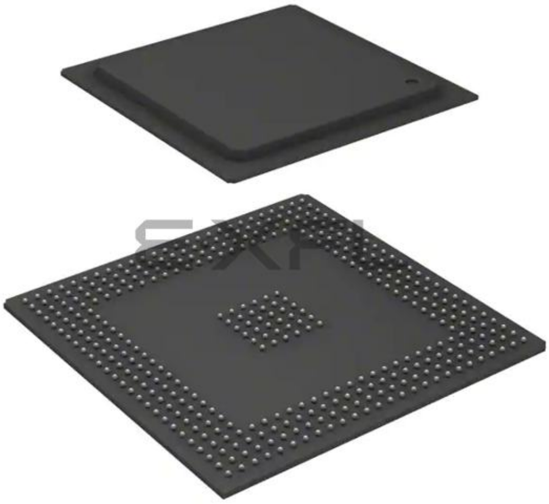
NXP USA Inc. - MPC563MVR56R2 Datasheet



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	PowerPC
Core Size	32-Bit Single-Core
Speed	56MHz
Connectivity	CANbus, EBI/EMI, SCI, SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	56
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 2.7V
Data Converters	A/D 32x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	388-BBGA
Supplier Device Package	388-PBGA (27x27)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mpc563mvr56r2

Figures

Figure Number	Title	Page Number

floating-point value does so using the double-precision floating-point format. Therefore, all floating-point numbers are stored in double-precision format.

All floating-point arithmetic instructions operate on data located in FPRs and, with the exception of the compare instructions (which update the CR), place the result into an FPR. Information about the status of floating-point operations is placed into the floating-point status and control register (FPSCR) and in some cases, after the completion of the operation’s writeback stage, into the CR. For information on how the CR is affected by floating-point operations, see Section 3.7.4, “Condition Register (CR).”

	MSB 0	LSB 63
	FPR0	
	FPR1	
	. . .	
	. . .	
	FPR31	
Reset	Unchanged	

Figure 3-5. Floating-Point Registers (FPRs)

3.7.3 Floating-Point Status and Control Register (FPSCR)

The FPSCR controls the handling of floating-point exceptions and records status resulting from the floating-point operations. FPSCR[0:23] are status bits. FPSCR[24:31] are control bits.

FPSCR[0:12] and FPSCR[21:23] are floating-point exception condition bits. These bits are sticky, except for the floating-point enabled exception summary (FEX) and floating-point invalid operation exception summary (VX). Once set, sticky bits remain set until they are cleared by an mcrfs, mtfsfi, mtfsf, or mtfsb0 instruction.

Table 3-4 summarizes which bits in the FPSCR are sticky status bits, which are normal status bits, and which are control bits.

Table 3-4. FPSCR Bit Categories

Bits	Type
[0], [3:12], [21:23]	Status, sticky
[1:2], [13:20]	Status, not sticky
[24:31]	Control

FEX and VX are the logical ORs of other FPSCR bits. Therefore these two bits are not listed among the FPSCR bits directly affected by the various instructions.

Table 3-34. Register Settings following a Software Emulation Exception

Register Name	Bits	Description
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	DCMPEN	This bit is set according to (BBCMCR[EN_COMP] AND BBCMCR[EXC_COMP])
	Other	Cleared to 0

[1] If the exception occurs during an instruction fetch in Decompression On mode, the SRR0 register will contain a compressed address.

Execution resumes at offset 0x01000 from the base address indicated by MSR[IP].

3.15.4.14 Implementation-Dependent Instruction Protection Exception (0x1300)

The implementation-specific instruction storage protection error interrupt occurs in the following cases:

- The fetch access violates storage protection and MSR[IR] = 1.
- The fetch access is to guarded storage and MSR[IR] = 1.

The register settings for instruction protection exceptions are shown in Table 3-35.

Table 3-35. Register Settings following an Instruction Protection Exception

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)[1]	All	Set to the effective address of the instruction that caused the exception
Save/Restore Register 1 (SRR1)	0:2	Cleared to 0
	3	Set to 1 if the fetch access was to a guarded storage when MSR[IR] = 1, otherwise clear to 0
	4	Set to 1 if the storage access is not permitted by the protection mechanism (IMPU in BBC) and MSR[IR] = 1; otherwise clear to 0
	5:15	Cleared to 0
	16:31	Loaded from bits [16:31] of MSR. In the current implementation, bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR[IR]
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	DCMPEN	This bit is set according to (BBCMCR[EN_COMP] AND BBCMCR[EXC_COMP])
	Other	Cleared to 0

- Implements a parked master on the U-bus, resulting in zero clock delays for RCPU fetch accesses to the U-bus
- Fully utilizes the U-bus pipeline for fetch accesses
- Avoids undesirable delays through a tight interface with the L2U module (fully utilizing U-bus bandwidth and back-to-back accesses)
- Supports program trace and show cycles
- Supports a special attribute for debug port fetch accesses.

4.1.2 IMPU Key Features

- There are four regions in which the base address and size can be programmed.
- Available region sizes include 2 Kbytes, 8 Kbytes, 16 Kbytes, 32 Kbytes, 64 Kbytes, 128 Kbytes, 256 Kbytes, 512 Kbytes, 1 Mbyte, 2 Mbytes, 4 Mbytes, 8 Mbytes, 16 Mbytes....4 Gbytes.
- Overlap between regions is allowed.
- Each of the four regions supports the following attributes:
  - User/supervisor
  - Guard attribute (causes an interrupt in case of speculative fetch attempt)
  - Compressed/non-compressed (MPC562/MPC564 only)
  - Regions are enabled or disabled in software.
- Global region entry declares the default access attributes for all memory areas not covered by the four regions:
- The RCPU gets the instruction storage protection exception generated upon
  - An access violation of protection attributes
  - A fetch from a guarded region.
- The RCPU MSR[IR] bit controls IMPU protection.
- Programming is performed by using the RCPU mtspr/mfspr instructions to/from implementation specific special-purpose registers.
- The IMPU supplies relocation addresses of all the exceptions within the internal memory space.
- The IMPU implements external interrupt vector splitting to reduce the external interrupt latency.
- There is a special reset exception vector for decompression on mode (MPC562/MPC564 only).

