TOSHIBA RISC PROCESSOR

TMPR4955/TMPR4956F

(64-bit RISC MICROPROCESSOR)

1. GENERAL DESCRIPTION

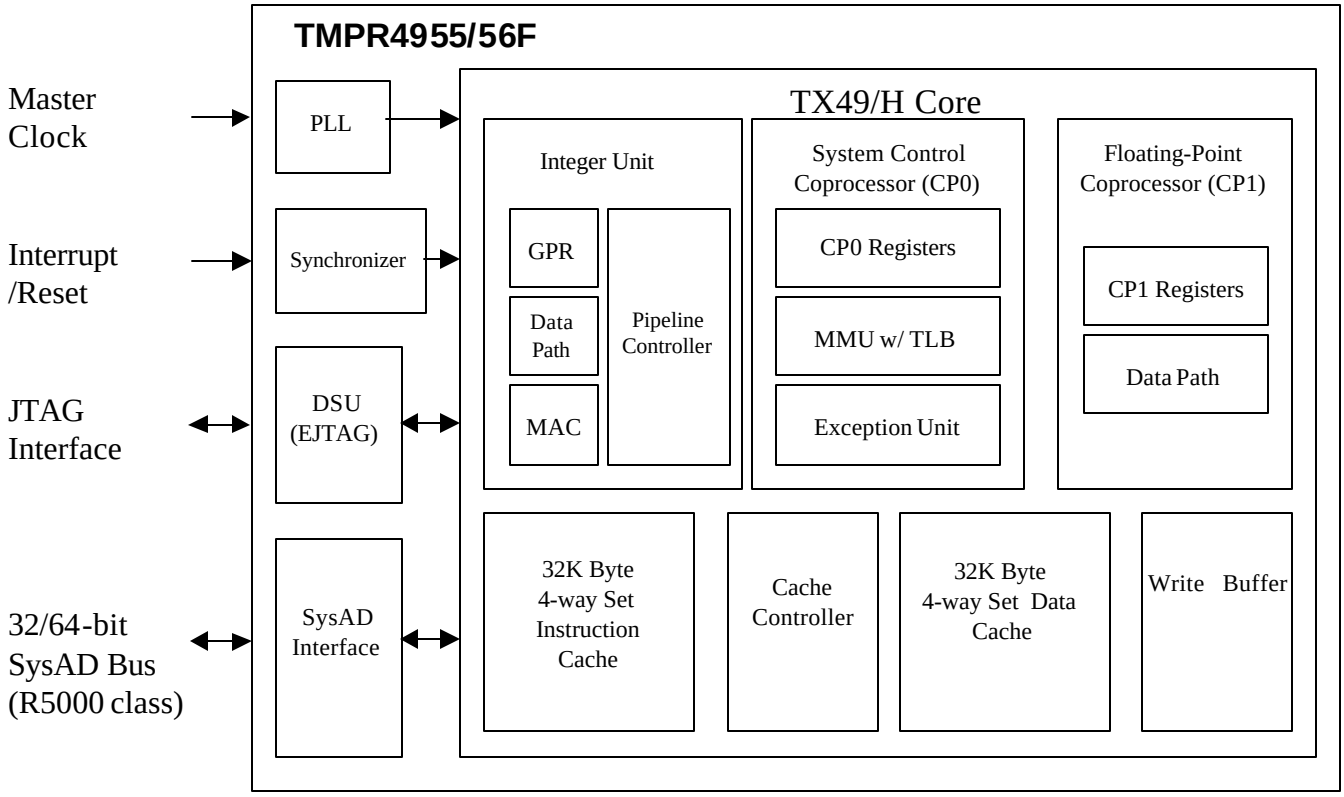
The TMPR4955/56F is a 64-bit RISC (Reduced Instruction Set Computer) microprocessor that is a low-cost, low-power microprocessor developed for interactive consumer applications including set-top terminals, LBP(Laser Beam Printer), and video games.

2. FEATURES

- True 64-bit microprocessor, with TX49/H core.
- Optimized 5-stage pipeline
- System Address/Data bus
  TMPR4955 : 32-bit System Address/Data bus
  TMPR4956 : 32-bit or 64-bit System Address/Data bus
- Floating-Point Operation
  Single or double-precision floating-point unit (IEEE Standard 754 exceptions)
- 36-bit physical address space and 64-bit virtual address space.
- On-chip 32-Kbyte Instruction Cache and 32-Kbyte Data Cache.
  4-way set associative and Lock function support
- Low power consumption
  3.3 /2.5V Dual power supply (I/O:3.3V,Internal:2.5V)
  Reduced power mode (Doze/Halt)
- Instruction cache prefetching
- Memory management unit
  contains 48-double entry JTLB, 2-entry Instruction TLB, and 4-entry Data TLB
- Software compatibility with all MIPS processors
  MIPS I, II, and III Instruction Set Architecture (ISA)
- EJTAG (Enhanced JTAG) debug support
- Package :
  TMPR4955 : 160-pin QFP
  TMPR4956 : 208-pin QFP
- Maximum operating frequency
  Internal:167/200MHz

3. SYSTEM CONFIGURATION

3.1 TMPR4955/56F BLOCK DIAGRAM



3.2 BLOCK FUNCTION

❑ **TX49/H Core**

- True 64-bit microprocessor
- 32, 64-bit integer general purpose registers
- 32, 32-bit floating point general purpose registers
- Optimized 5-stage pipeline
- Instruction Set
  - Upward compatible with MIPS I,MIPS II, MIPS III ISA
  - MAC(Multiply and Accumulate) instructions
  - PREF(Prefetch) instruction
- On-chip 32-Kbyte Instruction Cache and 32-Kbyte Data Cache
  - 4-way set associative and Lock function support
  - Data Cache: Write-back and Write-through support
- MMU
  - 36-bit physical address space and 64-bit virtual address space
  - 48-double-entry (even/odd) Joint TLB
  - 2-entry Instruction TLB and 4-entry Data TLB
- IEEE754 compatible single and double precision FPU
- Debug Support Unit (DSU) with EJTAG support
- Power management modes (HALT/DOZE)

❑ **SysAD BUS I/F**

- Bus protocol conversion

  It converts TX4955/56 Internal GBus Read/Write request into outside SysAD Bus protocol.

❑ **Sysnchronizer**

- The external interrupt

  It takes contents of interrupt register and bitwise OR of external interrupt signal (INT(5:0)).

❑ **Clock Generator**

- Generates the internal operating clock of the TX4955/56 from external crystal oscillator.

❑ **Debug Support Unit (DSU)**

- EJTAG function support

  Consists of an Enhanced JTAG (EJTAG) Module and a Debug Support Unit (DSU). It can be used to provide single-step execution and hardware break-points for debugging processor systems. EJTAG utilizes JTAG interface and extends the ability to access the inside register contents, host sytem peripherals, and system memory.

4. PIN DESCRIPTION

4.1 PIN OUT (TX4955 160-pin QFP)

1	Vss	41	Vss	81	VccInt	121	SysAD28
2	VccIO	42	TRST*	82	NMI*	122	SysAD29
3	JTDO	43	RdRdy*	83	ExtRqst*	123	VccInt
4	JTDI	44	WrRdy*	84	Reset*	124	Vss
5	JTCK	45	ValidIn*	85	ColdReset*	125	SysAD30
6	JTMS	46	ValidOut*	86	VccIO	126	VccIO
7	VccIO	47	Release*	87	Endian	127	Vss
8	Vss	48	VccIO	88	VccIO	128	SysAD31
9	SysAD4	49	PLLReset*	89	Vss	129	SysADC2
10	SysAD5	50	VccInt	90	SysAD16	130	VccInt
11	VccInt	51	TintDis	91	VccInt	131	Vss
12	Vss	52	Vss	92	Vss	132	SysADC3
13	SysAD6	53	SysCmd0	93	SysAD17	133	VccIO
14	VccIO	54	SysCmd1	94	SysAD18	134	Vss
15	Vss	55	SysCmd2	95	VccIO	135	SysADC0
16	SysAD7	56	SysCmd3	96	Vss	136	VccInt
17	SysAD8	57	SysCmd4	97	SysAD19	137	Vss
18	VccInt	58	SysCmd5	98	VccInt	138	SysADC1
19	Vss	59	VccIO	99	Vss	139	SysAD0
20	SysAD9	60	Vss	100	SysAD20	140	VccIO
21	VccIO	61	SysCmd6	101	SysAD21	141	Vss
22	Vss	62	SysCmd7	102	VccIO	142	SysAD1
23	SysAD10	63	SysCmd8	103	Vss	143	SysAD2
24	SysAD11	64	SysCmdP	104	SysAD22	144	VccInt
25	VccInt	65	VccInt	105	VccInt	145	Vss
26	Vss	66	Vss	106	Vss	146	SysAD3
27	SysAD12	67	VccIO	107	SysAD23	147	PCST8
28	VccIO	68	HALT/DOZE	108	SysAD24	148	PCST7
29	Vss	69	Int0*	109	VccIO	149	PCST6
30	SysAD13	70	Int1*	110	Vss	150	PCST5
31	SysAD14	71	Int2*	111	SysAD25	151	PCST4
32	VccInt	72	Int3*	112	VccInt	152	VccIO
33	Vss	73	Int4*	113	Vss	153	Vss
34	SysAD15	74	Int5*	114	SysAD26	154	VccIO
35	VccIO	75	VccIO	115	SysAD27	155	VssPLL
36	PCST3	76	Vss	116	VccIO	156	PLLCAP
37	PCST2	77	TPC3	117	VccIO	157	VccPLL
38	PCST1	78	TPC2	118	DivMode1	158	Vss
39	PCST0	79	TPC1	119	DivMode0	159	MasterClock
40	VccIO	80	DCLK	120	Vss	160	VccIO

Note: " * " means the signal is the low-active.

4.2 PIN OUT (TX4956 208-pin QFP)

1	Vss	43	VccInt	85	Vss	127	Vss	169	SysADC2
2	VccIO	44	Vss	86	VccInt	128	SysAD20	170	SysADC6
3	JTDO	45	SysAD15	87	Vss	129	SysAD52	171	VccInt
4	JTDI	46	SysAD47	88	VccIO	130	SysAD21	172	Vss
5	JTCK	47	GBS64*	89	Vss	131	SysAD53	173	SysADC3
6	JTMS	48	PCST3	90	HALT/DOZ	132	VccIO	174	SysADC7
7	VccIO	49	PCST2	91	Vss	133	Vss	175	VccIO
8	Vss	50	PCST1	92	Int0*	134	SysAD22	176	Vss
9	SysAD4	51	PCST0	93	Int1*	135	SysAD54	177	SysADC0
10	SysAD36	52	VccIO	94	Int2*	136	VccInt	178	SysADC4
11	SysAD5	53	Vss	95	Int3*	137	Vss	179	VccInt
12	SysAD37	54	Vss	96	Int4*	138	SysAD23	180	Vss
13	VccInt	55	TRST*	97	Int5*	139	SysAD55	181	SysADC1
14	Vss	56	RdRdy*	98	VccIO	140	SysAD24	182	SysADC5
15	SysAD6	57	WrRdy*	99	Vss	141	SysAD56	183	SysAD0
16	SysAD38	58	ValidIn*	100	Vss	142	VccIO	184	SysAD32
17	VccIO	59	ValidOut*	101	TPC3	143	Vss	185	VccIO
18	Vss	60	Release*	102	TPC2	144	SysAD25	186	Vss
19	SysAD7	61	VccIO	103	TPC1	145	SysAD57	187	SysAD1
20	SysAD39	62	Vss	104	DCLK	146	VccInt	188	SysAD33
21	SysAD8	63	PLLReset*	105	VccInt	147	Vss	189	SysAD2
22	SysAD40	64	VccInt	106	NMI*	148	SysAD26	190	SysAD34
23	VccInt	65	Vss	107	ExtRqst*	149	SysAD58	191	VccInt
24	Vss	66	TintDis	108	Reset*	150	SysAD27	192	Vss
25	SysAD9	67	Vss	109	ColdRese	151	SysAD59	193	SysAD3
26	SysAD41	68	VccInt	110	VccIO	152	VccIO	194	SysAD35
27	VccIO	69	Vss	111	Endian	153	VccIO	195	PCST8
28	Vss	70	SysCmd0	112	VccIO	154	DivMode1	196	PCST7
29	SysAD10	71	SysCmd1	113	Vss	155	DivMode0	197	PCST6
30	SysAD42	72	SysCmd2	114	SysAD16	156	Vss	198	PCST5
31	SysAD11	73	SysCmd3	115	SysAD48	157	SysAD28	199	PCST4
32	SysAD43	74	VccIO	116	VccInt	158	SysAD60	200	VccIO
33	VccInt	75	Vss	117	Vss	159	SysAD29	201	Vss
34	Vss	76	SysCmd4	118	SysAD17	160	SysAD61	202	VccIO
35	SysAD12	77	SysCmd5	119	SysAD49	161	VccInt	203	VssPLL
36	SysAD44	78	VccIO	120	SysAD18	162	Vss	204	PLLCAP
37	VccIO	79	Vss	121	SysAD50	163	SysAD30	205	VccPLL
38	Vss	80	SysCmd6	122	VccIO	164	SysAD62	206	Vss
39	SysAD13	81	SysCmd7	123	Vss	165	VccIO	207	MasterClock
40	SysAD45	82	SysCmd8	124	SysAD19	166	Vss	208	VccIO
41	SysAD14	83	SysCmdP	125	SysAD51	167	SysAD31		
42	SysAD46	84	VccInt	126	VccInt	168	SysAD63		

Note: " * " means the signal is the low-active.

