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	e200z4d
Core Size	32-Bit Single-Core
Speed	80MHz
Connectivity	CANbus, LINbus, SPI, UART/USART
Peripherals	DMA, POR, PWM, WDT
Number of I/O	-
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 32x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mpxs2005vlq80

Chapter 32
Mode Entry Module (MC_ME)

Table 3-1. 144 LQFP pin function summary (continued)

Pin #	Port/function	Peripheral	Output function	Input function
122	A[12]	SIUL	GPIO[12]	GPIO[12]
		DSPI_2	SOUT	—
		FlexPWM_0	A[2]	A[2]
		FlexPWM_0	B[2]	B[2]
		SIUL	—	EIRQ[11]
123	JCOMP	—	—	JCOMP
124	C[15]	SIUL	GPIO[47]	GPIO[47]
		FlexRay	CA_TR_EN	—
		eTimer_1	ETC[0]	ETC[0]
		FlexPWM_0	A[1]	A[1]
		CTU_0	—	EXT_IN
		FlexPWM_0	—	EXT_SYNC
125	D[0]	SIUL	GPIO[48]	GPIO[48]
		FlexRay	CA_TX	—
		eTimer_1	ETC[1]	ETC[1]
		FlexPWM_0	B[1]	B[1]
126	$V_{DD\_HV\_IO}$	—		
127	$V_{SS\_HV\_IO}$	—		
128	D[3]	SIUL	GPIO[51]	GPIO[51]
		FlexRay	CB_TX	—
		eTimer_1	ETC[4]	ETC[4]
		FlexPWM_0	A[3]	A[3]
129	D[4]	SIUL	GPIO[52]	GPIO[52]
		FlexRay	CB_TR_EN	—
		eTimer_1	ETC[5]	ETC[5]
		FlexPWM_0	B[3]	B[3]
130	$V_{DD\_HV\_REG\_2}$	—		
131	$V_{DD\_LV\_COR}$	—		
132	$V_{SS\_LV\_COR}$	—		
133	F[0]	SIUL	GPIO[80]	GPIO[80]
		FlexPWM_0	A[1]	A[1]
		eTimer_0	—	ETC[2]
		SIUL	—	EIRQ[28]

Table 3-3. Supply pins (continued)

Supply			Pin #	
Symbol	**Description**		**144 pkg**	**257 pkg**
V_{DD} 1V2	VDD_LV_COR Decoupling pins for core logic. Decoupling capacitor must be connected between these pins and the nearest $V_{DD\_LV\_COR}$ pin.		131	VDD_LV[1]
V_{SS} 1V2	VSS_LV_COR Decoupling pins for core logic. Decoupling capacitor must be connected between these pins and the nearest $V_{DD\_LV\_COR}$ pin.		132	VSS_LV[2]
V_{DD} 1V2	VDD_LV_COR / Decoupling pins for core logic. Decoupling capacitor must be connected between these pins and the nearest $V_{DD\_LV\_COR}$ pin.		135	VDD_LV[1]
V_{SS} 1V2	VSS_LV_COR / Decoupling pins for core logic. Decoupling capacitor must be connected between these pins and the nearest $V_{DD\_LV\_COR}$ pin.		137	VSS_LV[2]

NOTES:
1 VDD_LV balls are tied together on the 257 MAPBGA substrate.
2 VSS_LV balls are tied together on the 257 MAPBGA substrate.
3 VDD_HV balls are tied together on the 257 MAPBGA substrate.
4 VSS_HV balls are tied together on the 257 MAPBGA substrate.

3.3 System pins

Table 3-4. System pins

Symbol	Description	Direction	Pin #	
			144 pkg	**257 pkg**
Dedicated pins				
MDO0[1]	Nexus Message Data Output — line	Output only	9	E1
$\overline{\text{NMI}}$[2]	Non Maskable Interrupt	Input only	1	E4
XTAL	Input for oscillator amplifier circuit and internal clock generator	Input only	29	N1
EXTAL	Oscillator amplifier output	Output only	30	R1
TMS[2]	JTAG state machine control	Input only	87	M16
TCK[2]	JTAG clock	Input only	88	L15
JCOMP[3]	JTAG compliance select	Input only	123	C10

9.3.9.2 Presampling Register 0 (PSR0)

Address: Base + 0x084 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRES15	PRES14	PRES13	PRES12	PRES11	PRES10	PRES9	PRES8	PRES7	PRES6	PRES5	PRES4	PRES3	PRES2	PRES1	PRES0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-13. Presampling Register 0 (PSR0)

Table 9-16. PSR0 field descriptions

Field	Description
PRES*n*	Presampling enable 0 Presampling for channel *n* is disabled 1 Presampling for channel *n* is enabled

9.3.10 Conversion timing registers

9.3.10.1 Conversion Timing Register 0 (CTR0)

This register configures the conversion timing for channels 0–14. Timings for channel 15 (the TSENS channel) are configured using CTR1 (see Section 9.3.10.2, Conversion Timing Register 1 (CTR1)).

Address: Base + 0x094 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INP LATCH	0	OFFSHIFT		0	INPCMP		0	INPSAMP							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

Figure 9-14. Conversion Timing Register 0 (CTR0)

Table 9-17. CTR0 field descriptions

Field	Description
INPLATCH	Configuration bit for latching phase duration (see Figure 9-15)

fractional part and thus, two registers (STAW1AR and STAW1BR) are provided. The comparison is done first for integer part using the threshold values programmed in STAW1AR. If integer part lies in the range, the fractional part comparison is skipped otherwise it is compared with the values programmed in STAW1BR. Table 9-48 summarizes this feature for STEP1.