4.1.3 ICDU Key Features

The following are instruction code decompression unit key features of the MPC562/MPC564. See Appendix A, “MPC562/MPC564 Compression Features” for more information.

- Instruction code on-line decompression based on “instruction classes” algorithm.
- No need for address translation between compressed and non-compressed address spaces — ICDU provides “next instruction address” to the RCPU
- In most cases, instruction decompression takes one clock
- Code decompression is pipelined:

10.9.2 Memory Controller Status Registers (MSTAT)

	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Field	—								WPER0	WPER1	WPER2	WPER3	—			
$\overline{\text{HRESET}}$	0000_0000_0000_0000															
Addr	0x2F C178															

Figure 10-22. Memory Controller Status Register (MSTAT)

Table 10-7. MSTAT Bit Descriptions

Bits	Name	Description
0:7	—	Reserved
8:11	WPER0 – WPER3	Write protection error for bank x. This bit is asserted when a write-protect error occurs for the associated memory bank. A bus monitor (responding to $\overline{\text{TEA}}$ assertion) will, if enabled, prompt the read of this register if $\overline{\text{TA}}$ is not asserted during a write cycle. WPERx is cleared by writing one to the bit or by performing a system reset. Writing a zero has no effect on WPER.
12:15	—	Reserved

10.9.3 Memory Controller Base Registers (BR0–BR3)

	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	BA															
$\overline{\text{HRESET}}$(BR0)	Unchanged															
$\overline{\text{HRESET}}$(BR[1:3])	Unchanged															
Addr	0x2F C100 (BR0); 0x2F C108 (BR1); 0x2F C110 (BR2); 0x2F C118 (BR3)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
	BA	AT			PS		SST	WP	—	BL	WEBS	TBDIP	LBDIP	SETA	BI	V
$\overline{\text{HRESET}}$(BR0)	Unchanged				ID[4:5]		00		Undefined		0	Undefined			1	$\overline{\text{ID3}}$
$\overline{\text{HRESET}}$(BR[1:3])	Unchanged															X[2]

[1] The reset value is determined by the value on the internal data bus during reset (reset-configuration word).

[2] See Table 10-9 for reset value.

Figure 10-23. Memory Controller Base Registers 0–3 (BR0–BR3)

- The “on” resistance of the internal switches is 0 Ω and the “off” resistance is infinite

13.7.5.1 Analog Input Considerations

The source impedance of the analog signal to be measured and any intermediate filtering should be considered whether external multiplexing is used or not. Figure 13-53 shows the connection of eight typical analog signal sources to one QADC64E analog input signal through a separate multiplexer chip. Also, an example of an analog signal source connected directly to a QADC64E analog input channel is displayed.

$$f_{PWM} = \frac{f_{SYS}}{N_{MCPSM} \bullet N_{COUNTER} \bullet N_{MDASM}}$$

where:

- N_{MCPSM} is the overall MCPSM clock divide ratio (2, 3, 4,...,16).
- $N_{COUNTER}$ is the divide ratio of the prescaler of the counter (used as a time reference) that drives the 16-bit counter bus.
- N_{MDASM} is the maximum count reachable by the counter when using n bits of resolution (this count is equal to 2^n).

A few examples of frequencies and resolutions that can be obtained are shown in Table 17-17.

Table 17-17. MDASM PWM Example Output Frequencies/Resolutions at f_{SYS} = 40 MHz

Resolution (bits)	N_{MCPSM}	$N_{COUNTER}$	N_{MDASM}	PWM output frequency (Hz)[1]
16	16	256	65536	0.15
16	2	1	65536	305.17
15	16	256	32768	0.29
15	2	1	32768	610.35
14	16	256	16384	0.59
14	2	1	16384	1 220.70
13	16	256	8192	1.19
13	2	1	8192	2 441.41
12	16	256	4096	2.38
12	2	1	4096	4 882.81
11	16	256	2048	4.77
11	2	1	2048	9 765.63
9	16	256	512	19.07
9	2	1	512	39 062.50
7	16	256	128	76.29
7	2	1	128	156 250

[1] This information is valid only if the MDASM is connected to an MMCSM operating as a free-running counter.

When using 16 bits of resolution on the comparator (MODE[2:0] = 0b000), the output can vary from a 0% duty cycle up to a duty cycle of 65535/65536. In this case it is not possible to have a 100% duty cycle. In cases where 16-bit resolution is not needed, it is possible to have a duty cycle ranging from 0% to 100%. Setting bit 15 of the value stored in register B to one results in the output being ‘always set’. Clearing bit 15 (to zero) allows normal comparisons to occur and the normal output waveform is obtained. Changes to and from the 100% duty cycle are done synchronously on an A or B match, as are all other width changes.

Figure 17-44 shows a counter submodule and a DASM combination as an example of period measurement. The software designates whether the rising or falling edge of the input signal is to be used for the measurements. When the edge is detected, the state of the 16-bit counter bus is stored in register A and the content of register B1 is transferred to register B2. After register B2 is safely latched, the content of register A is transferred to register B1. This procedure gives the software coherent current and previous samples in registers A and B2 at all times. An interrupt is available for the cases where the software needs to be aware of each new sample. Note that a software option is provided to also generate an interrupt after the first edge.