4.3 PIN FUNCTION

The following is a list of interface, interrupt, and miscellaneous pins available on the TMPR4955/56F.

4.3.1 SYSTEM INTERFACE

PIN NAME	I / O	FUNCTION
SysAD(63:0)	I / O	**System address / data bus** A 64-bit address and data bus for communication between the processor and an external agent. (TX4955 is 32-bit)
SysCmd(8:0)	I / O	**System command / data identifier bus** A 9-bit bus for command and data identifier transmission between the processor and an external agent.
SysADC(7:0)	I / O	**System command/data check bus (TX4955 is 4-bit)** An 8-bit bus containing parity check bits for the SysAD bus during data cycles.
SysCmdP	I / O	**Reserved for system command/data identifier bus parity** For the TX4956 this signal is unused on input and zero on output.
ValidIn*	I	**Valid input** The external agent asserts ValidIn* when it is driving a valid address or data on the SysAD bus and a valid command or data identifier on the SysCmd bus.
ValidOut*	O	**Valid output** The processor asserts ValidOut* when it is driving a valid address or data on the SysAD bus and a valid command or data identifier on the SysCmd bus.
ExtRqst*	I	**External request** An external agent asserts ExtRqst* to request use of the System interface.
Release*	O	**Release interface** Signals that the system interface needs to submit an external request.
WrRdy*	I	**Write Ready** Signals that an external agent can now accept a processor write request.
RdRdy*	I	**Read Ready** Signals that an external agent can now accept a processor read request.

4.3.2 CLOCK / CONTROL INTERFACE

PIN NAME	I / O	FUNCTION
MasterClock	I	**Master clock** Master clock input that establishes the processor operating frequency.
DivMode(1:0)	I	**Set the operational frequency of the System interface** DivMode(1:0) MasterClock PClock 0 41.8MHz/50MHz 167MHz/200MHz 1 66.8MHz/80MHz 167MHz/200MHz 2 83.5MHz/100MHz 167MHz/200MHz 3 55.7MHz/66Mhz 167MHz/200MHz
TintDis	I	**Timer-Interrupt disable input** 0 enable Timer-Interrupt 1 disable Timer-Interrupt
HALT/DOZE	O	**HALT/DOZE mode output** This signal output the status of HALT or DOZE mode. This signal indicates that the TX4956 is in the HALT or DOZE mode when this signal is “H”.
PLLReset*	I	**PLL reset input** A signal to halt the PLL oscillation of the TX4956 built-in clock generator. 0 PLL is halt(no oscillation) 1 PLL is enabled.
Endian	I	**Endianess input** Indicates the initial setting of the endian during a reset. 0 Little Endian 1 Big Endian

4.3.3 INTERRUPT INTERFACE

PIN NAME	I / O	FUNCTION
Int(5:0)*	I	**Interrupt** Five general processor interrupts, bit-wise ORed with bits 5:0 of the interrupt register and visible as bits 15:10 of the Cause register.
NMI*	I	**Non-maskable interrupt** Non-maskable interrupt, ORed with bit 6 of the interrupt register.

4.3.4 JTAG INTERFACE

PIN NAME	I / O	FUNCTION
JTDI	I	**JTAG data input / Debug interrupt input** Run-time mode : input serial data to data/instruction register of JTAG. Real-time mode : interrupt line to change the debug unit state from real time mode to Run-time mode.
JTCK	I	**JTAG clock input** The processor receives a serial clock on JTCK. On the rising edge of JTCK, both JTDI and JTMS are sampled.
JTDO/TPC(0)	O	**JTAG data output / PC Trace output** Run-time mode : output serial data from data/instruction register of JTAG. Real-time mode : output non-sequential program.
JTMS	I	**JTAG command** JTAG command signal, indicating the incoming serial data is command data.
DCLK	O	**Debug Clock** A clock output for a real-time debug system. The timing of a serial monitor bus and PC trace interface signal are all defined by this debug clock DCLK. The operation clock of the TX4956 is divided by 3 at the time of a serial monitor bus operation.
PCST(8:0)	O	**PC trace status** Output PC trace status information and the mode of the serial monitor bus.
TPC(3:1)	O	**PC trace output** Output a non-sequential program counter at DCLK.
TRST*	I	**Test Reset input** A reset input for a real-time debug system. When TRST* is asserted, the debug support unit (DSU) is initialized.

4.3.5 INITIALIZATION INTERFACE

PIN NAME	I / O	FUNCTION
Reset*	I	**Soft (Warm) Reset** This signal must be asserted synchronously with MasterClock for a soft reset.
ColdReset*	I	**Cold reset** This signal indicates to the processor that the +3.3V power supply is stable and the processor should initiate a cold reset sequence, resetting the PLL.
GBS64*	I	**64-bit external bus mode** Indicates the initial setting of the bus size during a reset. 0 64-bit mode 1 32-bit mode
PLLCAP	I	**PLL connect to capacitor** A capacitor is connected between PLLCAP and the VccPLL to ensure the proper operation of the phase-locked loop.

4.3.6 OTHERS

PIN NAME	I / O	FUNCTION
VccPLL	I	**Quiet V_{CC} for PLL** Quiet V_{CC} for the internal phase locked loop. (2.5v)
VssPLL	I	**Quiet V_{SS} for PLL** Quiet V_{SS} for the internal phase locked loop.
VccIO	I	**Vcc** Power supply pin for IO.(3.3v)
VccInt	I	**Vcc** Power supply pin for internal.(2.5v)
VSS	I	**Vss** Ground pin

5. TX4955/TX4956 System Interface

5.1 TX4955 and TX4956

System interface protocols for the TX4955 and TX4956 are fundamentally the same. This document provides explanations based on the TX4956.

- TX4955: 32-bit bus interface
- TX4956: Has two modes: 64-bit bus interface mode and 32-bit bus interface mode

5.2 Terminology

The following terms are used in this section.

- External agent: Logic device that is directly connected to the processor via the system interface so a processor can issue (instructions).
- System event: Event issued inside a processor which, when generated, means that access to external system resources is required.
- Sequence: Strict order of requests that the processor generates in order to provide service for system events.
- Protocol: Shift of signals for each cycle generated on the system interface so processor requests or external requests can be asserted.
- Syntax: Strict definition of the bit pattern on the encoded bus (command bus, etc.).

5.3 Explanation of System Interface

The TX4956 processor supports 64-bit address/data interfaces. This processor makes it possible to construct a processor system by processors and main memory. This processor also supports 32-bit external address/data interfaces. System interfaces consist of the following components:

- 64-bit address/data bus, SysAD
- 8-bit SysAD check bus, SysADC
- 9-bit command bus, SysCmd
- 1-bit SysCmd check parity, SysCmdP
- handshake signals
- – RdRdy*, WrRdy*
  – ExtRqst*, Release*
  – ValidIn*, ValidOut*

The TX4956 processor accesses external resources using the system interface in order to correct cache misses, uncached operation, and other problems.

5.3.1 Interface bus

Figure 5.3-1 illustrates the 64-bit address/data bus SysAD (63:0), which is the main communication bus of the system interface, and the 9-bit command bus SysCmd (8:0). SysAD and SysCmd are bi-directional busses. In other words, these two busses are used for the processor to issue processor requests and for the external agent to issue external requests.

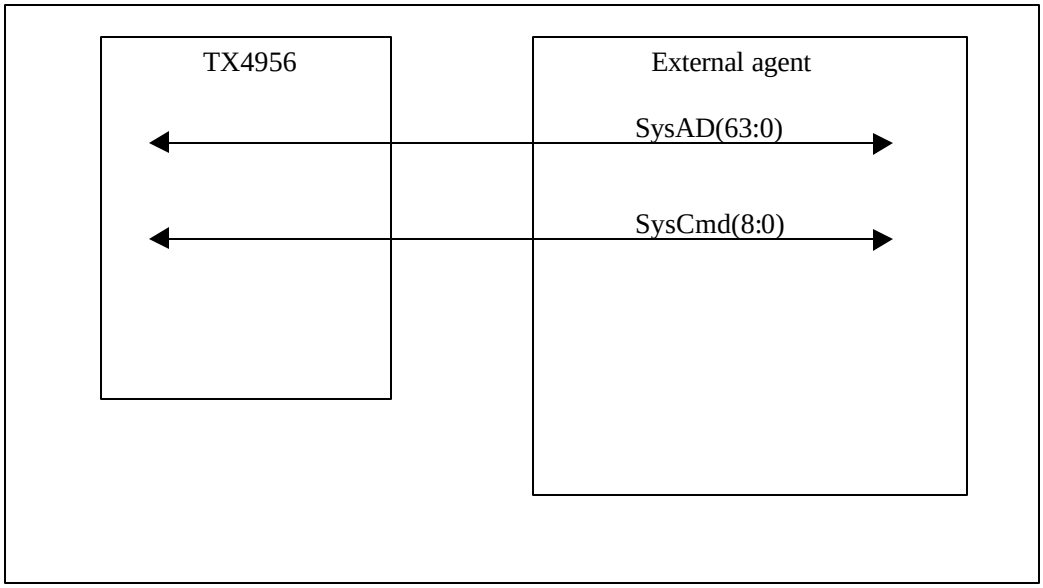


Figure 5.3-1 System Interface Bus

Requests sent via the system interface consist of the following:

- Address
- System interface command that strictly specifies the type of request
- Series of data elements for when the request is a particular write or read process.

5.3.2 Address cycle and data cycle

Cycles during which valid addresses exist on the SysAD bus are referred to as address cycles. Also, cycles during which valid data exist on the SysAD bus are referred to as data cycles. Validity is determined depending on the ValidIn* signals and ValidOut* signals.

The SysCmd bus is used to identify the contents of the SysAD bus for all cycles at which it is to be valid. The most significant bit of the SysCmd bus is used to indicate whether the current cycle is an address cycle or a data cycle.

- In the case of an address cycle [SysCmd(8) = 0], the remaining bits SysAD(7:0) of the SysCmd bus contain the system interface commands.
- In the case of a data cycle [SysCmd(8) = 1], the remaining bits SysAD(7:0) of the SysCmd bus contain the data identifier.

5.3.3 Issue cycle

Two types of processor issue cycles exist with the TX4956.

- Processor read request issue cycles.
- Processor write request issue cycles.

The TX4956 judges the issue cycle of the processor read request by sampling the RdRdy* signal. It also judges the issue cycle of the processor write request by sampling the WrRdy* signal from the external agent.

As illustrated in Figure 3-2, RdRdy* must be asserted two cycles before the processor read request address cycle in order to define the address cycle as an issue cycle.