Table 9-48. Algorithm S (STEP1) threshold comparison

STDR2[IDATA] (integer part)	STDR2[FDATA] (fractional part)	STSR1[ERR_S1]
> STAW1AR[THRH]	Any value	Set
< STAW1AR[THRL]	Any value	Set
== STAW1AR[THRH]	> STAW1BR[THRH]	Set
== STAW1AR[THRL]	< STAW1BR[THRL]	Set

For Algorithm S STEP2, (VREF/VREF) is measured in order to check the integrity of sampling signal. For this particular conversion, no higher threshold value is required as the ideal value is 0xFFF. Only lower threshold value is programmed in STAW2R.

For Algorithm C, a separate register is provided for Step0. In Step0 an offset for other steps is measured. The converted data is compared with the threshold values provided by STAW5R if STAW4R[AWDE] is set. For other steps, this offset is subtracted from converted data before performing watchdog checks.

9.4.11.6 Watchdog timer

The watchdog timer is an additional check which monitors the sequence of the self testing algorithm implemented and also that the algorithm is completed within a safe time period. The Watchdog timers can be enabled for CPU as well CTU conversions. Each algorithm has a different watchdog timer which runs independently of the other. The watchdog timer for a particular algorithm can be enabled by setting STAW*n*R[WDTE] bit. The safe time value can be programmed in STBRR[WDT] field (default value is 10 ms assuming a 120 MHz clock).

The safe time is measured starting from Step0 of the algorithm (including all normal chain conversions in between) to the point where Step0 of the same algorithm starts again.

The sequence is as follows:

- Program NCMR0 to select channels to be converted for normal conversion in scan mode (MODE = 1).
- Select the Self Testing algorithm in STCR3.ALG. By default, all three algorithms are selected i.e. all algorithms will be executed step by step one after the other.
- Enable self testing channel by setting EN bit in STCR2 register.
- Program safe period value in STBRR.WDT field.
- Enable watchdog timer by setting STAWxR.WDTE bit. Assume setting of WDTE bit to be ‘t0’. (It is important to do all the programming first and then enable WDTE bit as the safe time check is also performed between setting of WDTE bit and start of STEP0 to check that algorithm has started within the safe time).
- Start the normal conversion by setting MCR[NSTART].

- **Sequential mode**: each event source for the incoming signals can generate one trigger event output, the next event source generates the next trigger event output, and so on in a predefined sequence. For the ADC, a commands list is entered by the CPU and the sequence of the selected incoming trigger events generate commands or stream of commands.

The TGS Mode is selected using the TGS_M bit in the TGS Control Register.

13.4.4 TGS in triggered mode

The structure of the TGS in Triggered Mode is shown in Figure 13-3.

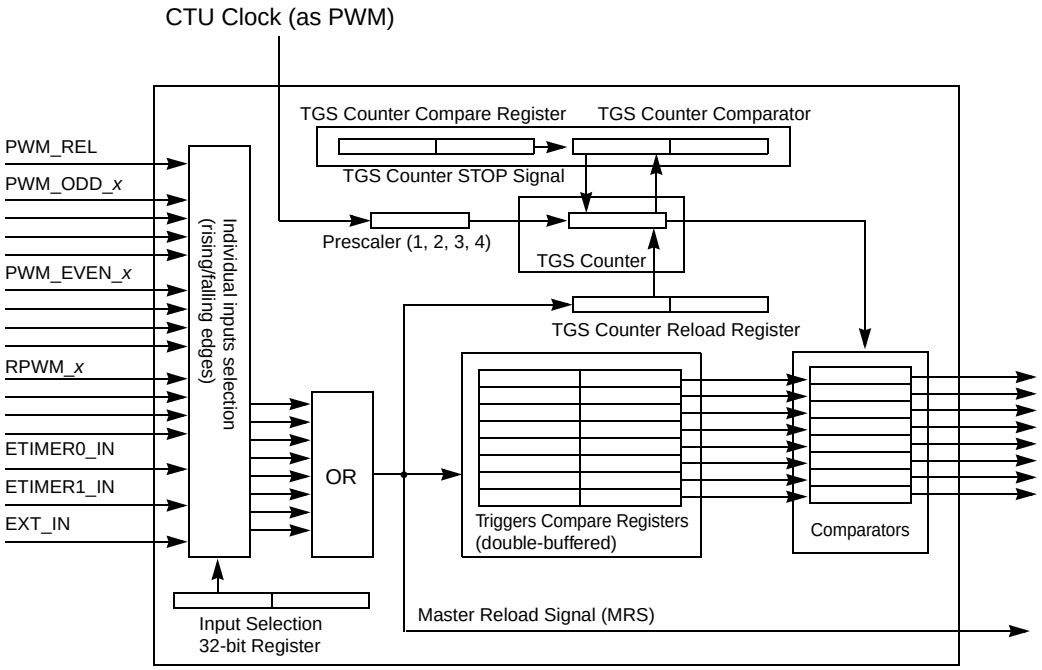


Figure 13-3. TGS in triggered mode

The TGS has 16 input signals, each of which is selected from the input selection register (TGSISR), selecting the states inactive, rising, falling or both. Depending on the selection, up to 32 input events can be enabled. These signals are OR-ed in order to generate the MRS. The MRS, at the beginning of the control cycle "N" (defined by the MRS occurrence), is used to pre-load the TGS counter register, using the pre-load value written into the double-buffered register (TGSCRR), during the control cycle "N-1", and to reload all the double-buffered registers (Trigger Compare registers, TGSCR, TGSCRR itself etc).

The triggers list registers, consist of 8 compare registers. Each triggers list register is associated with a comparator. On reload (MRS occurrence), the comparators are disabled: 1 TGS clock cycle is necessary to enable them and to start the counting. The MRS is output together with individual trigger signals. The MRS can be performed by hardware or by software. The MRS_SG bit in the CTU control register, if set to 1, generates equivalent software MRS (i.e. resets/reloads TGS Counter and reloads all double-buffered registers). This bit is cleared by each hardware or software MRS occurrence.