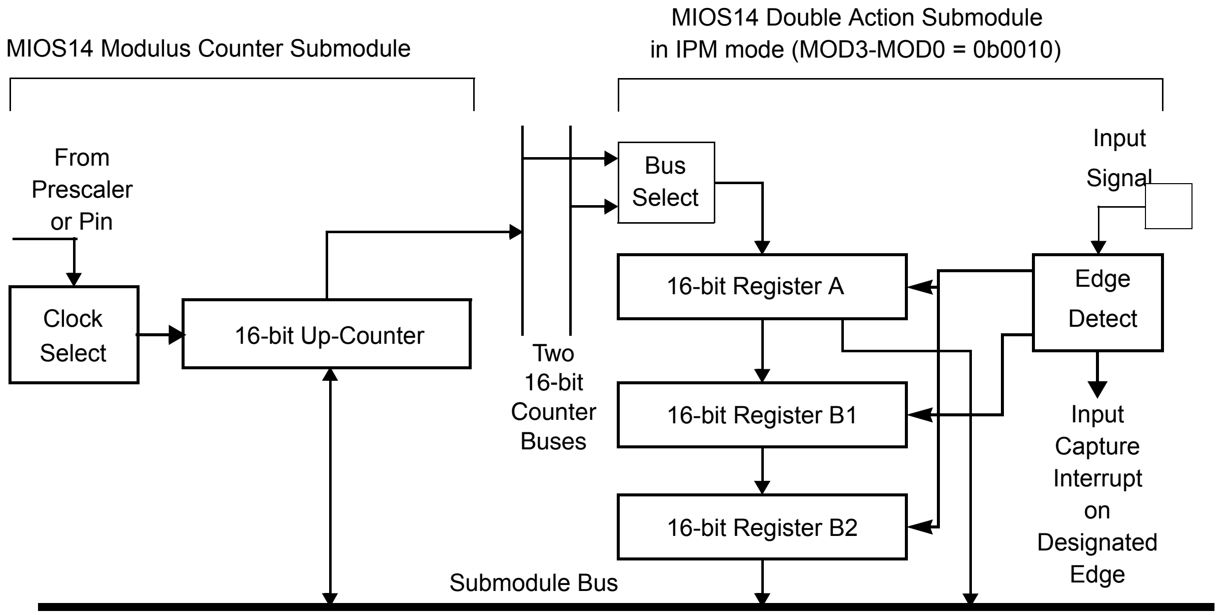


Figure 17-44. MIOS14 Example: Double Capture Period Measurement

17.13.3 MIOS14 Double Edge Single Output Pulse Generation

Software can initialize the MIOS14 to generate both the rising and the falling edge of an output pulse. With a MDASM, pulses as narrow as 50 ns can be generated since software action is not needed between the edges. Pulses as long as 2.1 s can be generated. When an interrupt is desired, it can be selected to occur on every edge or only after the second edge.

Figure 17-45 shows how a counter submodule and a MDASM can be used to generate both edges of a single output pulse. The software puts the compare value for one edge in register A and the other one in register B2. The MDASM automatically creates both edges and the pulse can be selected by software to be a high-going or a low-going. After the trailing edge, the MDASM stops to await further commands from the software. Note that a single edge output can be generated by writing to only one register.

Table 18-8. SHORT_REG[SH_TCAN] Bit Settings

SH_TCAN2	SH_TCAN1	SH_TCAN0	Effects On TouCAN Modules
1	1	0	TouCAN_B[B_CNRX0, B_CNTX0] shorted to TouCAN_A[A_CNRX0, A_CNTX0]. Both modules communicate via A_CNTX0, A_CNRX0. See Figure 18-24 for example of this bit setting
1	1	1	TouCAN_B[B_CNRX0, B_CNTX0] and TouCAN_C[C_CNRX0, C_CNTX0] shorted to TouCAN_A[A_CNRX0, A_CNTX0] All modules communicate via A_CNTX0, A_CNRX0.

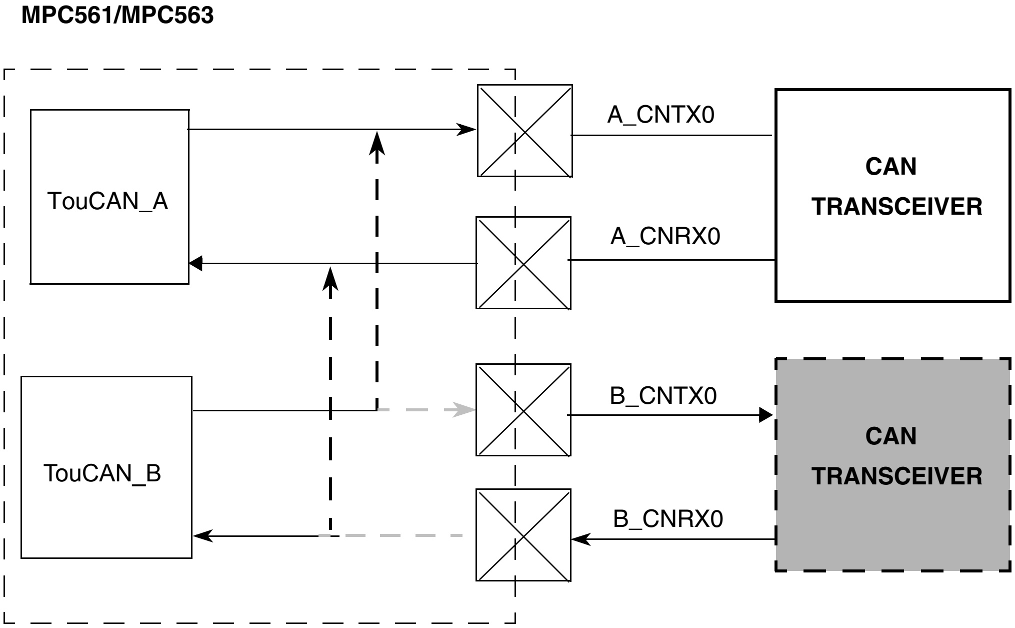


Figure 18-24. Example of TouCAN Internal Short with SH_TCAN = 0b110

Table 18-9. SHORT_REG[SH_TPU] Bit Settings

SH_TPU0	A_TPUCH0	B_TPUCH0	Effect on TPU3 Modules
1	Input	Input	Data on pad A_TPUCH0 will be the input to A_TPUCH0 and B_TPUCH0
1	Input	Output	Output data on B_TPUCH0 will be the input to A_TPUCH0
1	Output	Input	Output on A_TPUCH0 will be the input data to B_TPUCH0
1	Output	Output	No Short
0	X	X	No Short