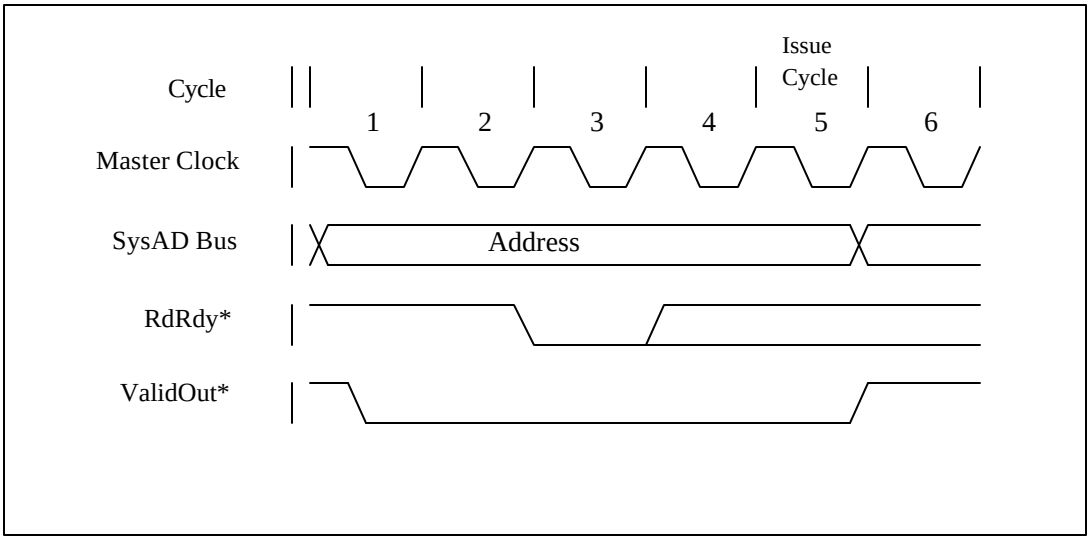


Figure 3-2 RdRdy* Signal Status in case of Read Request

As illustrated in Figure 3-3, WrRdy* is asserted two cycles before the initial address cycle of the processor write request, and the address cycle must be defined as the issue cycle.

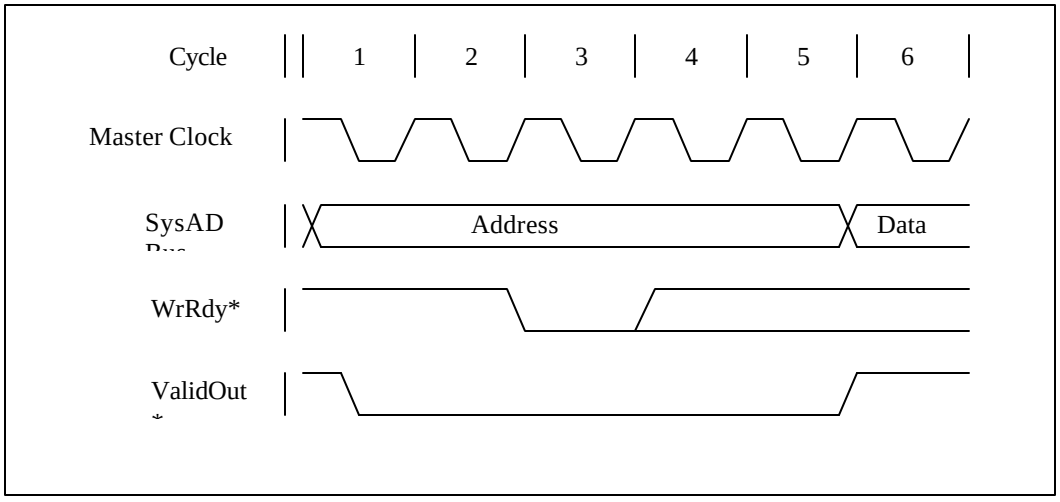


Figure 3-3 WrRdy* Signal Status in case of Write Request

The TX4956 repeats the request address cycle until the conditions of the valid issue cycle are met. If the processor request is a data transmission, then data transmission starts at the point when the issue cycle is complete. There is only one issue cycle no matter what the processor request is.

The TX4956 accepts external requests even while trying to issue processor requests. If the external agent asserts ExtRqst*, the processor responds to the external agent by releasing the system interface and going into the slave state. Rules relating to the issue cycle of processor requests are strictly applied in determining the processor run operation as well. The TX4956 performs one of the following:

- Complete issuing of processor requests before external requests are received.
- Release the system interface and go into the slave mode without completing issuance of the processor requests.

In the latter of the above situations, the TX4956 issues processor requests after external requests are complete. Rules relating to issuing are also provided to processor requests.

5.3.4 Handshake signal

The processor uses the eight control signals explained below to manage the flow of requests.

- RdRdy* and WrRdy* are used by the external agent to indicate that it is possible to receive new read transactions (RdRdy*) or new write transactions (WrRdy*).
- ExtRqst* and Release* are used to transfer SysAD bus and SysCmd bus control. ExtRqst* is used by the external agent to indicate the necessity of controlling the interface. Release* is asserted by the processor when transferring the system interface access privileges.
- The TX4956 processor uses ValidOut* and the external agent ValidIn* signals to indicate the valid command/data on the SysCmd/SysAD bus.

5.4 System Interface Protocol

Figure 4-1 illustrates the system interface that operates between registers. In other words, processor output is directly transferred from the output register and changes and the Master Clock rising edge.

Processor input is directly transferred to the input register and the input register latches these input signals at the rising edge of the Master Clock. In this way, it becomes possible for the system interface to operate at the fastest clock frequency.

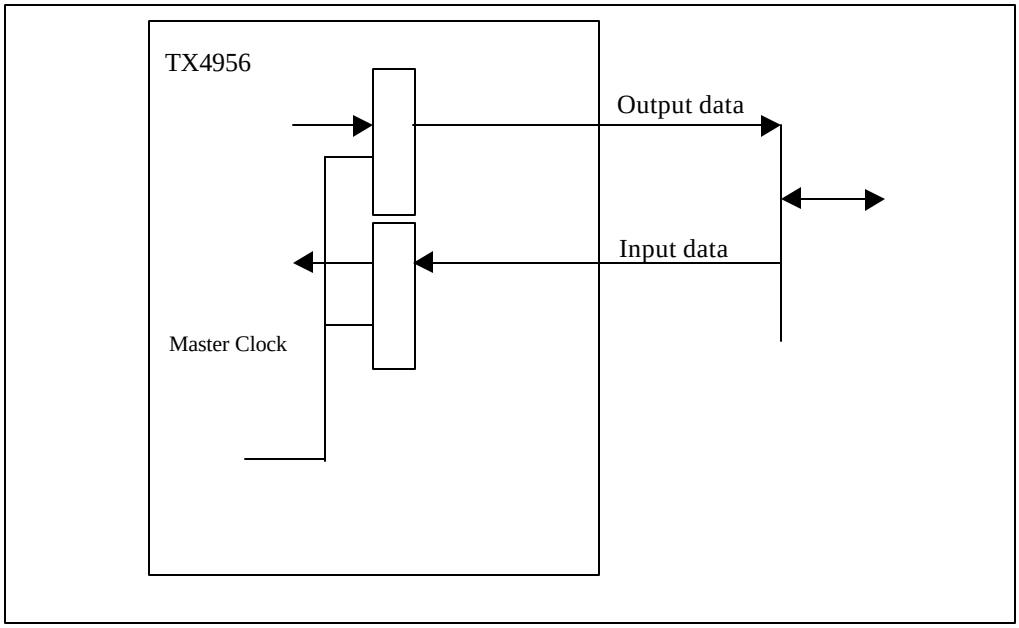


Figure 4-1 Operation of the System Interface Between Registers

5.4.1 Master state and slave state

The system interface is placed in the master state when the TX4956 processor is driving the SysAD bus and SysCmd bus. In contrast, the system interface is in the slave state when the external agent is driving the SysAD bus and SysCmd bus.

The processor asserts the ValidOut* signal if the SysAD bus and SysCmd bus become valid when the system interface is in the master state. The external agent asserts the ValidIn* signal if the SysAD bus and SysCmd bus become valid when the system interface is in the slave state.

5.4.2 Shifting from the master state to the slave state

The system interface remains in the master state unless it enters one of the following states:

- The external agent issues a request, then usage of the system interface is granted (external arbitration).
- The processor issues a read request and shifts into the slave mode by itself.

5.4.3 External arbitration

The external agent cannot issue external requests via the system interface unless the system interface goes

into the slave state. Shifts from the master state to the slave state are arbitrated by the processor using the system interface handshake signals ExtRqst* and Release.* This shift is performed as follows below.

1) The external agent sends notification that it would like to issue an external request by asserting the ExtRqst* signal.
2) The processor releases the system interface and changes its state from the master state to the slave state by asserting the Release* signal for 1 cycle.
3) The system interface returns to the master state when issuing of the external request is complete.

5.4.4 Shifting to the slave state on its own

Shifting to the slave state on its own means that the shift from the master state to the slave state is started by the processor when the processor read request is still on hold. The Release* signal is automatically asserted after the read transaction. Self-invoked shifting to the slave state occurs either during the issue cycle of the read request or several cycles after that.

After shifting to the slave state on its own, the processor returns to the master state at the end of the next external request. This is made possible by a read request or other type of external request.

The SysAd bus and SysCmd bus drives must start after the external agent confirms that the processor autonomously shifted to the slave state. While the system interface is in the slave state, the external agent can start making external requests without requesting access to the system interface (without asserting the ExtRqst* signal).

The system interface returns to the master state when the external request ends.

If a processor read request is on hold after a read request is issued, the processor automatically changes the system interface into the slave state even if the system interface access necessary for the system agent to issue the external request has not been requested. By shifting to the slave state in this manner, the external agent becomes able to return read response data.

5.5 Processor Requests and External Requests

Requests are broadly categorized as processor requests and external requests. This section will describe these two categories.

When a system event is generated, either a single request or a series of requests (referred to as processor requests) are issued via the system interface so the processor can access an external resource and invoke the service for the event. In order for this operation to be performed properly, the processor system interface must be connected to a system agent that meets the two following conditions:

1) It is in compliance to the system interface protocol.
2) It can regulate access to system resources.

An external agent that requests access to the processor cache or the status registers generates an external request. This access request is transferred via the system interface. Figure 5-1 illustrates the system event and request cycles.

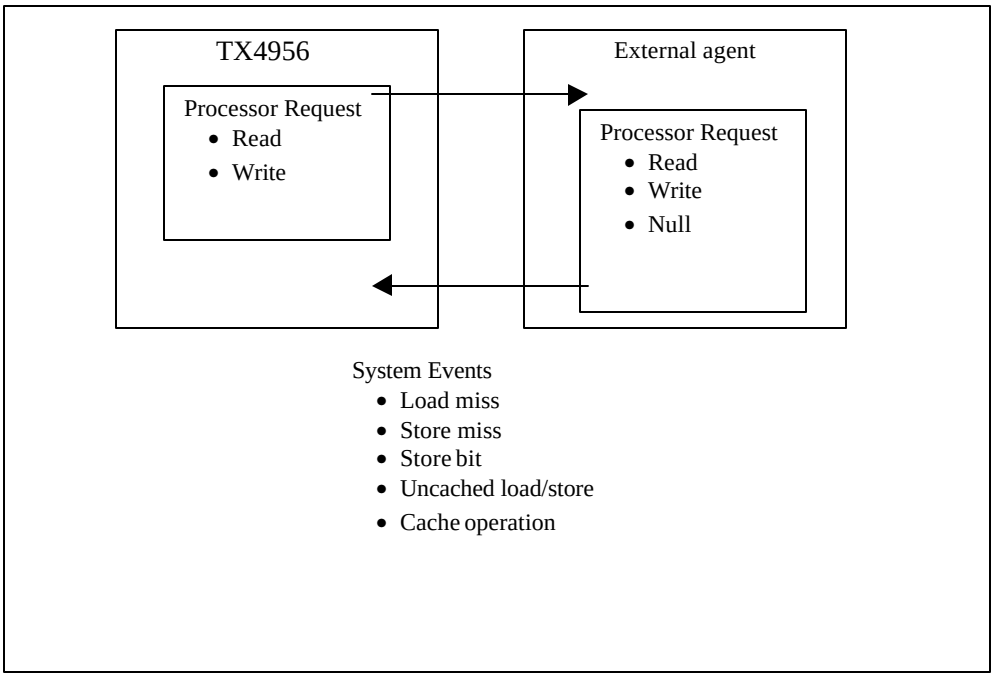


Figure 5-1 Requests and System Events

5.5.1 Rules relating to processor requests

The following rules apply to processor requests.