The TGS counter compare register and the TGS counter comparator, are used to stop the TGS counter when it reaches the value stored in the TGS counter compare register, before an MRS occurs.

Register address: ECSM Base + 0x0057

	0	1	2	3	4	5	6	7
R	WRITE	SIZE			PROTECTION			
W								
RESET:	-	-	-	-	-	-	-	-

= Unimplemented

Figure 21-13. Platform Flash Memory ECC Attributes (PFEAT) Register

Table 21-15. PFEAT field descriptions

Field	Description
WRITE	**AMBA-AHB HWRITE** 0 = AMBA-AHB read access 1 = AMBA-AHB write access
SIZE	**AMBA-AHB HSIZE[0:2]** 0b000 = 8-bit AMBA-AHB access 0b001 = 16-bit AMBA-AHB access 0b010 = 32-bit AMBA-AHB access 0b1xx = Reserved
PROTECTION	**AMBA-AHB HPROT[0:3]** Protection[3]: Cacheable 0 = Non-cacheable,1 = Cacheable Protection[2]: Bufferable 0 = Non-bufferable,1 = Bufferable Protection[1]: Mode 0 = User mode, 1 = Supervisor mode Protection[0]: Type 0 = I-Fetch, 1 = Data

21.4.2.15 Platform Flash Memory ECC Data Registers (PFEDRL and PFEDRH)

These two 32-bit registers contain a 64-bit field, PFEDR, for capturing the data associated with the last, properly-enabled ECC event in the platform flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the platform flash memory causes the address, attributes and data associated with the access to be loaded into the PFEAR, PFEMR, PFEAT, PFEDRH, and PFEDRL registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

These registers can only be read from the IPS programming model; any attempted write is ignored. If no flash memory ECC event is defined to be handled for this module, accesses to these registers will terminate with an error.

Table 21-19. Platform RAM syndrome mapping for single-bit correctable errors (continued)

PRESR	Data Bit in Error
0x57	DATA ODD BANK[22]
0x58	DATA ODD BANK[9]
0x5b	DATA ODD BANK[23]
0x5d	DATA ODD BANK[24]
0x5e	DATA ODD BANK[25]
0x61	DATA ODD BANK[10]
0x62	DATA ODD BANK[11]
0x64	DATA ODD BANK[12]
0x67	DATA ODD BANK[26]
0x68	DATA ODD BANK[13]
0x6b	DATA ODD BANK[27]
0x6d	DATA ODD BANK[28]
0x6e	DATA ODD BANK[29]
0x70	DATA ODD BANK[14]
0x73	DATA ODD BANK[15]
0x75	DATA ODD BANK[16]
0x76	DATA ODD BANK[17]
0x79	DATA ODD BANK[18]
0x7a	DATA ODD BANK[19]
0x7c	DATA ODD BANK[20]
0x7f	DATA ODD BANK[30]
0x81	ECC EVEN[0]
0x82	ECC EVEN[1]
0x84	ECC EVEN[2]
0x87	DATA EVEN BANK[31]
0x88	ECC EVEN[3]
0x90	ECC EVEN[4]
0xa0	ECC EVEN[5]
0xc0	ECC EVEN[6]
0xc3	DATA EVEN BANK[0]
0xc5	DATA EVEN BANK[1]
0xc6	DATA EVEN BANK[2]
0xc9	DATA EVEN BANK[3]
0xca	DATA EVEN BANK[4]
0xcc	DATA EVEN BANK[5]
0xcf	DATA EVEN BANK[21]
0xd1	DATA EVEN BANK[6]
0xd2	DATA EVEN BANK[7]
0xd4	DATA EVEN BANK[8]
0xd7	DATA EVEN BANK[22]
0xd8	DATA EVEN BANK[9]
0xdb	DATA EVEN BANK[23]

during erase sequence interlock writes are ignored. The user may terminate the erase sequence by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV assuming DONE is low, EHV is high and ESUS is low. An erase abort forces the module to step 8 of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. The user may not abort an erase sequence while in erase suspend.

CAUTION

Aborting an erase operation leaves the FC blocks being erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

23.1.5.7 C90FL Erase Suspend/Resume

The erase sequence may be suspended to allow read access to the FC. The erase sequence may also be suspended to program (Erase-Suspended Program) the FC. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to FC locations targeted for program and blocks targeted for erase return indeterminate data. Programming locations in blocks targeted for erase during erase-suspended program may result in corrupted data. Read While Write may also be used to read the array during an erase sequence providing the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the module to start the sequence which places it in erase suspend. The user must wait until MCR[DONE] = 1 before the module is suspended and further actions are attempted. MCR[DONE] goes high no more than Tesus after MCR[ESUS] is set to a 1. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence the user must first clear MCR[EHV]. If a program sequence is initiated the values of SOC specific shadow enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its “erase” value. FC reads while MCR[ESUS] = 1 from the block(s) being erased return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase suspended program) before MCR[ESUS] can be cleared to resume the operation. The module continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

CAUTION

Repeated suspends at a high frequency may result in the operation timing out, and the flash module will respond by completing the operation with a fail code (MCR[PEG] = 0). The minimum time between erase suspends to ensure this does not occur is 200 µs.

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Free Running Timer, the Error Counter Register and the Message Buffers, which cannot be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit, which will resume the clocks and negate the LPM_ACK bit.