22.5.3 CALRAM Overlay Configuration Register (CRAM_OVLCR)

	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	OVL	DERR	CLPS	—												
SRESET	0000_0000_0000_0000															
Addr	0x38 0028															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Field	—															
SRESET	0000_0000_0000_0000															

Figure 22-11. CALRAM Overlay Configuration Register (CRAM_OVLCR)

Table 22-7. CRAMOVLCR Bit Descriptions

Bits	Name	Description
0	OVL	Overlay enable — When set, the CALRAM overlay mode operation is enabled. In this mode CALRAM allows eight programmable sections (four to 512 bytes) of the on-chip U-bus Flash memory module to be overlaid by sections of the CALRAM. 0 CALRAM module overlay is disabled 1 CALRAM module overlay is enabled
1	DERR	Data error 0 CALRAM module will not generate machine check exception due to normal L-bus array access to the enabled portion overlay region even if overlay is enabled 1 CALRAM module will generate machine check exception due to normal L-bus array access to the enabled portion of overlay region even if overlay is enabled
2	CLPS	Collapse the total overlay region from 4 Kbytes to 32 bytes; that is, the size is forced to be four bytes for each for the eight regions regardless of the values programmed in CRAM_RBAx[0:3]; these bits are also referred to as RGN_SIZE[0:3]. 0 Overlay region of four Kbytes; region size as specified by CRAM_RBAx[0:3]. 1 Overlay region of 32 bytes; each region size is four bytes long regardless of the values in CRAM_RBAx[0:3].
3:31	—	Reserved

22.5.4 CALRAM Ownership Trace Register (CRAM_OTR)

This register is provided to support a separate module called READI. Refer to Chapter 24, “READI Module.”

The reads from this register will return 0’s.

NOTE

CRAM_OTR is also defined as READI_OTR. See Section 24.6.1.1, “User-Mapped Register (OTR).”

- On the fly trap enable programming of the different internal breakpoints using the serial interface of the development port (refer to Section 23.4, “Development Port”). Software control is also available.
- Watchpoints do not change the timing of the machine
- Internal breakpoints and watchpoints are detected on the instruction during instruction fetch
- Internal breakpoints and watchpoints are detected on the load/store during load/store bus cycles
- Both instruction and load/store breakpoints and watchpoints are handled and reported on retirement. Breakpoints and watchpoints on recovered instructions (as a result of exceptions, interrupts or miss prediction) are not reported and do not change the timing of the machine.
- Instructions with instruction breakpoints are not executed. The machine branches to the breakpoint exception routine BEFORE it executes the instruction.
- Instructions with load/store breakpoints are executed. The machine branches to the breakpoint exception routine AFTER it executes the instruction. The address of the access is placed in the BAR (breakpoint address register).
- Load/store multiple and string instructions with load/store breakpoints first finish execution (all of it) and then the machine branches to the breakpoint exception routine.
- Load/store data compare is done on the load/store, AFTER swap in store accesses and BEFORE swap in load accesses (as the data appears on the bus).
- Internal breakpoints may operate either in masked mode or in non-masked mode.
- Both “go to x” and “continue” working modes are supported for the instruction breakpoints.

23.2.1.1 Restrictions

There are cases when the same watchpoint can be detected more than once during the execution of a single instruction, e.g. a load/store watchpoint is detected on more than one transfer when executing a load/store multiple/string or a load/store watchpoint is detected on more than one byte when working in byte mode. In all these cases only one watchpoint of the same type is reported for a single instruction. Similarly, only one watchpoint of the same type can be counted in the counters for a single instruction.

Because watchpoint events are reported upon the retirement of the instruction that caused the event, and more than one instruction can retire from the machine in one clock, consequent events may be reported in the same clock. Moreover the same event, if detected on more than one instruction (e.g., tight loops, range detection), in some cases will be reported only once. Note that the internal counters count correctly in these cases.

Do not put a breakpoint on an mtspr instruction that accesses the ICTRL register when ICTRL[IFM] = 1 because this causes unpredictable behavior.

23.2.1.2 Byte and Half-Word Working Modes

The CPU watchpoints and breakpoints support enables detection of matches on bytes and half-words even when accessed using a load/store instruction of larger data widths, for example when loading a table of bytes using a series of load word instructions. In order to use this feature, program the byte mask for each of the L-data comparators and to write the needed match value to the correct half-word of the data instruction. Therefore, a valid data status will be output and the interrupt status will be saved for the next transmission.

The sequencing error encoding indicates that the inputs from the external development tool are not what the development port and/or the CPU was expecting. Two cases could cause this error:

1. The processor was trying to read instructions and there was data shifted into the development port, or
2. The processor was trying to read data and there was instruction shifted into the development port. The port will terminate the read cycle with a bus error.

This bus error will cause the CPU to signal that an interrupt (exception) occurred. Since a status of sequencing error has a higher priority than exception, the port will report the sequencing error first, and the CPU interrupt on the next transmission. The development port will ignore the command, instruction, or data shifted in while the sequencing error or CPU interrupt is shifted out. The next transmission after all error status is reported to the port should be a new instruction, trap enable or command (possibly the one that was in progress when the sequencing error occurred).

The interrupt-occurred encoding is used to indicate that the CPU encountered an interrupt during the execution of the previous instruction in debug mode. Interrupts may occur as the result of instruction execution (such as unimplemented opcode or arithmetic error), because of a memory access fault, or from an unmasked external interrupt. When an interrupt occurs the development port will ignore the command, instruction, or data shifted in while the interrupt encoding was shifting out. The next transmission to the port should be a new instruction, trap enable or debug port command.

Finally, the null encoding is used to indicate that no data has been transferred from the CPU to the development port shift register.

23.4.6.11 Fast Download Procedure

The download procedure is used to download a block of data from the debug tool into system memory. This procedure can be accomplished by repeating the following sequence of transactions from the development tool to the debug port for the number of data words to be down loaded:

```
INIT:    Save RX, RY
         RY <- Memory Block address- 4

         …

repeat:  mfspr    RX, DPDR
         DATA word to be moved to memory
         stwu     RX, 0x4(RY)
until here

         …

         Restore RX,RY
```

Figure 23-12. Download Procedure Code Example

24.6 Programming Model

The READI registers do not follow the recommendations of the IEEE-ISTO 5001 - 1999, but are loosely based on the 0.9 release of the standard. See http://www.nexus5001.org/.