- After a processor read request is issued, the processor cannot issue the next read request until after it receives a read response.
- When in the R4000 compatible mode, after a write request is issued, at least 4 cycles must pass from when the write request issue cycle is complete until the processor can issue the next request. This is because two dummy system cycles are inserted as illustrated in Figure 5-2 by consecutive write requests of single data cycles.

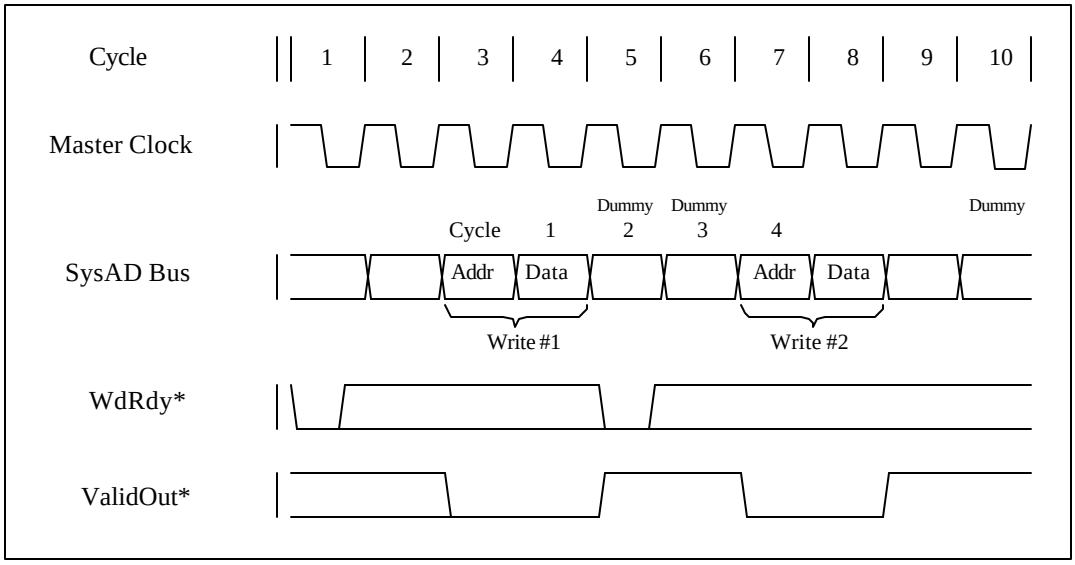


Figure 5-2 Timing of Consecutive Write Cycles

5.5.2 Processor requests

The term “processor request” refers to either a single request or a series of requests issued via the system interface in order to access external resources. As illustrated in Figure 5-3, there are two types of processor request: read and write.

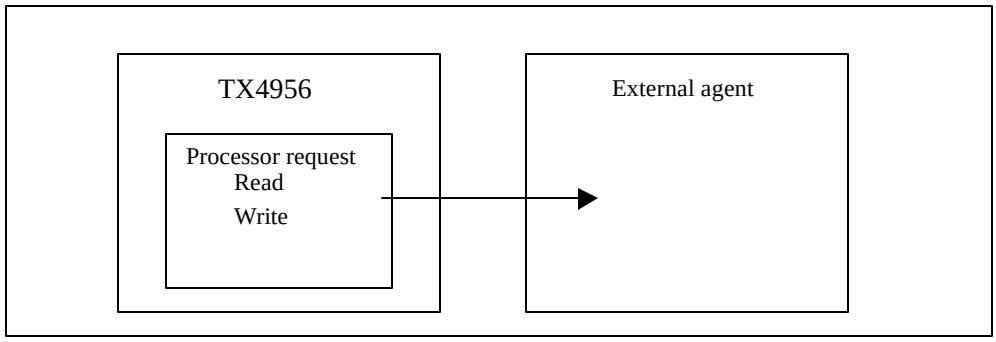


Figure 5-3 Processor Requests

Read requests are requests that read data from main memory or other memory resources in block double word, partial double word, word, and partial word units.

Write requests are requests that write data to main memory or other system resources in block, double word, partial double word, word, and partial word units.

Processor requests are managed by the TX4956 processor in the same manner as the R4000/R4400 non-secondary cache mode.

The processor issues requests strictly according to a sequential method. In other words, the processor cannot issue the next request while a previous request is on hold. For example, after issuing a read request, the processor waits for a read response before issuing the next request. The processor only issues

write requests when there are no read requests on hold.

When using processor input signals RdRdy* and WrRdy*, the external agent can control the processor request flow. RdRdy* is the signal that controls the processor read flow, and WrRdy* controls the processor write request flow. Figure 5-4 illustrates the sequence of the processor request cycle.

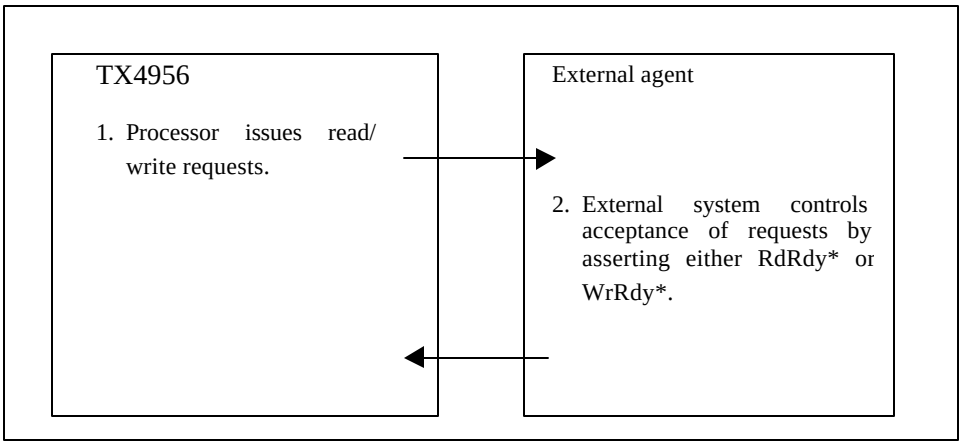


Figure 5-4 Processor Requests

5.5.3 Processor read requests

When the processor issues a read request, the external agent must access the specified resource and return the requested data.

The external agent returns response data for processor read requests so they can be executed separately from the requests. In other words, the external agent can start an external request before returning response data for the processor read request. A processor read request is complete when the final word of the response data is received from the external agent.

Depending on the data identifier combined with the response data, an error in the response data may be pointed out. The processor would then treat this error as a bus error.

If data have not been returned to the issued processor read request, the applicable request is said to be "on hold." This state continues until the requested read data are returned.

The external agent must be able to accept processor read requests at any time if either of the two following conditions is met.

- There is no processor read request that is on hold.
- The RdRdy* signal is asserted for 1 cycle 2 cycles before the issue cycle.

5.5.4 Processor write request

When the processor issues a write request, the specified resources are accessed, then the data are written to those resources.

Processor write requests are complete when the final data word is transferred to the external agent.

The external agent must be able to accept processor write requests at any time if either of the two following conditions are met.

- There is no processor read request that is on hold.
- The WrRdy* signal is asserted for 1 cycle 2 cycles before the issue cycle.

5.5.5 External requests

As illustrated in Figure 5-5, there are three types of external request: read, write, and null. This section will explain read responses, which are special external request cases.

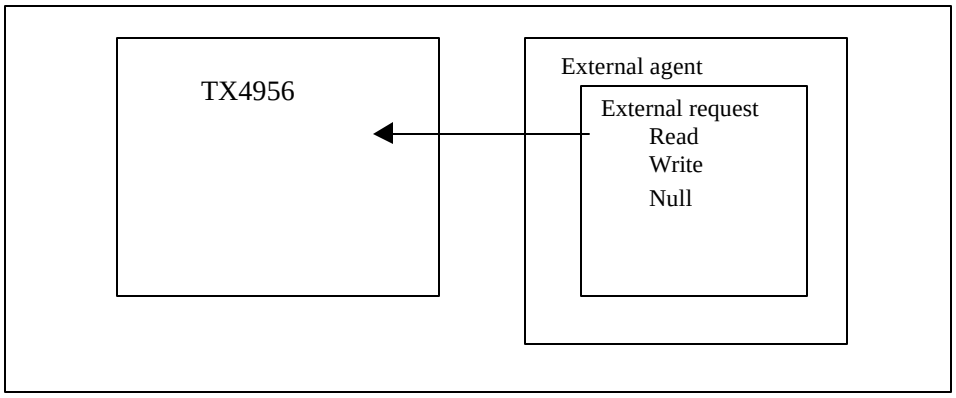


Figure 5-5 External Requests

Read requests are used to call 1-word data from processor internal resources. Write requests are used to write 1-word data to the processor internal resources. Null requests are requests that do not require processor operation.

As illustrated in Figure 5-6, the processor uses arbitration signals ExtRqst* and Release* to control the flow of external requests. The external agent cannot issue external requests unless access privileges to the system interface are obtained. In order to do so, the external agent asserts the ExtRqst* signal, then waits until the processor asserts the Release* signal for 1 cycle.

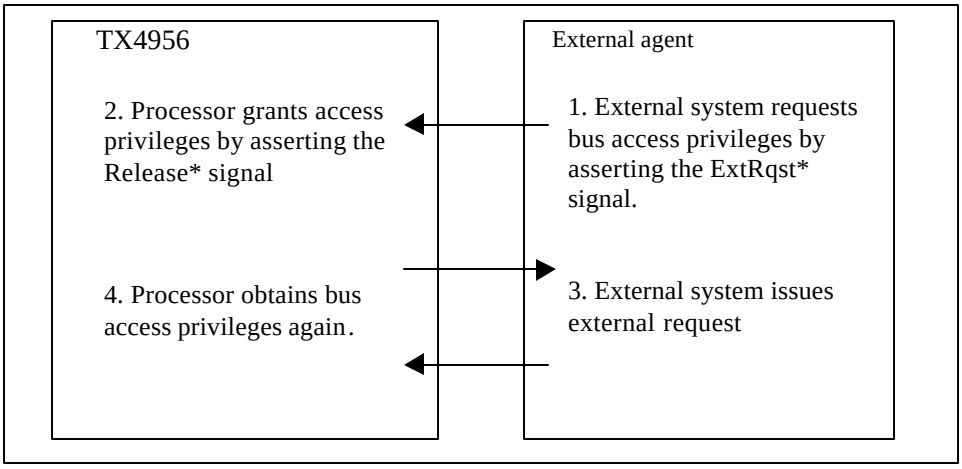


Figure 5-6 External Requests

After the external request is issued, the system interface access privileges always return to the processor. The processor will not accept another external request until the current one is complete.

If there is no processor request that is on hold, the processor decides based on the interior state whether to accept an external request or to issue a new processor request. The processor can issue a new processor request even if the external agent requested access to the system interface.

The external agent sends notification that it would like to start an external request by asserting the ExtRqst* signal. After that, the external agent waits for the processor to assert the Release* signal and send notification that preparations have been made to accept this request. The processor sends notification based on the next judgement criterion to be listed that preparations have been made to accept an external request.

- The processor ends processor requests that are in progress.
- The processor can accept an external request while waiting for the RdRdy* signal to be asserted so a processor read request can be issued. However, this request must be transferred to the processor at least 1 cycle before the RdRdy* signal is asserted.
- The processor can accept an external request while waiting for the WrRdy* signal to be asserted so a processor write request can be issued. However, this request must be transferred to the processor at least 1 cycle before the WrRdy* signal is asserted.
- If waiting for a response to a read request after the processor shifted itself to the slave state, the external agent can issue an external request before sending read response data.

5.5.6 External read requests

In contrast to processor read requests, data are directly returned as a response to the request for external read requests. No other requests can be issued until the processor returns the requested data. External read requests are complete when the processor returns the requested data word. Depending on the data identifier combined with the response data, an error in the response data may be pointed out. The processor would process the error as a bus error.

Note: The TX4956 does not have any resources that can read external read requests. The processor returns to the external read request undefined data and data identifiers in which SysCmd(5) of the errant data bit is set.