24.4.9.3 Stop Mode

This is a system low power mode in which all MCU clocks are stopped for maximum power savings. If FlexCAN receives the global Stop Mode request during Freeze Mode, it sets the LPM_ACK bit, negates the FRZ_ACK bit and then sends a Stop Acknowledge signal to the CPU, in order to shut down the clocks globally. If Stop Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOT_RDY and LPM_ACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Exiting Stop Mode is done in one of the following ways:

- CPU resuming the clocks and removing the Stop Mode request
- CPU resuming the clocks and Stop Mode request as a result of the Self Wake mechanism

In the Self Wake mechanism, if the SLF_WAK bit in MCR Register was set at the time FlexCAN entered Stop Mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAK_INT bit in the ESR Register and, if enabled by the WAK_MSK bit in MCR, generates a Wake Up interrupt to the CPU. Upon receiving the interrupt, the CPU should resume the clocks and remove the Stop Mode request. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. Table 24-12 details the effect of SLF_WAK and WAK_MSK upon wake-up from Stop Mode. Note that wake-up from Stop Mode only works when both bits are asserted.

Table 24-12. Wake-up from Stop Mode

SLF_WAK	WAK_MSK	MCU Clocks Enabled	Wake-up Interrupt Generated
0	0	No	No
0	1	No	No
1	0	No	No
1	1	Yes	Yes

24.4.10 Interrupts

The module can generate up to 70 interrupt sources (64 interrupts due to message buffers and 6 interrupts due to Ored interrupts from MBs, Bus Off, Error, Tx Warning, Rx Warning and Wake Up). The number of actual sources depends on the configured number of Message Buffers.

- Enhanced dual edge capture functionality
- The option to supply the source for each complementary PWM signal pair from any of the following:
  - external digital pin
  - internal timer channel
  - external ADC input, taking into account values set in ADC high and low limit registers.

25.1.3 Modes of operation

Care must be exercised when using this module in certain chip operating modes. Some motors (such 3-phase AC motors) require regular software updates for proper operation. Failure to do so could result in destroying the motor or inverter. Because of this, PWM outputs are placed in their inactive states in STOP mode, and optionally under WAIT and debug modes. PWM outputs will be reactivated (assuming they were active to begin with) when these modes are exited.

Table 25-1. Modes when PWM operation is restricted

Mode	Description
STOP	Peripheral and CPU clocks are stopped. PWM outputs are driven inactive.
WAIT	CPU clocks are stopped while peripheral clocks continue to run. PWM outputs are driven inactive as a function of the WAITEN bit.
DEBUG	CPU and peripheral clocks continue to run, but CPU maybe stalled for periods of time. PWM outputs are driven inactive as a function of the DBGEN bit.

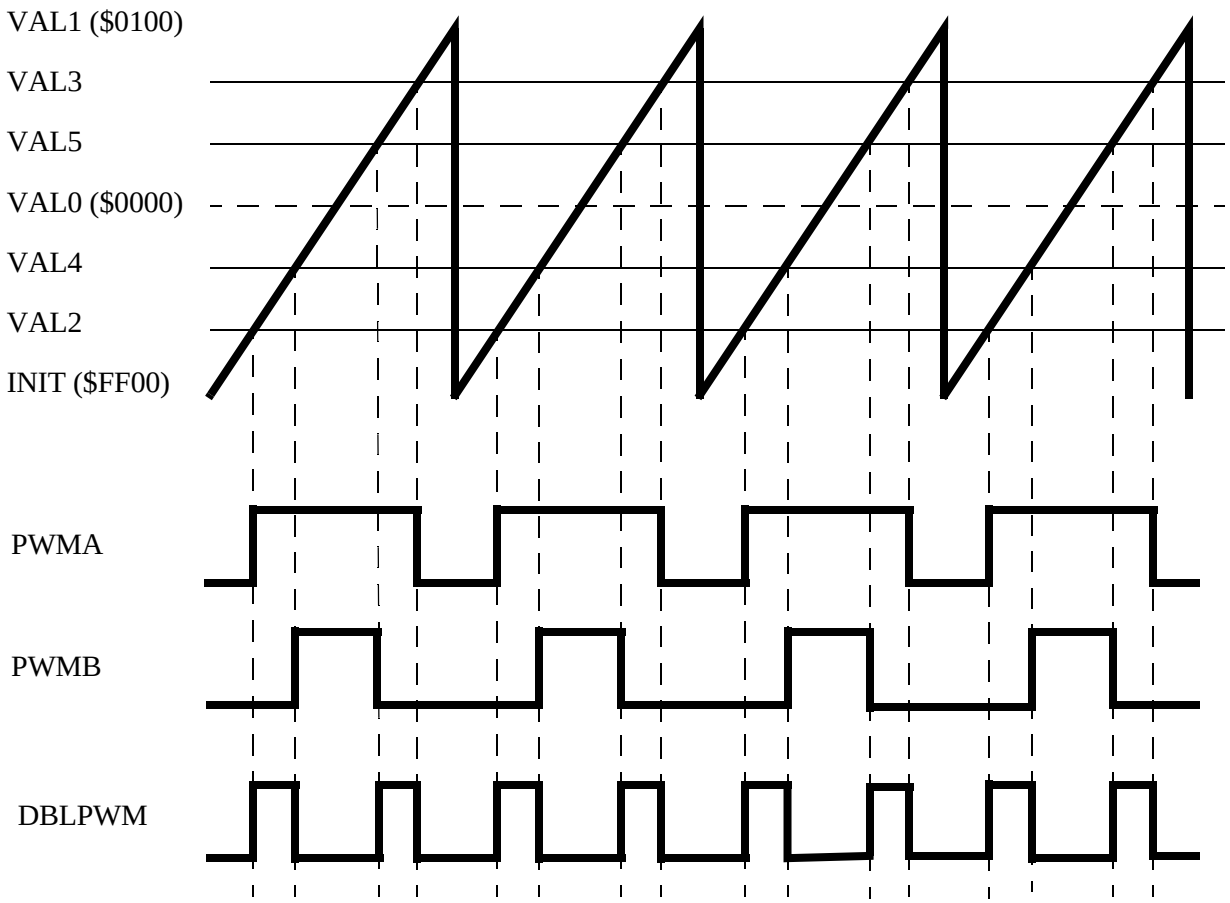


Figure 25-7. Double Switching Output Example

25.3.2.5 ADC Triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. Figure 25-8 shows how this is accomplished. When specifying complimentary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators are free to perform other functions. In this example, the software doesn't have to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

25.3.3.3 Counter synchronization

Referring to Figure 25-15, the 16 bit counter will count up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Sync signal which is one of four possible sources used to cause the 16 bit counter to be initialized with INIT. If Local Sync is selected as the counter initialization signal, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level.