READI registers are classified into two categories: user-mapped register and tool-mapped registers.

User-mapped register (a memory-mapped register):

- Ownership trace register

Tool-mapped registers (registers which can be accessed only through the development tool and are not memory mapped):

- Device ID register
- Development control register
- Mode control register 4-bit
- User base address register
- Read/write access register
- Upload/download information register
- Data trace attributes register 1
- Data trace attributes register 2

24.6.1 Register Map

READI registers are accessible via the auxiliary port. They can be classified into two categories: user-mapped registers and tool-mapped registers.

24.6.1.1 User-Mapped Register (OTR)

The operating system writes the ID for the current task/process in the single user-mapped register, the READI ownership trace (OTR) register. Table 24-4 shows the location of the register bits. Their functions are explained below.

The current task/process (CTP) field is updated by the operating system software to provide task/process ID information. The OTR register can only be accessed by supervisor data attributes. Only CPU writes to this register will be transmitted. This register is not accessible via the auxiliary port download request message.

NOTE

This is the only READI register that is reset by $\overline{\text{HRESET}}$.

To read control or status from memory-mapped locations the following sequence would be required.

1. The tool confirms that the device is ready. The tool transmits the download request public message (TCODE 18) which contains read attributes and target address.
2. When device reads data it transmits upload/download information message (TCODE 19) containing read data. Device is now ready for next access.

For a block write to memory-mapped locations the following sequence would be required.

1. The tool confirms that the device is ready. The tool transmits the download request public message (TCODE 18) which contains block write attributes, first write data, and target address.
2. The tool waits for device ready for upload/download message (TCODE 16). When it is transmitted by device, tool transmits upload/download information message (TCODE 19) containing next write data. This step is repeated until all data is written

For a block read from memory-mapped locations the following sequence would be required.

1. The tool confirms that the device is ready. The tool transmits the download request public message (TCODE 18) which contains block read attributes and target address.
2. The tool waits for upload/download information message (TCODE 19) from device, which contains read data. This step is repeated until all data is read.

Refer to Section 24.10, “Read/Write Access,” for more details on read/write access protocol.

24.6.3 Accessing READI Tool Mapped Registers Via the Auxiliary Port

To write control or status data to READI tool mapped registers the following sequence would be required.

1. The tool confirms that the device is ready. The tool transmits the download request message (TCODE 18) which contains write data, and register opcode.
2. The tool waits for device ready for upload/download message (TCODE 16) before initiating next access.

To read control or status from READI tool mapped registers the following sequence would be required

1. The tool confirms that the device is ready. The tool transmits the upload request message (TCODE 17) which contains the target opcode.
2. When device reads data it transmits upload/download information message (TCODE 19) containing read data. Device is now ready for next access.

Refer to Section 24.10, “Read/Write Access,” for more details on read/write access protocol.

24.6.4 Partial Register Updates

Registers may be updated via the auxiliary port using the download request message with the message containing only N (where N is less than register width) most-significant bits of the register. In such cases the bits not transmitted will be reset to 0b0. The bits transmitted will be aligned such that the last bit transmitted will be the most significant bit of the register. Therefore a message size that is divisible by the input port size should be transmitted.

Table 24-22. Message Field Sizes[1,2] (continued)

Message	TCODE	Field # 1	Field # 2	Field # 3	Max Size[3]	Min. Size[4]
Data Trace — Data Write Synchronization Message	13 (0xD)	Variable Max = 1 Min = 1	Variable Max = 25 Min = 1	Variable Max = 32 Min = 8	64 bits	16 bits
Data Trace — Data Read Synchronization Message	14 (0xE)	Variable Max = 1 Min = 1	Variable Max = 25 Min = 1	Variable Max = 32 Min = 8	64 bits	16 bits
Watchpoint Message	15 (0xF)	Fixed = 6	NA	NA	12 bits	12 bits
Auxiliary Access — Device Ready for Upload/Download Message	16 (0x10)	NA	NA	NA	6 bits	6 bits
Auxiliary Access — Upload Request (Tool requests information) Message	17 (0x11)	Fixed = 8	NA	NA	14 bits	14 bits
Auxiliary Access —Download Request (Tool provides Information) Message	18 (0x12)	Fixed = 8	Variable Max = 80 Min = 8	NA	94 bits	22 bits
Auxiliary Access — Upload/Download Information (Device/Tool provides Information) Message	19 (0x13)	Variable Max = 80 Min = 8	NA	NA	86 bits	14 bits
Resource Full Message[6]	27 (0x1B)	Variable Max = 4 Min =1	NA	NA	10 bits	7 bits
Dev Port Access — DSDI Data (Tool Provides Information) Message	56 (0x38)	Variable Max = 35 Min = 10	NA	NA	41 bits	16 bits
Dev Port Access — DSDO Data (Device Provides Information) Message	57 (0x39)	Variable Max = 35 Min = 10	NA	NA	41 bits	16 bits
Dev Port Access —BDM Status (Device Provides Information) Message	58 (0x3A)	Fixed = 1	NA	NA	7 bits	7 bits
Program Trace — Indirect Branch Message With Compressed Code	59 (0x3B)	Variable Max = 8 Min = 1	Fixed = 6	Variable Min = 1 Max = 23	43 bits	14 bits
Program Trace — Direct Branch Synchronization Message With Compressed Code (PTSM = 0) [7]	60 (0x3C)	Fixed = 6	Variable Max = 23 Min = 1	NA	35 bits	13 bits