5.5.7 External write requests

When the external agent issues a write request, the specified resources are accessed, then the data are written to those resources. External requests are complete when the data word is transferred to the processor.

The only processor resource that an external write request can use are the Interrupt registers.

5.5.8 Read responses

As illustrated in Figure 5-7, read responses return data to processor read requests. Read responses are external requests, strictly speaking, but there is only one difference with other external requests: read responses do not request permission to use the system interface. Therefore, read responses are handled separately from other external requests and are simply referred to as read responses.

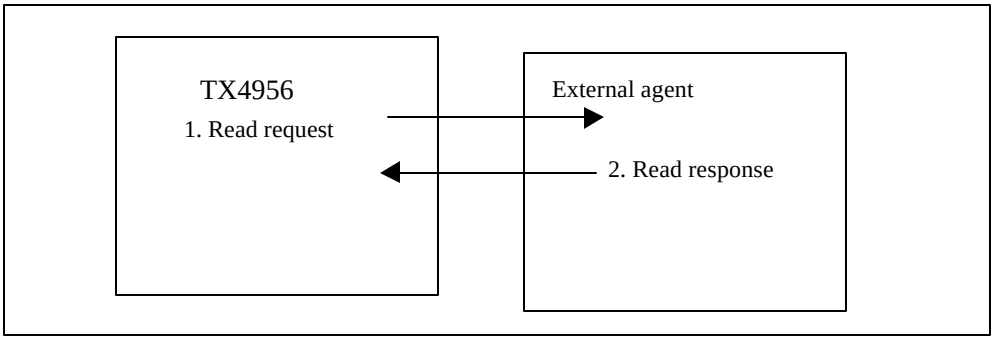


Figure 5-7 Read Response

5.6 Handling of Requests

This section will describe in detail sequences, protocol, and syntax for both processor and external requests.

- Load miss
- Store miss
- Store bit
- Cache operation
- Load Linked/Store Conditional

5.6.1 Load miss

If a processor load miss occurs in the primary cache, the processor cannot proceed to the next process if the cache line that contains the loaded data elements is not received from the external agent.

If the current cache line set in which the write back bit (W bit) is set is replaced by a new cache line, the current cache line must be written back.

The processor checks the coherency properties in the TLB entries for pages including the requested cache lines. If the coherency properties are non-coherent, then a non-coherent read request is issued. Table 6-1 indicates the measures that can be taken when a load miss occurs in the primary cache.

Page Properties	State of the replaced data cache line	
	Dirty (W = 0)/Invalid	Dirty (W = 1)
Non-coherent	NCR	NCR/W

NCR: Processor non-coherent, block read request
NCR/W: Processor non-coherent, block write requests continue after the block read request

Table 6-1 Load Miss to the Primary Cache

5.6.2 Store miss

When a store miss occurs in the primary cache, the processor cannot proceed to the next process if it does not receive from the external agent a cache line that includes a store target address. The processor checks the coherency properties in the TLB entries for pages including the requested cache line, then confirms whether to invalidate write transactions to that cache line or not.

After that, the processor executes one of the following requests:

- If the coherency properties are non-coherent write back or non-coherent write through (write allocate), then a non-coherent block read request is issued.
- If the coherency properties are non-coherent write through (non-write allocate), then a non-block write request is issued. Table 6-2 indicates the measures taken when there is a store miss to the primary cache.

Page Properties	State of the replaced data cache line	
	Dirty (W = 0)/Invalid	Dirty (W = 1)
Non-coherent write back or non-coherent write through (write allocate)	NCR	NCR/W
Non-coherent write through (non-write allocate)	NCW	NA

NCR: Processor non-coherent, block read request
NCR/W: Processor non-coherent, block write requests continue after the block read request
NCW: Processor non-coherent write request

Table 6-2 Store Miss to Primary Cache

5.6.3 Store hits

Operation in the system interface is determined by whether a line is write back or write through. When in the primary cache mode, all lines set to write back are set to the dirty exclusion state (W = 1). In other words, burst transactions do not occur even if a store hit occurs. Lines set to write through generate processor write requests for store data.

5.6.4 Uncached load or store

When performing uncached load operations, the processor issues non-coherent read requests for double words, partial double words, words, or partial words. Also, when performing uncached store operations, the processor issues write requests for double words, partial double words, words, or partial words.

When in the 64-bit mode, the TX4956 judges that there is valid parity and data in the entire 64-bit SysAD bus even for data requests of less than double words. Even if there was a partial word request for example, all parity must be correctly returned for all 64 bits. If not, then parity check must be disabled.

When in the 32-bit mode, the TX4956 judges that there is valid parity and data in the entire 32-bit SysAD bus even for data requests of less than words. Even if there was a partial word request for example, all parity must be correctly returned for all 32 bits. If not, then parity check must be disabled.

All write transactions by the TX4956 are buffered in the 4-stage write buffer of the system interface. If there are entries in the write buffer when a block request is required, the write buffer is flushed before a read request is generated (for cache misses or read transactions to uncached areas). Data cache misses or uncached data load transactions flush the write buffer.

5.6.5 Cache instruction operation

Various operations are made available to the Cache instruction in order to maintain the primary cache status and contents. When Cache instruction operations are in progress, write requests or invalidate requests can be issued from the processor.

5.6.6 Load Linked Store Conditional Operation

TX4956 and TX4955 are not supporting the Load-Link/Store-Conditional instruction sequence.

5.7 Processor Request and External Request Protocol

This section explains the bus arbitration protocol for both processor requests and external requests on a cycle-by-cycle basis. Table 7-1 below describes the abbreviations used in the following timing diagram of the bus.

Range	Abbreviation	Meaning
Total	Unsd	Unused
SysAD bus	Addr	Physical address
	Data<n>	Data number *n* of the data block
SysCmd bus	Cmd	Undefined system interface command
	Read	Processor or external read request command
	Write	Processor or external write request command
	SINull	External null request command that releases the system interface
	NData	Non-coherent data identifier for datum other than the final datum
	NEOD	Non-coherent data identifier for the final datum

Table 7-1 System Interface Request

5.7.1 Processor request protocol

Processor request protocol is as follows.

- Read
- Write
- Null write

5.7.2 Processor read request protocol

The processor read request protocol is as described in the following sequence. The next step numbers correspond to the numbers in Figure 7-1.

1. RdRdy* is asserted to Low by the external agent. This means that the external agent is ready to accept read requests.
2. When the system interface is in the master state, the read command is transmitted to the SysCmd bus, then the processor read request is issued by transmitting the read address to the SysAD bus.
3. At the same time, the processor asserts the ValidOut* signal for one cycle. This means that valid data are being transmitted to the SysCmd bus and SysAD bus.
4. The processor goes into the slave state by itself either at the issue cycle of a read request or after the Release* signal is asserted for one cycle and the issue cycle of the read request is complete.

   Note: The external agent must not assert the ExtRqst* signal as a means of returning a read response. It must however wait to shift to the slave state on its own. If an external request other than a read response is issued, ExtRqst* can be asserted either before the read response or in the process of the read response.

5. The SysCmd bus and SysAD bus are released from the processor one cycle after the Release* signal is asserted.
6. The SysCmd bus and SysAD bus are driven by the external agent within two cycles after the Release* signal is asserted.

When shifting to the slave state (from Cycle 5 in Figure 7-1), the external agent can return the requested data as a read response. Notification of an error in the returned data is sent if either the data requested by a read response were returned or if the requested data could not be fetched. In this case, the processor handles the result as a bus error exception.

Figure 7-1 illustrates a situation in which the slave state is autonomously shifted to after a processor read request is issued.

Note: The timing of the SysADC bus and SysCmdP bus are the same as the timing of the SysAD bus and SysCmd bus timing, respectively.

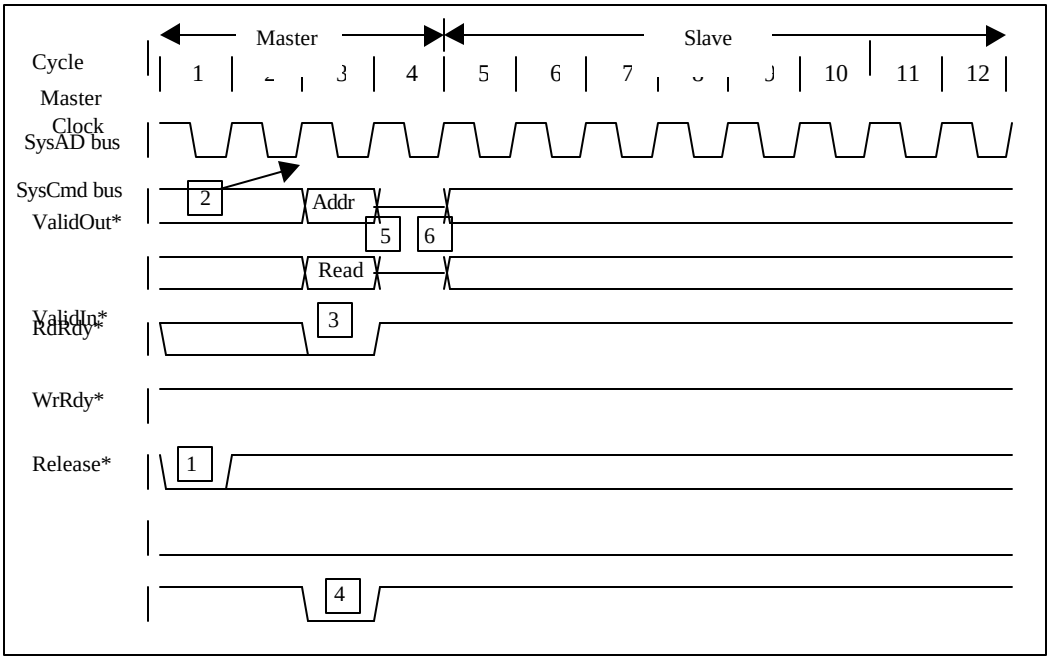


Figure 7-1 Processor Read Request Protocol

If the Release* signal is asserted, this means that either there is autonomous shifting to the slave state or there is a response to the ExtRqst* signal assertion. In this case, the processor can accept either a read response or an external request other than a read response. If an external request other than a read response is issued, the processor asserts Release* for 1 cycle, then autonomously shifts to the slave state again after the external request process.

5.7.3 Processor write request protocols

Either of the two following protocols is used in issuing processor write requests.

- The word write request protocol (see Note below) is used for double word, partial double word, word or partial word writing.

-

  Note: Words are called to differentiate from the block request protocol. It is actually possible to transfer data in double word, partial double word, word, or partial word units.

- The block write request protocol is used for block write transactions.

The system interface is used in the master state to issue processor double word write requests. Figure 7-2 illustrates processor non-coherent single word write request cycles.

1. In order to issue a processor single word write request, a write command is sent to the SysCmd bus, and a write address is sent to the SysAD bus.
2. The processor asserts the ValidOut* signal.
3. The processor sends the data identifier to the SysCmd bus and transmits data to the SysAD bus.
4. The data identifier for this data cycle must receive an indication that this is the final data cycle. ValidOut* is deasserted at the end of the cycle.

   Note: The SysADC bus and SysCmd bus timing is the same as the SysAD bus and SysCmd bus, respectively.