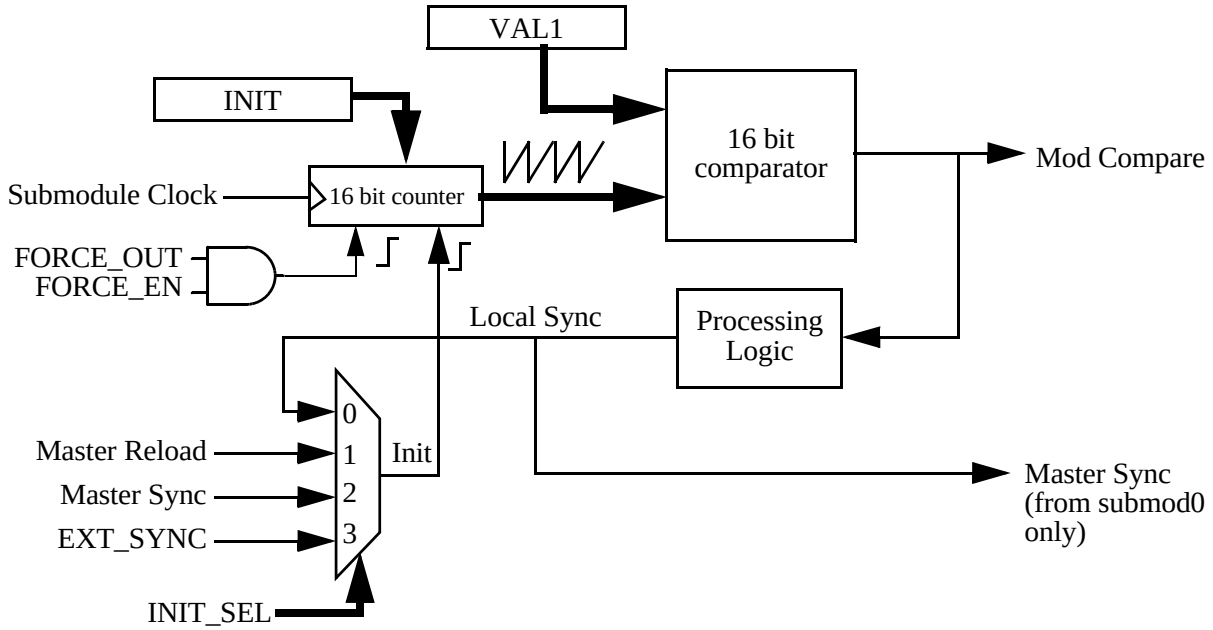


Figure 25-15. Submodule timer synchronization

The Master Sync signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0. The VAL1 register and associated comparator of the other submodules can then be freed up for other functions such as PWM generation, input captures, output compares, or output triggers.

The EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is usually commensurate to the sampling frequency of the software control algorithm, the submodule counter period will therefore equal the sampling period. As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that the FORCE_EN bit is set. As indicated by Figure 25-15, this constitutes a second init input into the counter which will cause the counter to initialize regardless of which signal is selected as the counter init signal.

26.5.2.75 ECC Error Injection Address Register (FR_EEIAR)

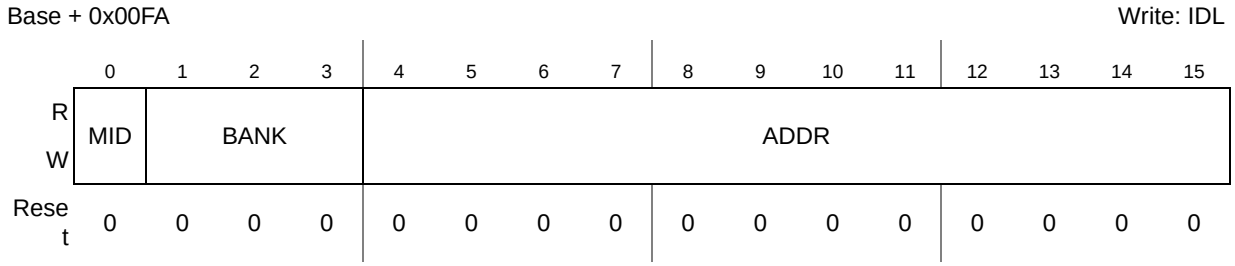


Figure 26-106. ECC Error Injection Address Register (FR_EEIAR)

This register defines the memory module, bank, and address where the ECC error has to be injected.

Table 26-86. FR_EEIAR Field Descriptions

Field	Description
MID	**Memory Identifier** — This flag defines the memory instance for ECC error injection. 0 PE DRAM 1 CHI LRAM
BANK	**Memory Bank** — This field defines the memory bank for ECC error injection. For MID=0: 000 BANK0: PE DRAM [7:0] 001 BANK1: PE DRAM [15:8] others reserved For MID=1: 000 BANK0: FR_MBCCFR(2n) 001 BANK1: FR_MBFIDR(2n) 010 BANK2: FR_MBIDXR(2n) 011 BANK3: FR_MBCCFR(2n+1) 100 BANK4: FR_MBFIDR(2n+1) 101 BANK5: FR_MBIDXR(2n+1) others reserved
ADDR	**Memory Address** — This flag defines the memory address for ECC error injection.