[6 bits]	[8 bits]	[8-80 bits]
TCODE (18)	Opcode	Register Value

Max Length = 94 bits
Min Length = 22 bits

Figure 24-57. Write Register Message

[6 bits]	[8-80 bits]
TCODE (19)	Return Value

Max Length = 86 bits
Min Length = 14 bits

Figure 24-58. Read/Write Response Message

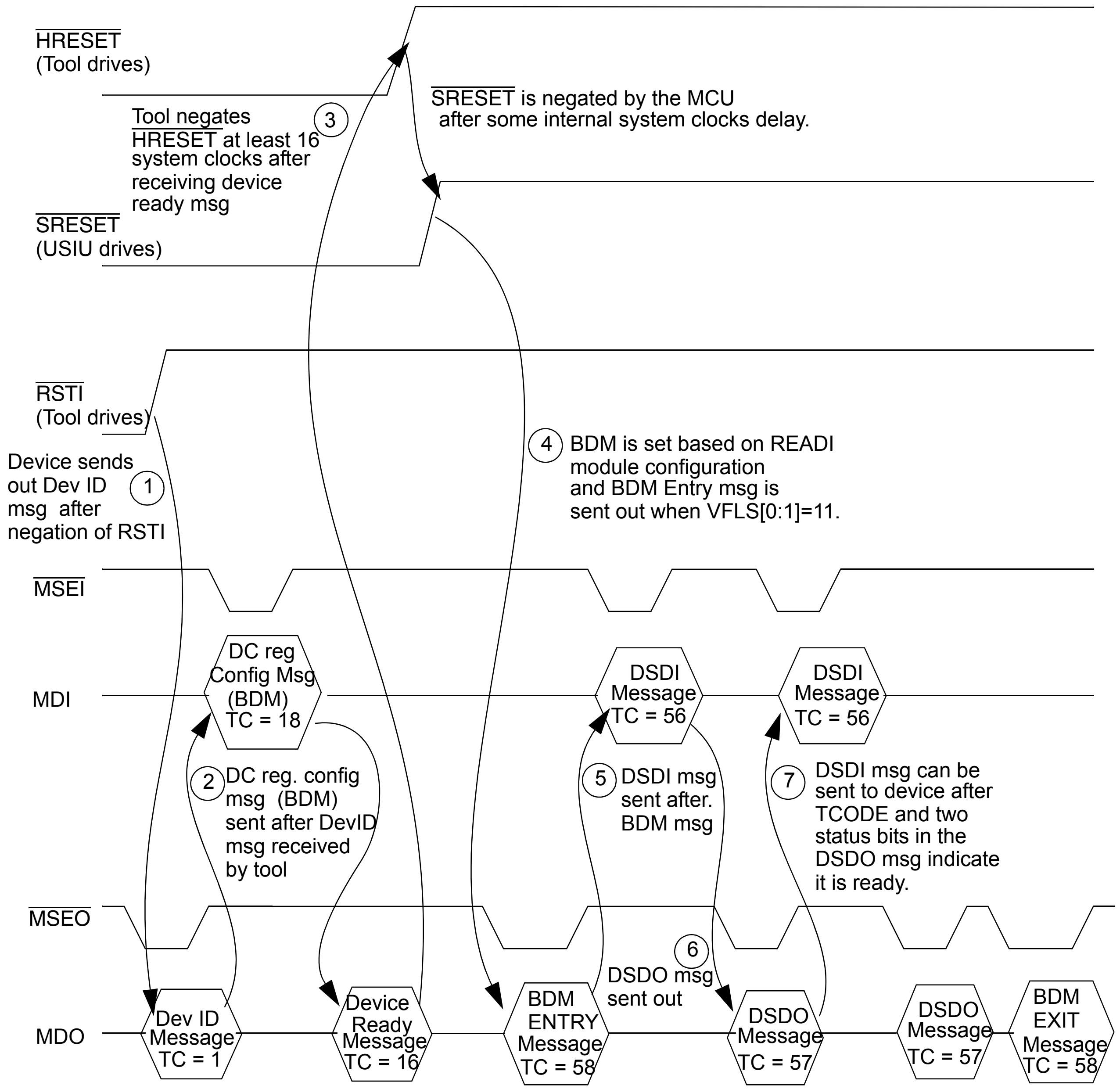


Figure 24-84. RCPU Development Access Timing Diagram — Debug Mode Entry Out-of-Reset

Figure 24-85 shows the transmission sequence of DSDI/DSDO data messages.