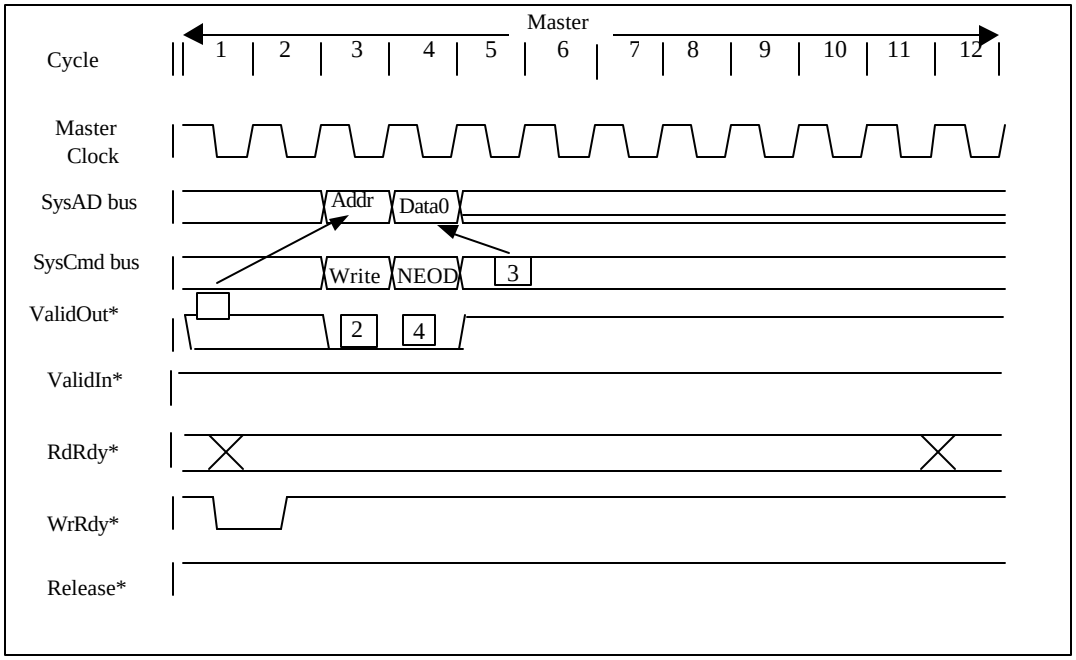


Figure 7-2 Processor Non-coherent Single Word Write Request Protocol

5.7.4 Processor single write requests

There are three processor single write requests as follow below.

With later G2SConfig-Regster, these modes are selected.

1. R4000 compatible write
2. Reissue write
3. Pipeline write

1. R4000 compatible write

When in the R400 compatible write mode, 4 cycles are required for single write operation. After the address is asserted for 1 cycle, it is followed by 2 cycles of dummy data. This applies whether in the 64-bit bus mode or the 32-bit bus mode. Figure 7-3 illustrates its basic operation.

In the case of the TX4956, the WrRdy* signal must be asserted for 1 cycle 2 cycles before the write operation is issued. When in the R4000 compatible signal write mode, the external agent receives the write data then immediately asserts WrRdy*, making it possible to stop write operation that continues after 4 cycles. The 2 cycles of dummy data that follow these write data give the external agent time to stop the next write operation.

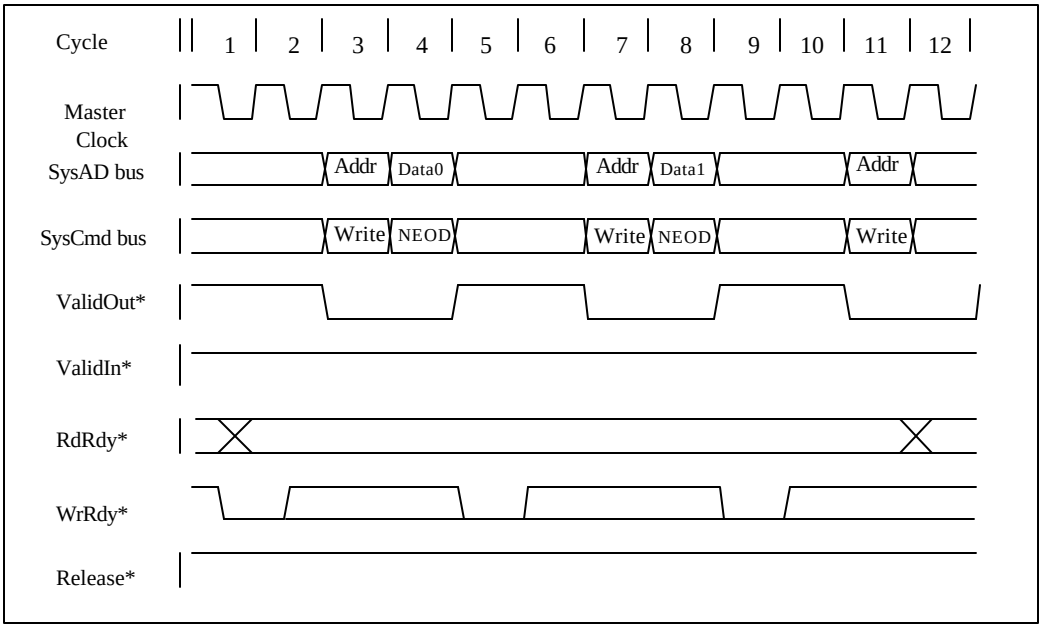


Figure 7-3 R4000 Compatible Write

2. Reissue write

When in the reissue write mode, the WrRdy* signal is asserted for 1 cycle 2 cycles before the address cycle, and the write operation is reissued when the WrRdy* signal is asserted during the address cycle. Figure 7-4 illustrates the reissue write protocol.

- By asserting (Low) the WrRdy* signal in the first and third cycles, Addr0/Data0 issues a write operation in the third or fourth cycle.
-
- By deasserting (High) the WrRdy* signal in the fifth cycle, Addr1/Data1 does not issue a write operation in the fifth and sixth cycles.
- By asserting (Low) the WrRdy* signal again in the eighth and tenth cycles, Addr1/Data issues a write operation in the tenth and eleventh cycles.

•

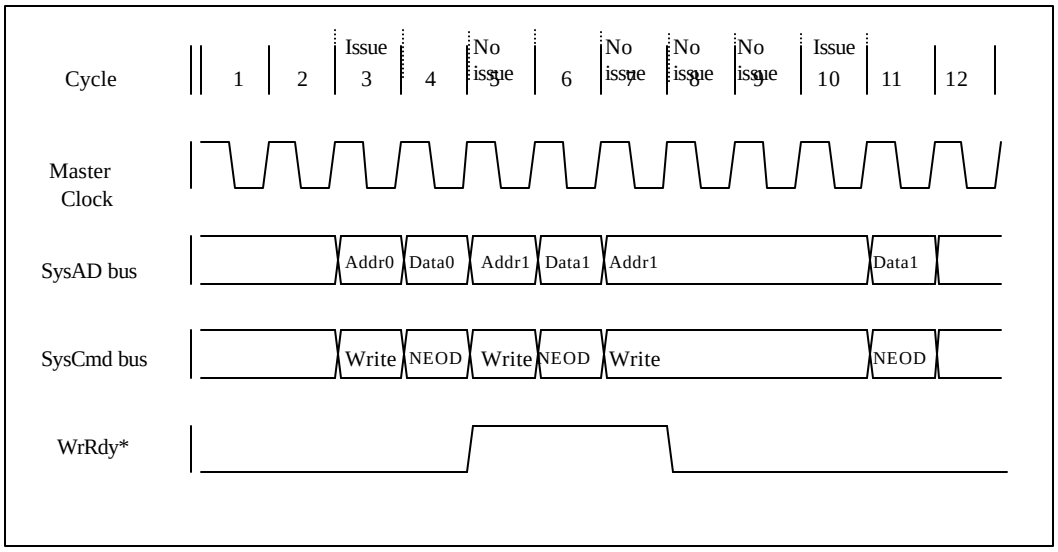


Figure 7-4 Reissue Write Protocol

3. Pipeline write

Similar to when in the R4000 compatible write mode, the pipeline write protocol issues a write operation if the WrRdy* signal is asserted for 1 cycle 2 cycles before the write operation is issued. However, the 2 cycles of dummy data after the write operation are deleted. The external agent must be able to accept one write operation or more after WrRdy* is deasserted. Figure 7-5 illustrates this protocol.

- Third, fourth cycle Addr0/Data0 is issued by asserting (Low) the WrRdy* signal in the first cycle.
- Fifth, sixth cycle Addr1/Data1 is issued by asserting (Low) the WrRdy* signal in the third cycle.
- Addr2 is not issued in the seventh cycle when the WrRdy* signal is deasserted (High) in the fifth cycle. Addr2/Data2 is issued in the tenth, eleventh cycle by asserting the WrRdy* signal again in the eighth cycle.

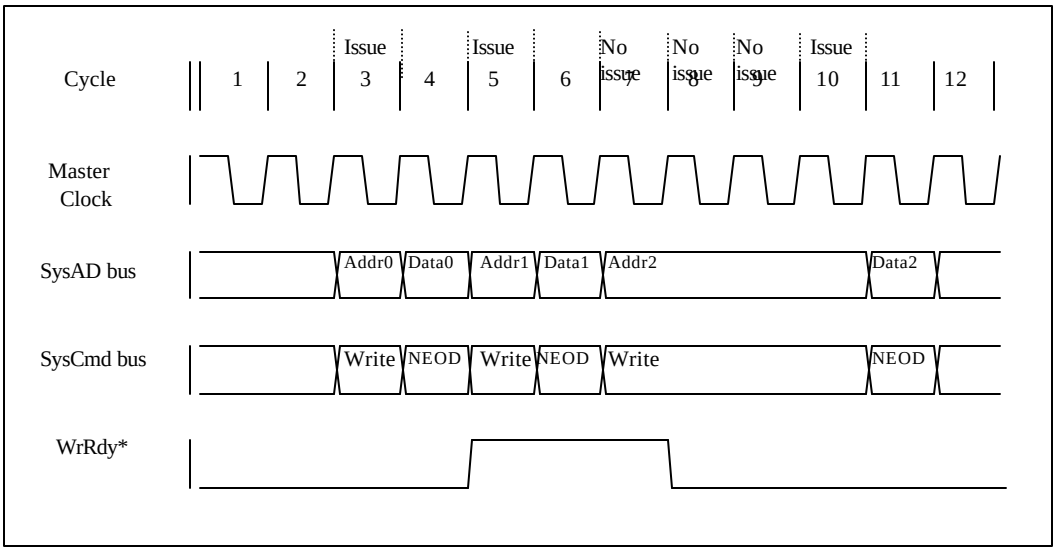


Figure 7-5 Pipeline Write Protocol

5.7.5 Processor block write request

The master state system interface is used to issue processor block write requests. The protocol is the same for both the 64-bit bus mode and the 32-bit bus mode. Figure 7-6 illustrates a processor non-coherent block request made for 8-word data with the "D" data pattern when in the 64-bit bus mode.
Because TX4956 is a thing of 64-bit bus width, 8-word data get off with four data transmission.

1. Processor sends a write command to the SysCmd bus, then sends a write address to the SysAD bus.
2. Processor asserts the ValidOut* signal.
3. Processor sends data identifier to the SysCmd bus and sends data to the SysAD bus.
4. Processor asserts the ValidOut* signal only for the number of cycles required to transfer the data block.
5. Final data cycle directive must be included in data identifiers for the final data cycle.

Note : There is transmission protocol of Processor block write request three kinds same as Processor single write requests. With later G2SConfig-Regster, these modes are selected.

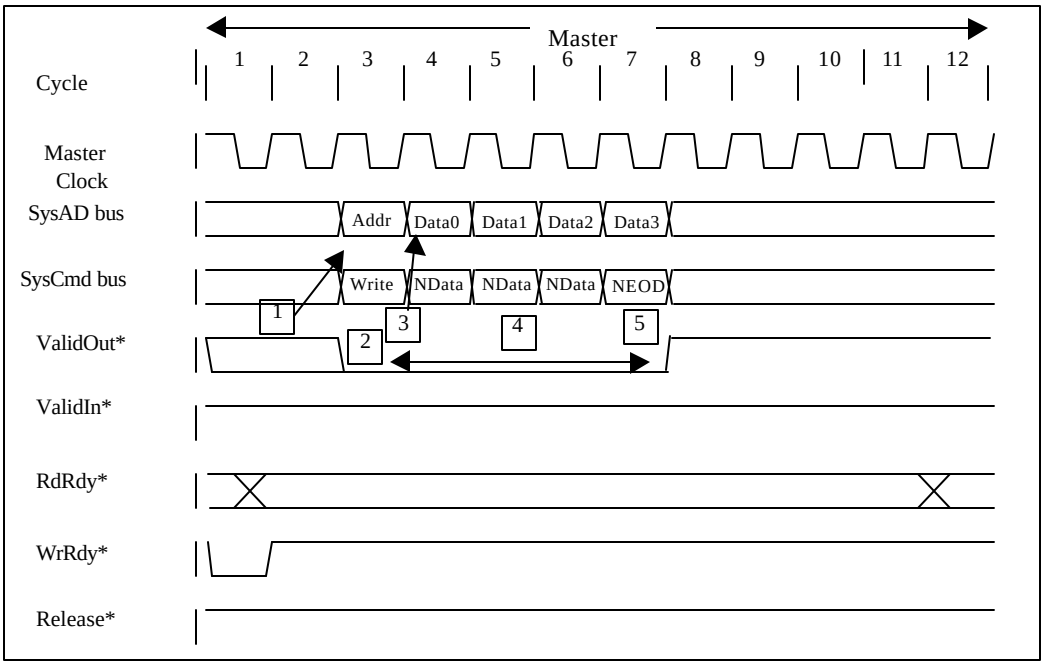


Figure 7-6 Processor non-coherent block request protocol

5.7.6 External request protocol

External requests can only be issued when the system interface is in the slave state. The external agent asserts the ExtRqst* signal, and requests use of the system interface. The processor asserts the Release* signal, releases the system interface, waits for it to enter the slave state, then the external agent issues an external request. If the system interface is already in the slave mode, namely, if the processor has put the system interface in the slave state on its own, then the external agent can immediately issue an external request.