26.5.2.76 ECC Error Injection Data Register (FR_EEIDR)

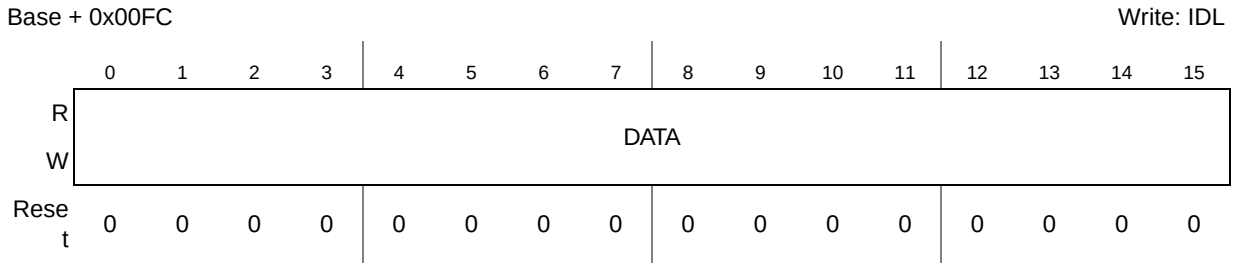


Figure 26-107. ECC Error Injection Data Register (FR_EEIDR)

This register defines the data distortion pattern for the error injection write. The number of valid bits depends on the selected memory and memory bank as shown in Table 26-84.

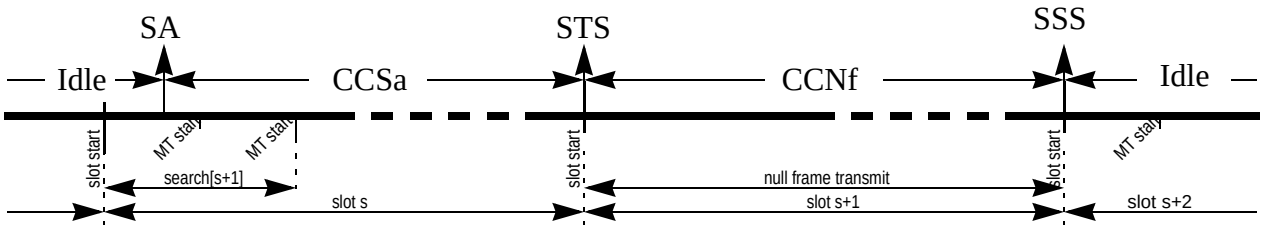


Figure 26-133. Null Frame Transmission from Idle state

A message buffer timing and state change diagram for null frame transmission from HLck state is given in Figure 26-134.

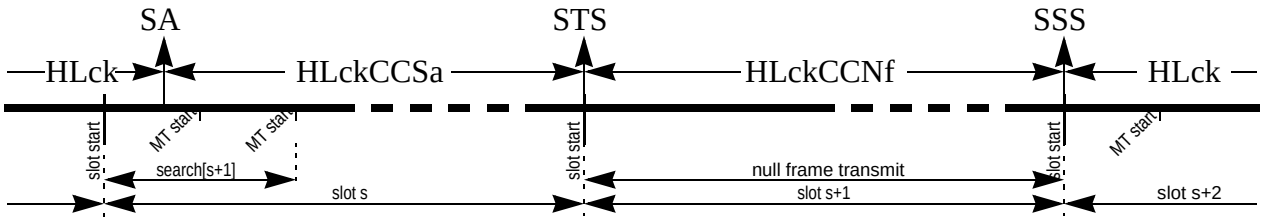


Figure 26-134. Null Frame Transmission from HLck state

If a transmit message buffer is in the CCSa or HLckCCSa state at the start of the transmission slot, a null frame is transmitted in any case, even if the message buffer is unlocked or committed before the transmission slot starts. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in Figure 26-135.