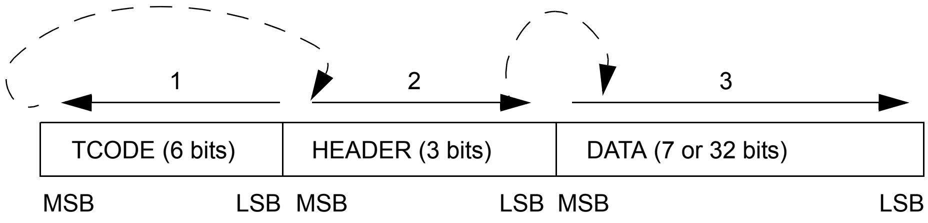
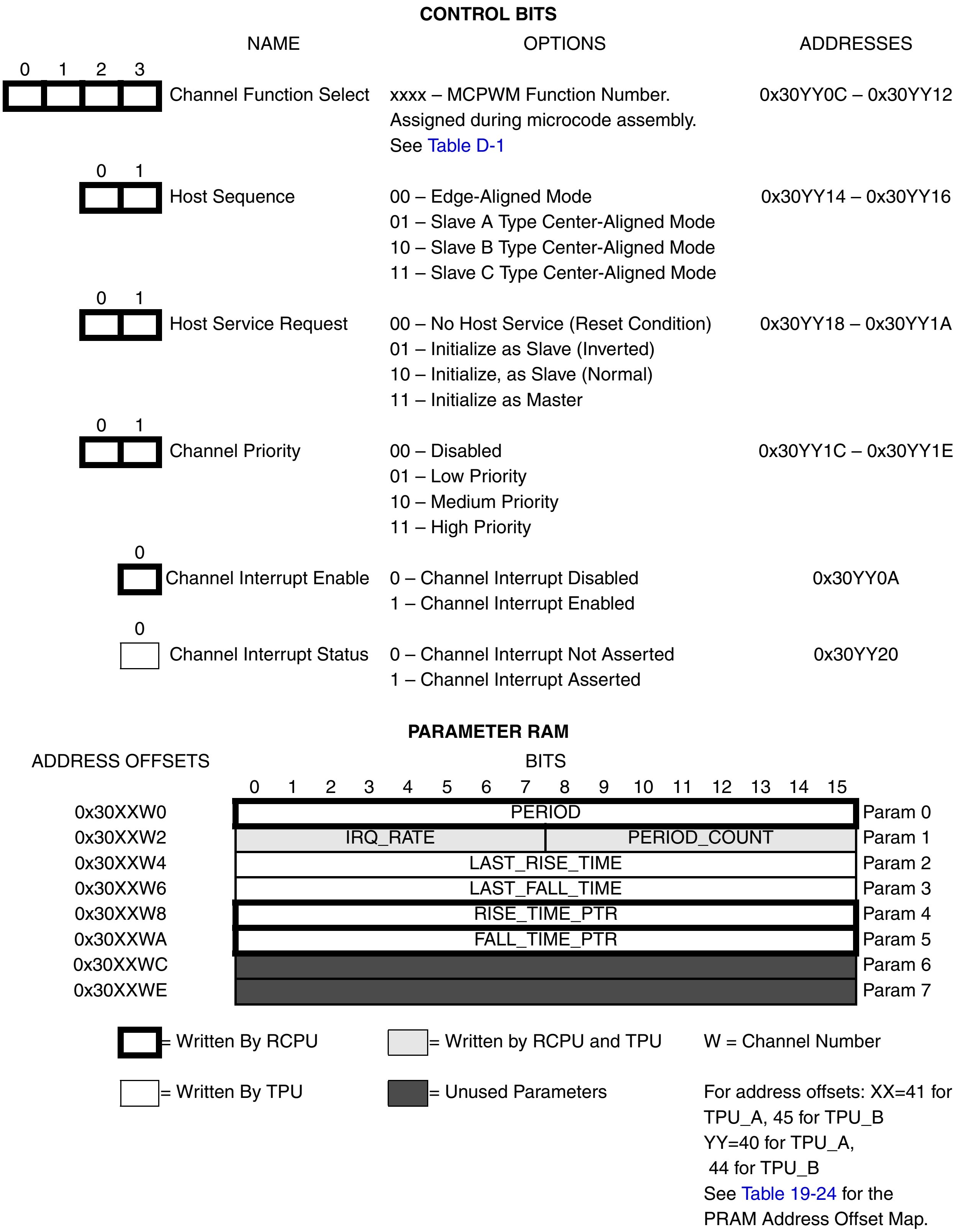


Figure 24-85. Transmission Sequence of DSDx Data Messages



CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0 1 2 3 ▣▣▣▣	Channel Function Select	xxxx – MCPWM Function Number. Assigned during microcode assembly. See Table D-1	0x30YY0C – 0x30YY12
0 1 ▣▣	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center-Aligned Mode 10 – Slave B Type Center-Aligned Mode 11 – Slave C Type Center-Aligned Mode	0x30YY14 – 0x30YY16
0 1 ▣▣	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize as Slave (Inverted) 10 – Initialize, as Slave (Normal) 11 – Initialize as Master	0x30YY18 – 0x30YY1A
0 1 ▣▣	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0 ▣	Channel Interrupt Enable	0 – Channel Interrupt Disabled 1 – Channel Interrupt Enabled	0x30YY0A
0 □	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

ADDRESS OFFSETS	BITS 0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	
0x30XXW0	▣ PERIOD		Param 0
0x30XXW2	▒ IRQ_RATE	▒ PERIOD_COUNT	Param 1
0x30XXW4	□ LAST_RISE_TIME		Param 2
0x30XXW6	□ LAST_FALL_TIME		Param 3
0x30XXW8	▣ RISE_TIME_PTR		Param 4
0x30XXWA	▣ FALL_TIME_PTR		Param 5
0x30XXWC	■	■	Param 6
0x30XXWE	■	■	Param 7

▣= Written By RCPU

▒= Written by RCPU and TPU

W = Channel Number

□= Written By TPU

■= Unused Parameters

For address offsets: XX=41 for TPU_A, 45 for TPU_B
YY=40 for TPU_A,
 44 for TPU_B
See Table 19-24 for the PRAM Address Offset Map.