In the case of the external agent, the system interface must be returned to the master state after issuing an external request. When the external agent issues a single external request, ExtRqst* must be deasserted 2 cycles after the cycle at which Release* is asserted. Also, when issuing a series of external requests, the ExtRqst* signal must be asserted before the last request cycle.

The processor continues processing external requests while ExtRqst* is asserted. However, until the processor completes a request that is currently being processed, it will not be able to release the system interface and put it into the slave state in preparation for the next external request. Also, until ExtRqst* is asserted, a series of external requests cannot be interrupted by a processor request. This protocol is the same for both the 64-bit and 32-bit bus modes.

5.7.7 External arbitration protocol

As previously mentioned, the ExtRqst* signal and Release* signal are used in system interface arbitration. Figure 7-7 illustrates the timing of the arbitration protocol when the slave state changes to the master state.

The arbitration cycle sequence is as follows.

1. The external agent asserts ExtRqst* when it becomes necessary to issue external requests.
2. The processor asserts Release* for 1 cycle when it becomes possible to process an external request.
3. The processor sets the SysAD bus and SysCmd bus to tri-state.
4. The external agent must start transmission to the SysAD bus and SysCmd bus 2 cycles after Release* is asserted.
5. The external agent deasserts ExtRqst* 2 cycles after Release* is asserted. This does not apply however to situations where an attempt is made to issue another external request.
6. The external agent sets the SysAD bus and SysCmd bus to tri-state when processing of the external request is complete.

The processor becomes able to issue processor requests 1 cycle after the external agent sets the busses to tri-state.

Note: The SysADC bus and SysCmdP bus timing is the same as that for the SysAD bus and SysCmd bus, respectively.

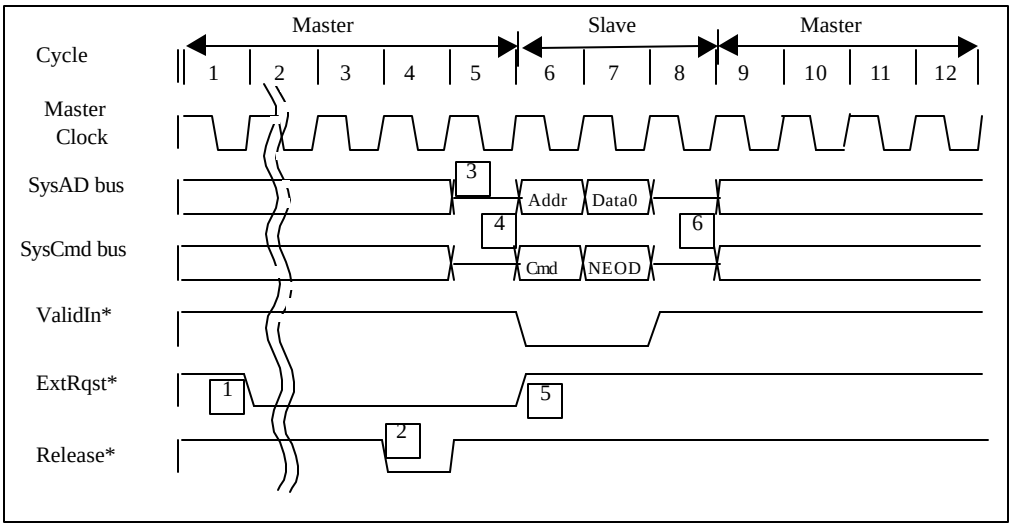


Figure 7-7 Arbitration Protocol Relating to External Requests

5.7.8 External read request protocol

External read requests are requests that read 1 word of data from processor-internal resources such as registers. External read requests cannot be partitioned. Namely, it is not possible to generate other requests between an external read request and the corresponding read response.

Figure 7-8 illustrates the timing of external read requests, which consist of the following steps.

1. The external agent requests use of the system interface by asserting ExtRqst*.
2. The processor asserts Release* for 1 cycle, then releases the system interface by deasserting Release* and puts the interface into the slave state.
3. After Release* is deasserted, the SysAD bus and SysCmd bus are set to tri-state for 1 cycle.
4. The external agent sends a read request command to the SysCmd bus, sends a read request address to the SysAD bus, and asserts ValidIn* for 1 cycle.
5. After sending the above address and command, the external agent sets the SysCmd and SysAD busses to tri-state, makes it possible for the processor to drive them, then releases them both. The processor that accessed the data to be read returns the data to the external agent. Therefore, the processor sends the data identifier to the SysCmd bus, sends the response data to the SysAD bus, then asserts ValidOut* for 1 cycle. This data identifier indicates that data are the response data of the final data cycle.
6. The system interface is in the master state. The processor continues to drive the SysCmd bus and SysAD bus even after the read response is returned.

Note: Timing of the SysADC bus and SysCmdP bus is the same as that for the SysAD and SysCmd busses, respectively.

External read requests can read data from only a single word in the processor. If data elements other than a word are requested, the processor response is not defined.

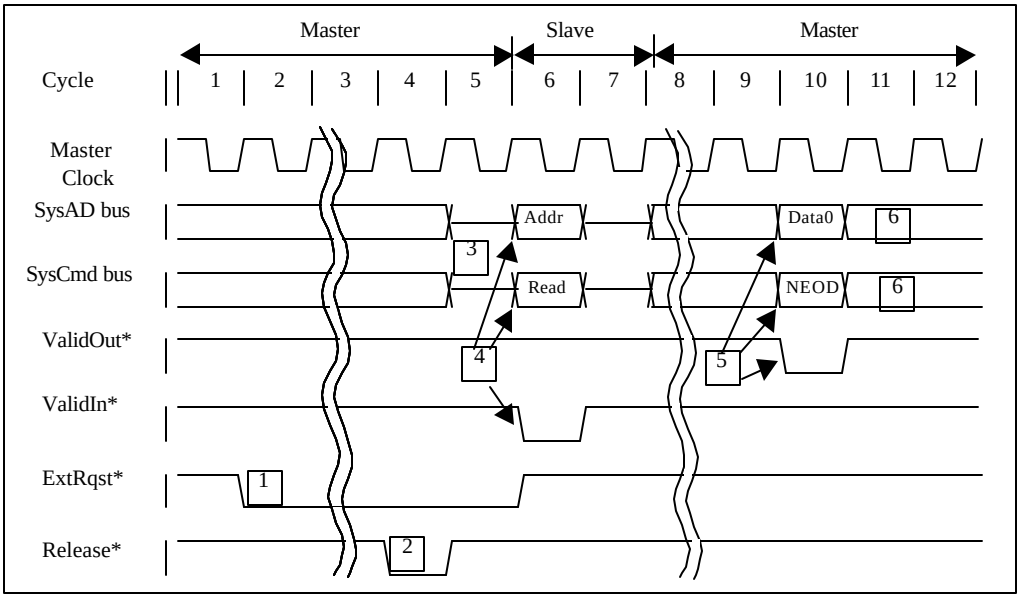


Figure 7-8 External Read Request when System Interface is in the Master State

Note: The processor contains no resources that can read by way of external read requests. The processor returns data identifiers with SysCmd(5) of the error data bits set along with undefined

data when it receives an external read request.

5.7.9 External null request protocol

The processor only supports one external null request. The system interface release external null request returns the system interface from the slave state to the master state. This request does not affect any other processors.

The only processing the external null request does is to have the processor return the system interface to the master state.

Figure 7-9 illustrates the timing of the external null request, which consists of the following steps.

1. The external agent requests use of the system interface by asserting ExtRqst*.
2. The processor asserts Release*, then the system interface is released from the processor and goes into the slave state.
3. The external agent drives the system interface, sends the external null request command to the SysCmd bus, then asserts the ValidIn* signal for 1 cycle.
4. The SysAD bus is not available during the external null request address cycle (there are no valid data in the bus).
5. The null request ends when the address cycle is issued.

In the case of a system interface release null request, the external agent releases the SysCmd bus and SysAD bus so the system interface can return to the master state. This protocol is the same for both the 32-bit and 64-bit bus modes.

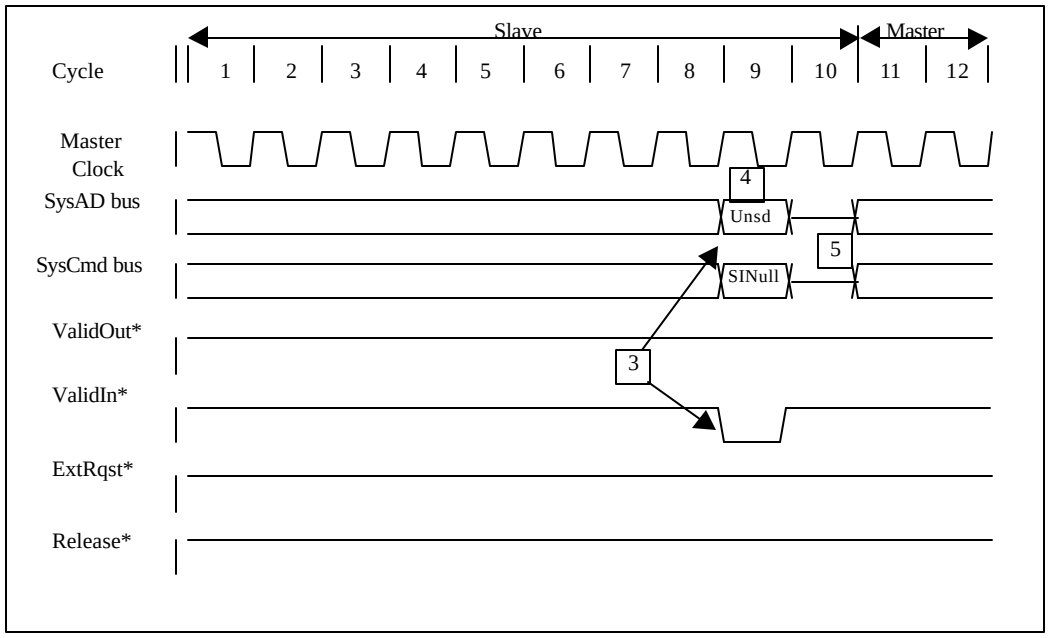


Figure 7-9 System Interface Release External Null Request

5.7.10 External write request protocol

The same protocol as the processor single word write protocol is used for external write requests, except when the ValidIn* signal is asserted instead of the ValidOut* signal.

Figure 7-10 illustrates the timing of the external write request, which consists of the following steps.

1. The external agent requests use of the system interface by asserting ExtRqst*.
2. The processor asserts Release*, then the system interface is released from the processor and goes into the slave state.
3. The external agent sends a write command to the SysCmd bus, and sends a write address to the SysAD bus while asserting ValidIn*.
4. The external agent sends data identifiers to the SysCmd bus, and sends data to the SysAD bus while asserting ValidIn*.
5. Data identifiers for this data cycle must contain an indication of a coherent or non-coherent final data cycle.
6. After a data cycle is issued, the write request is complete, the external agent sets the SysCmd and SysAD busses to tri-state, then the system interface returns to the master state. Timing of the SysADC bus and SysCmdP bus are each the same as the SysAD bus and SysCmd bus, respectively.

External write requests can write only 1 word of data to the processor. Operation of processors that have specified data elements other than a single word of data by an external write request is not defined.