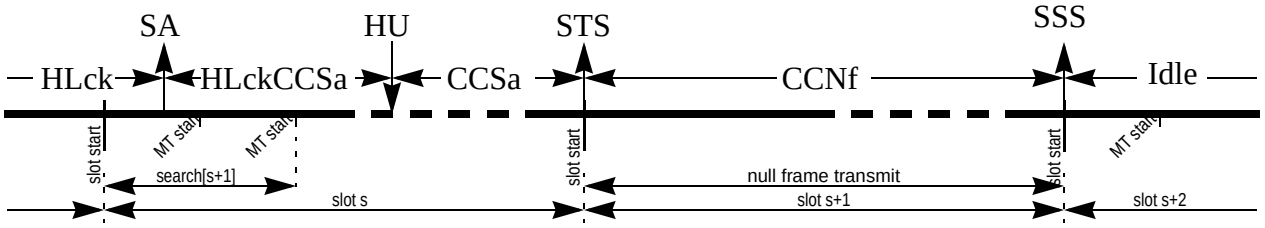
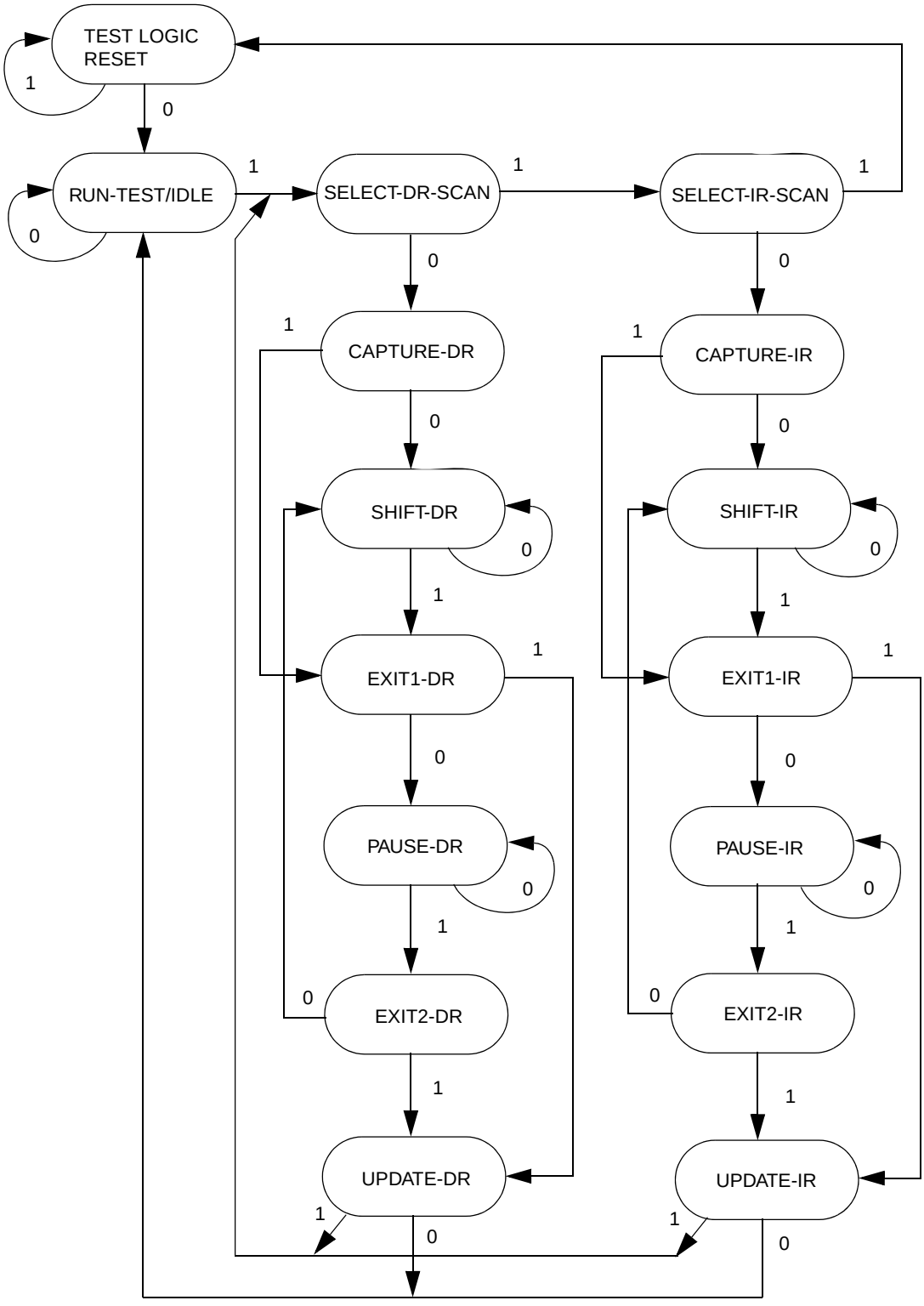


Figure 26-135. Null Frame Transmission from HLck state with unlock

Since the null frame transmission will not use the message buffer data, the application can lock/unlock the message buffer during null frame transmission. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in Figure 26-136.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 29-7. IEEE 1149.1-2001 TAP Controller Finite State Machine

Table 30-10. MPU_RGDAACn field descriptions (continued)

Field	Description
M1PE	Bus master 1 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
M1SM	Bus master 1 supervisor mode access control. This 2-bit field defines the access controls for bus master 1 when operating in supervisor mode. The M1SM field is defined as: 0b00 `r, w, x` = read, write and execute allowed 0b01 `r, –, x` = read and execute allowed, but no write 0b10 `r, w, –` = read and write allowed, but no execute 0b11 Same access controls as that defined by M1UM for user mode
M1UM	Bus master 1 user mode access control. This 3-bit field defines the access controls for bus master 1 when operating in user mode. The M1UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.
M0PE	Bus master 0 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
M0SM	Bus master 0 supervisor mode access control. This 2-bit field defines the access controls for bus master 0 when operating in supervisor mode. The M0SM field is defined as: 0b00 `r, w, x` = read, write and execute allowed 0b01 `r, –, x` = read and execute allowed, but no write 0b10 `r, w, –` = read and write allowed, but no execute 0b11 Same access controls as that defined by M0UM for user mode
M0UM	Bus master 0 user mode access control. This 3-bit field defines the access controls for bus master 0 when operating in user mode. The M0UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.

30.7 Functional description

In this section, the functional operation of the MPU is detailed. In particular, subsequent sections discuss the operation of the access evaluation macro as well as the handling of error-terminated AHB bus cycles.

30.7.1 Access evaluation macro

As previously discussed, the basic operation of the MPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in Figure 30-10, the access evaluation macro inputs the AHB system bus address phase signals (AHB_ap) and the contents of a region descriptor (RGDn) and performs two major functions: region hit determination (*hit_b*) and detection of an access protection violation (*error*).

Offset: 0x94 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CTO											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-40. UART current timeout register (UARTCTO)

Table 31-38. UARTCTO field descriptions

Field	Description
CTO	Current value of the timeout counter This field is reset whenever one of the following occurs: • A new value is written to the UARTPTO register • The value of this field matches the value of UARTPTO[PTO] • A hard or soft reset occurs • New incoming data is received When CTO matches the value of UARTPTO[PTO], UARTSR[TO] is set.

31.10.24 DMA Tx enable register (DMATXE)

This register enables the DMA Tx interface.

