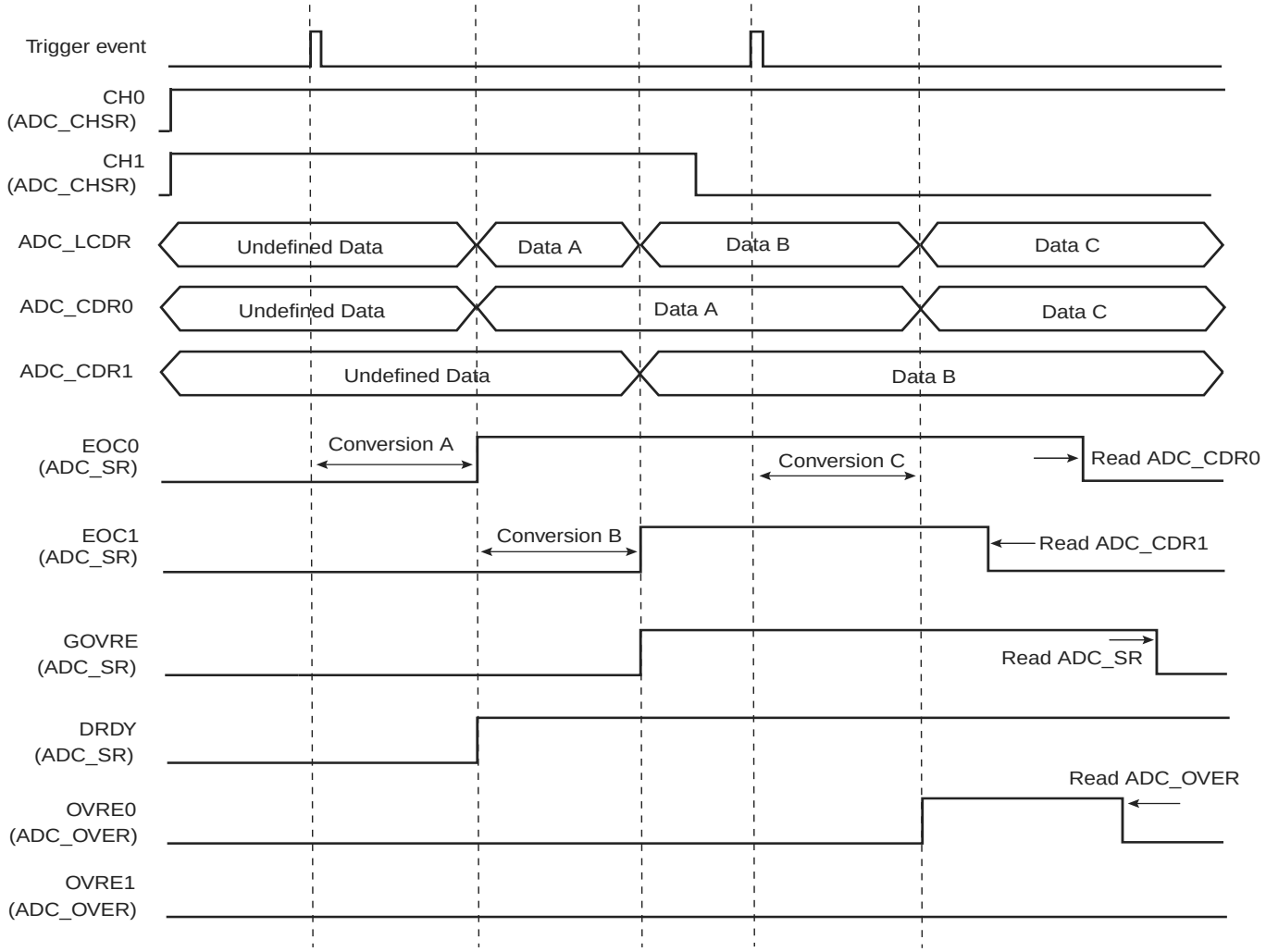
33.4 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

Table 33-1. I/O Line Description

Name	Description	Type
PWMx	PWM Waveform Output for channel x	Output

Figure 34-4. GOVRE and OVREx Flag Behavior



Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

34.7.11 ADC Interrupt Mask Register

Name: ADC_IMR

Address: 0x4003802C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	RXBUFF	ENDRX	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
EOC15	EOC14	EOC13	EOC12	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Mask x**
- **DRDY: Data Ready Interrupt Mask**
- **GOVRE: General Overrun Error Interrupt Mask**
- **COMPE: Comparison Event Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

35.6.5 Conversion Width

The WORD field of the DACC Mode Register allows the user to switch between half-word and word transfer.

In half-word transfer mode only one 10-bit data item is sampled (DACC_MR[9:0]) per DACC_CDR register write.

In word transfer mode each time the DACC_CDR register is written 2 data items are sampled. First data item sampled for conversion will be DACC_CDR[9:0] and the second DACC_CDR[25:16].

35.6.6 DAC Timings

The DAC startup time must be defined by the user in the STARTUP field of the DACC Mode Register.

The DAC maximum clock frequency is 13 MHz, therefore the internal trigger period can be configured through the CLKDIV field of the DACC Mode Register.