Figure D-12. MCPWM Parameters — Master Mode

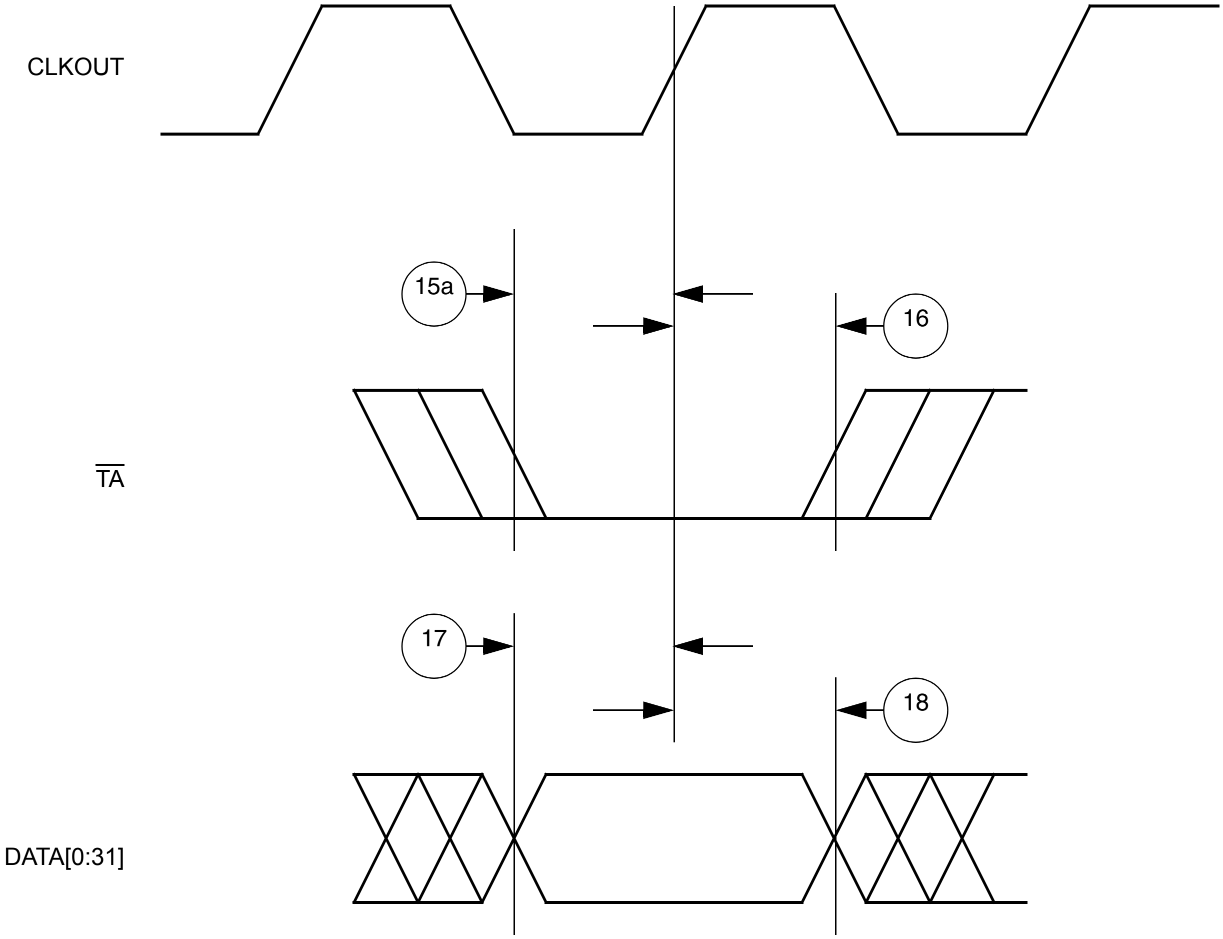


Figure F-15. Input Data Timing In Normal Case

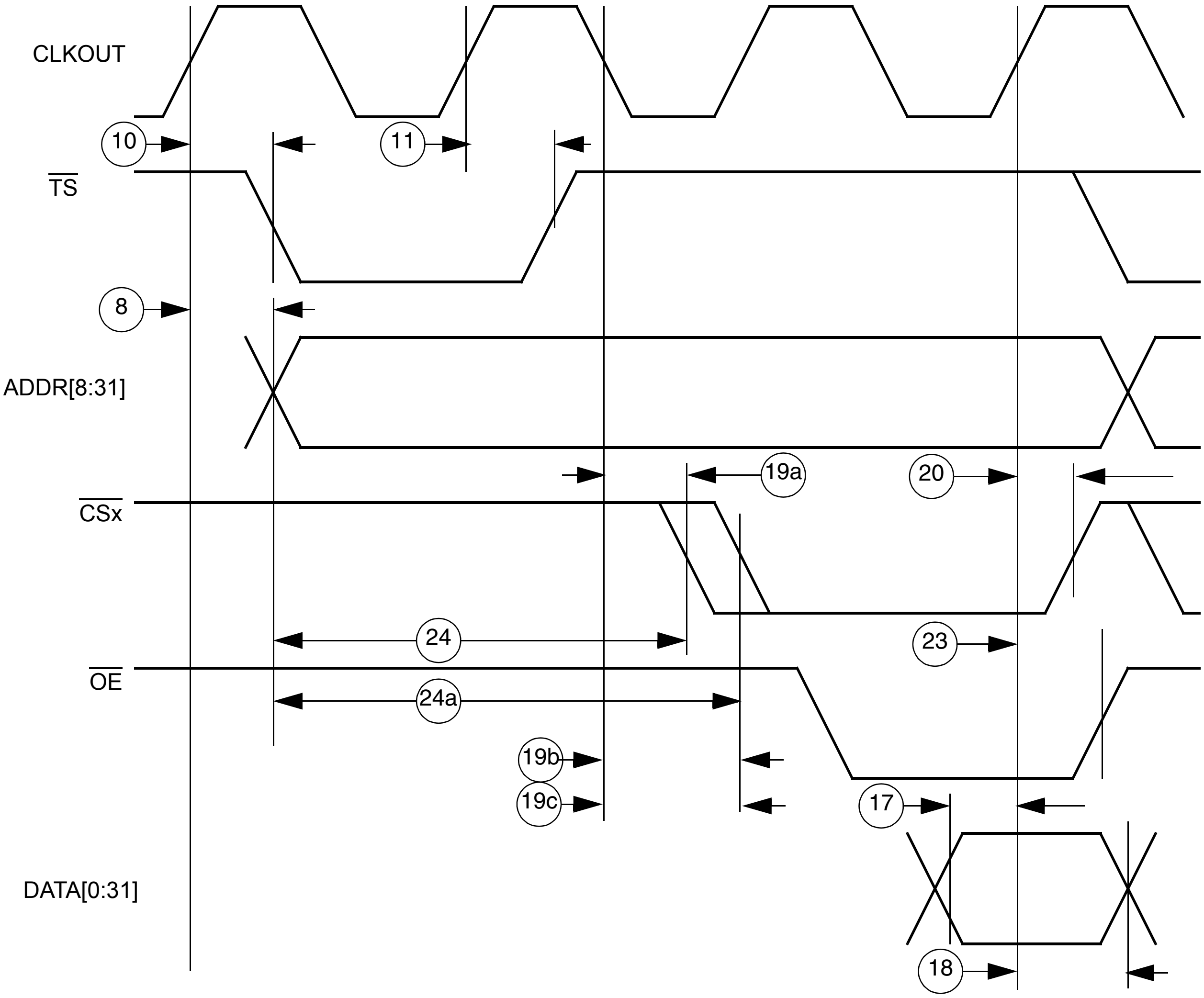


Figure G-18. External Bus Read Timing (GPCM Controlled – TRLX = ‘1’, ACS = ‘10’, ACS = ‘11’)