Offset: 0x98 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTE 15	DTE 14	DTE 13	DTE 12	DTE 11	DTE 10	DTE 9	DTE 8	DTE 7	DTE 6	DTE 5	DTE 4	DTE 3	DTE 2	DTE 1	DTE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31-41. DMA Tx enable register (DMATXE)

CAUTION

It is illegal to switch the flash from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

32.4.3.8 Pad Outputs-On

On completion of the step, if the PDO bit of the ME_<*target mode*>_MC register is cleared, then

- all pad outputs are enabled to return to their previous state
- the I/O pads power sequence driver is switched on

32.4.3.9 Peripheral Clocks Enable

Based on the current and target device modes, the peripheral configuration registers ME_RUN_PC0…7, ME_LP_PC0…7, and the peripheral control registers ME_PCTL0…143, the MC_ME enables the clocks for selected modules as required. This step is executed only after the process is completed.

32.4.3.10 Processor and Memory Clock Enable

If the mode transition is from any of the low-power modes HALT0 or STOP0 to RUN0…3, the clocks to the processor and system memory are enabled. The process of enabling these clocks is executed only after the Flash Module Switch-On process is completed.

32.4.3.11 Processor Low-Power Mode Exit

If the mode transition is from any of the low-power modes HALT0 orSTOP0 to RUN0…3, the MC_ME requests the processor to exit from its halted or stopped state. This step is executed only after the Processor and Memory Clock Enable process is completed.

32.4.3.12 System clock switching

Based on the SYSCLK bit field of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz int. RC osc. takes effect only after the S_IRCOSC bit of the register is set by hardware (i.e. the 16 MHz internal RC oscillator has stabilized).
- For cut2/3: The target clock configuration for the 4-40 MHz crystal osc. takes effect only after the S_XOSC bit of the register is set by hardware (i.e., the 4-40 MHz crystal oscillator has stabilized).
- The target clock configuration for the system FMPLL takes effect only after the S_PLL0 bit of the register is set by hardware (i.e. the system FMPLL has stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with “1111”. This is possible only in theTEST mode.

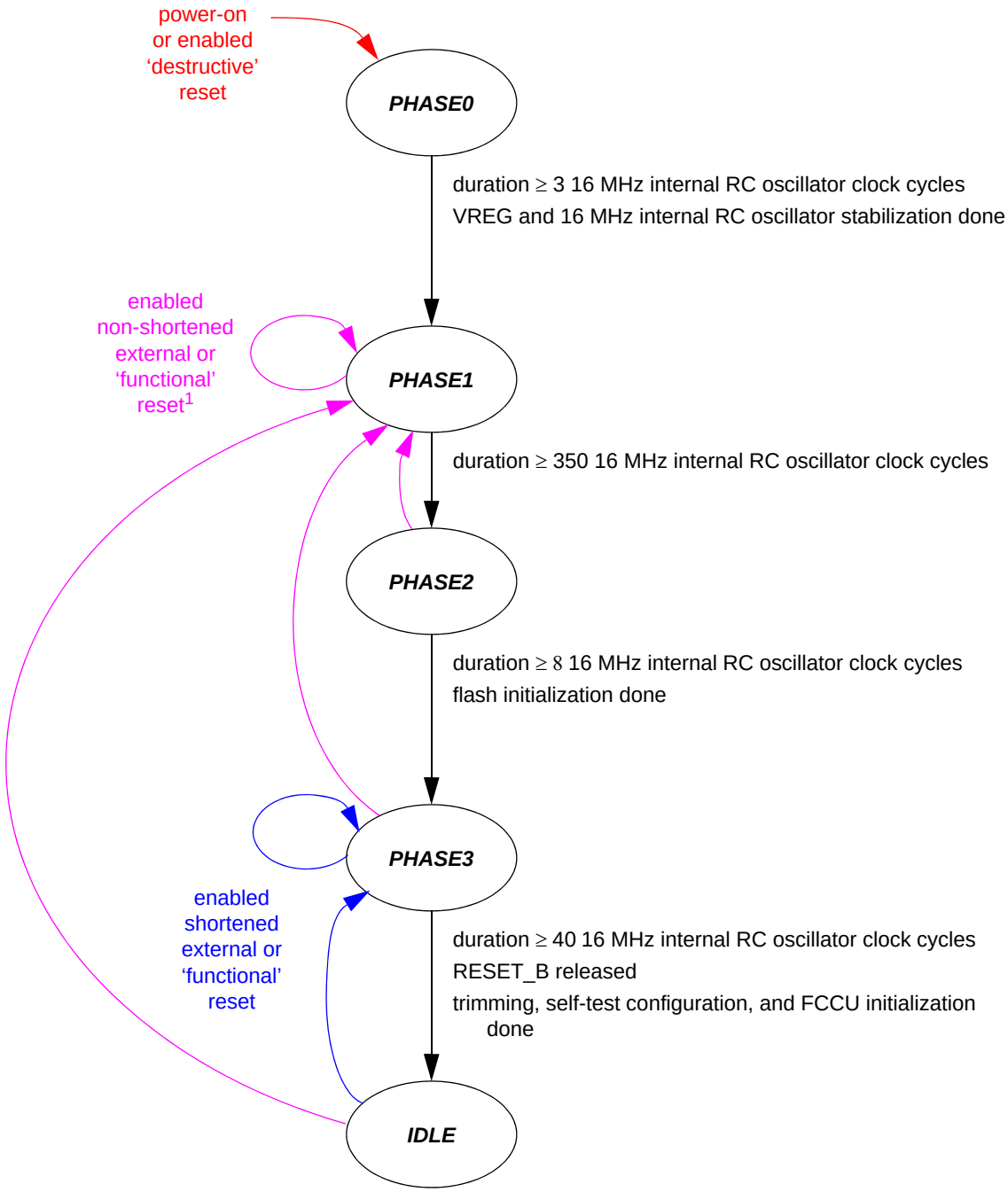


Figure 41-11. MC_RGM State Machine

41.4.1.1 PHASE0 Phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits ***PHASE0*** and enters ***PHASE1*** on verification of the following:

- all enabled 'destructive' resets have been processed
- all processes that need to be done in ***PHASE0*** are completed