35.6.7 Write Protection Registers

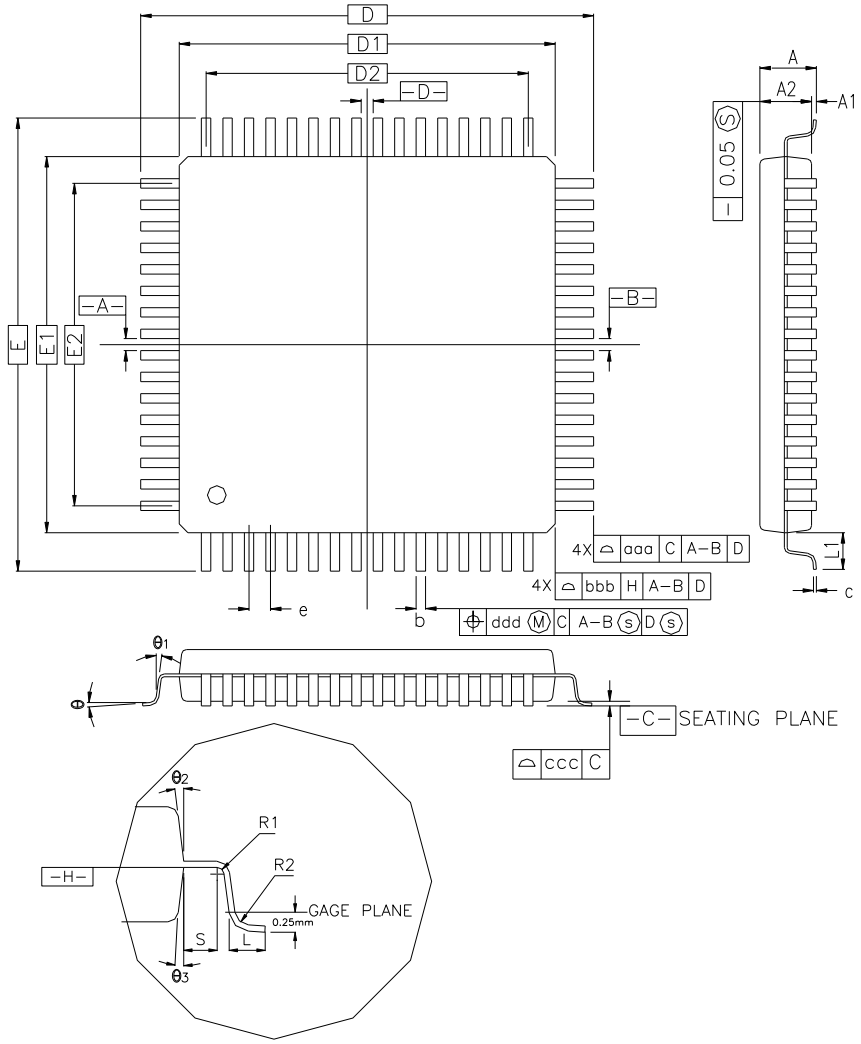
In order to bring security to the DACC, a write protection system has been implemented.

The write protection mode prevents the write of the DACC Mode Register. When this mode is enabled and the protected register is written an error is generated in the DACC Write Protect Status Register and the register write request is canceled. When a write protection error occurs, the WPROTERR flag is set and the address of the corresponding canceled register write is available in the WPROTADRR field of the DACC Write Protect Status Register.

Due to the nature of the write protection feature, enabling and disabling the write protection mode requires the use of a security code. Thus when enabling or disabling the write protection mode, the WPKEY field of the DACC Write Protect Mode Register must be filled with the “DAC” ASCII code (corresponding to 0x444143) otherwise the register write will be canceled.

37. Mechanical Characteristics

Figure 37-1. 100-lead LQFP Package Mechanical Drawing



CONTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A_1	0.05	—	0.15	0.002	—	0.006
A_2	1.35	1.40	1.45	0.053	0.055	0.057
D	16.00 BSC.			0.630 BSC.		
D_1	14.00 BSC.			0.551 BSC.		
E	16.00 BSC.			0.630 BSC.		
E_1	14.00 BSC.			0.551 BSC.		
R_2	0.08	—	0.20	0.003	—	0.008
R_1	0.08	—	—	0.003	—	—
θ	0°	3.5°	7°	0°	3.5°	7°
θ_1	0°	—	—	0°	—	—
θ_2	11°	12°	13°	11°	12°	13°
θ_3	11°	12°	13°	11°	12°	13°
c	0.09	—	0.20	0.004	—	0.008
L	0.45	0.60	0.75	0.018	0.024	0.030
L_1	1.00 REF			0.039 REF		
S	0.20	—	—	0.008	—	—
b	0.17	0.20	0.27	0.007	0.008	0.011
e	0.50 BSC.			0.020 BSC.		
D2	12.00			0.472		
E2	12.00			0.472		
TOLERANCES OF FORM AND POSITION						
aaa	0.20			0.008		
bbb	0.20			0.008		
ccc	0.08			0.003		
ddd	0.08			0.003		

Note: 1. This drawing is for general information only. Refer to JEDEC Drawing MS-026 for additional information.

Table 37-1. Device and 100-lead LQFP Package Maximum Weight

SAM3N4/2/1	800	mg

Table 37-2. 100-lead LQFP Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	e3

Table 37-3. 100-lead LQFP Package Characteristics

Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.