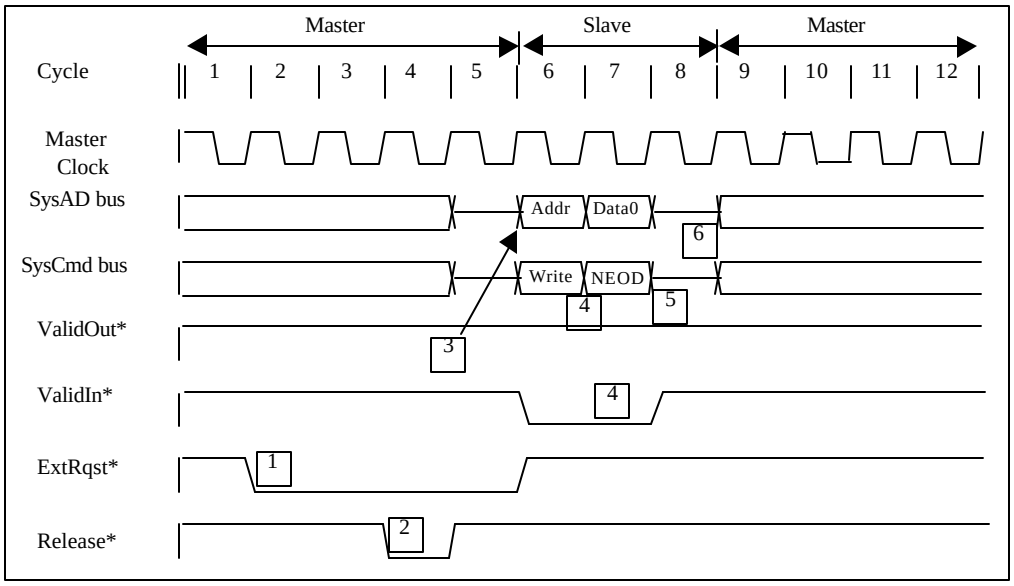


Figure 7-10 External Write Request when System Interface Starts in the Master State

5.7.11 Read response protocol

The external agent must use the read response protocol to return data to the processor if a processor read request has been received. The sequence of the read response protocol is as follows.

1. The external agent waits for the processor to automatically execute a shift into the slave state.
2. The processor uses either a single data cycle or a series of data cycles to return data.
3. After issuing the final data cycle, the read response ends, then the external agent sets the SysCmd and SysAD busses to tri-state.
4. The system interface returns to the master state.

Note: After issuing a read response, the processor automatically shifts to the slave state.

5. Data identifiers of a data cycle must indicate that the data are response data.
6. Final data cycle identifiers must contain an indication that a cycle is the final data cycle.

In the case of read responses to non-coherent block read requests, it is not necessary for the response data to check the initial cache state. The cache state is automatically set to exclusively dirty.

Data identifiers of data cycles can send notification of transfer data errors in those cycles. The external agent must return data blocks with the correct size even when there is an error in the data. Whether there is a single error or multiple errors in the read response data cycles, the processor processes them as bus errors.

Read responses must always be returned to the processor when a processor read request is being held. Processor operation is not defined if a read response was returned in a state where there were no processor read requests on hold.

Figure 7-11 illustrates a processor word read request and the subsequent word read response. Also, Figure 7-12 illustrates the read response to a processor block read request when the system interface is in the slave state.

Note: The SysADC bus and SysCmdP bus timing is the same as the SysAD bus and SysCmd timing, respectively.

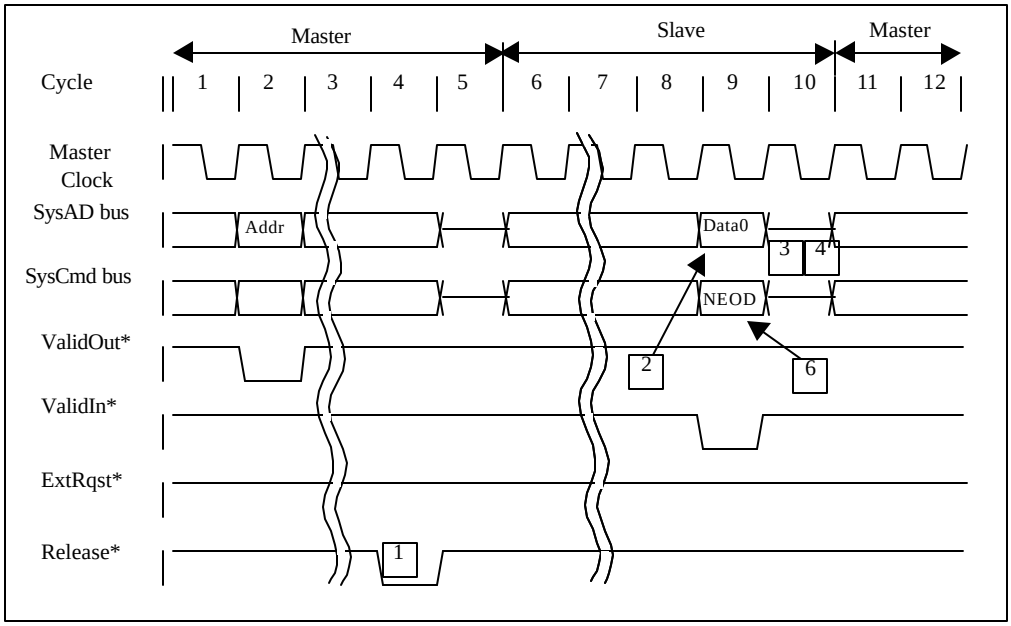


Figure 7-11 Processor Word Read Request and Subsequent Word Read Response

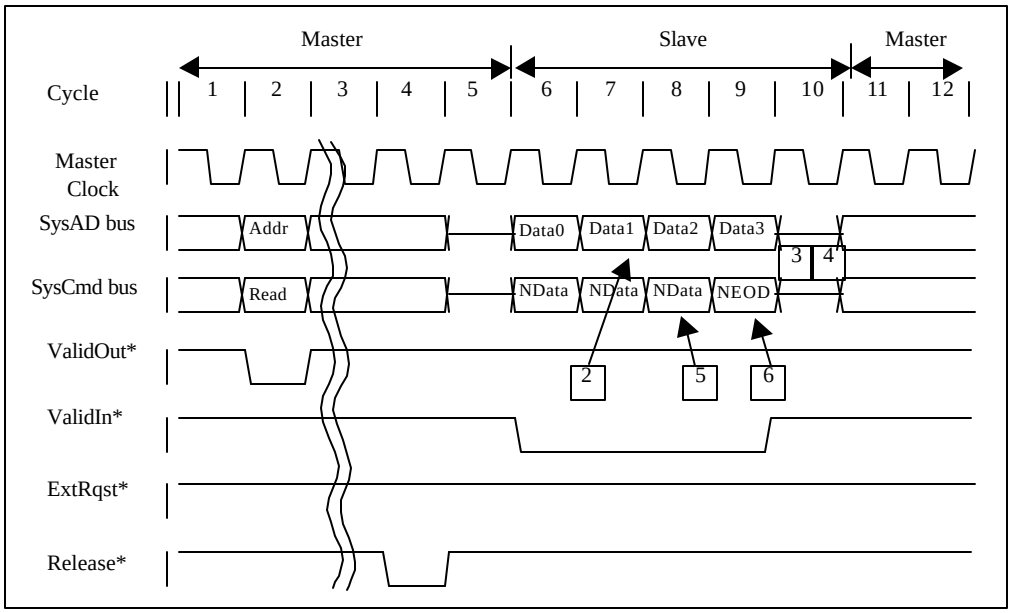


Figure 7-12 Block Read Response when System Interface is in the Slave State

5.8 Data Transfer

The maximum data transfer rate of the system interface is 1 double word per cycle.

The external agent can select the data transfer rate to the processor. For example, it is possible to transfer data and assert the ValidIn* signal just once for every *n* cycles instead of at all cycles. The external agent can transfer data to the processor at the selected transfer rate.

The processor interprets cycles as being valid when the ValidIn* signal is asserted and the SysCmd bus includes the data identifier. After a cycle has been interpreted as valid, the processor continues to accept data until a data word with an indication that it is the final data word appears.

5.8.1 Data transfer pattern

The term "data pattern" in the case of block write operations, refers to a string of text that indicates the "data" cycles repeated so that the appropriate data transfer speed can be obtained and the "unused" cycles. Dxx data patterns indicate a repetition data transfer rate where there is one double word in three cycles followed by two unused cycles. The data transfer rate is set by G2Sconfig register(0xF FF10 0000).

Table 8-1 indicates the data transfer rate, data pattern, and setting.

Maximum Data Transfer Rate	Data Pattern	Setting Bit(2:1)	Write mode
1 double word/3 Master Clock cycle	Dxx	00	R4000 compatible
Reserved	Reserved	01	Reserved
1 double word/1 Master Clock cycles	D	10	Pipeline write
1 double word/1 Master Clock cycles	D	11	Reissue write

Note: The setting Bit(2:1) is bit(2:1) of the G2Sconfig register (0xF FF10 0000).
Bit(2:1) is set to "00" when initialized.

Table 8-1 Data Transfer Rate, Data Pattern and Setting

5.8.2 Independent transfer on SysAD bus

A majority of applications connect the processor and external agent interior (both directions), and register format transceivers together in a point-to-point manner via the SysAD bus. The only two SysAD bus drives available for such applications are the processor and the external agent.

Depending on the application, it may be necessary to make additional connections on the SysAD bus for drivers and receivers to transfer data using the SysAD bus without involving the processor. Such transfers are referred to as independent transfers. In order to perform independent transfers, the external agent must use arbitration handshake signals and external null requests to properly tune SysAD bus control.

Independent transfer is performed on the SysAD bus according to the following steps.

1. The external agent requests access to the SysAD bus in order to issue an external request.
2. The processor releases the system interface and puts it in the slave state.
3. The external agent can independently transfer data using the SysAD bus. However, the ValidIn* signal must be asserted during that transfer.
4. When transfer is complete, the external agent must issue a system interface release null request and return the system interface to the master state.

5.9 System Interface Command and Data Identifier

In the case of a processor, there is a response time for each kind of processor transaction and for each external request. A minimum and maximum cycle count has been prescribed for each response time. Since processor requests themselves are restrained by system interface request protocols, checking the protocols makes it possible to determine the cycle count required for requests. The interval for the next interface operation is variable within the range of the minimum and maximum cycle counts.

- Stand-by time from when an external request is received and the processor releases the system interface, until when the interface enters the slave state (release latency).
- Response time to external request that requires a response (external response latency).

5.9.1 Release latency

Broadly defined, release latency is the number of cycles for which it is possible to wait from when the processor receives an external request until when the system interface is released and shifts to the slave state. If there are no processor requests currently in progress, the processor must delay release of the system interface for a few cycles since it is internal operation. Therefore, if release latency is strictly defined, it becomes the cycle count from when the ExtRqst* signal is asserted until when the Release* signal is asserted.

There are three types of release latency.

- Category 1: If external request signal is asserted 2 cycles before the final cycle of the processor request
- Category 2: If external request signal is asserted during processor request or is the final cycle even if it is asserted
- Category 3: If processor automatically shifts to the slave state

Table 9-1 indicates the minimum and maximum release latency inherent to categories 1, 2, and 3. However, note that these cycle counts may be changed at any time.

Category	Minimum cycle count	Maximum cycle count
1	3	5
2	1	24
3	0	0

Table 9-1 Release Latency for External Requests

5.10 System Interface Command and Data Identifiers

System interface commands specify the type of system interface and its properties. This specification is performed in the request address cycle. The system interface data identifiers specify the properties of the data transferred during the system interface data cycle.

Of the system interface commands and data identifiers used for external requests, set the bits and fields that are reserved to "1." In the case of system interface commands and data identifiers used for processor requests, bits, field contents and data identifiers that are reserved in the commands are undefined.

5.10.1 Syntax of commands and data identifiers

System interface commands and data identifiers consist of 9 bits. These commands are sent by the SysCmd bus from the processor to the external agent or from the external agent to the processor during either the address cycle or the data cycle. Bit 8 (MSB) of the SysCmd bus specifies whether the SysCmd contents at that time are a command or a data identifier (namely, whether the current cycle is an address cycle or a data cycle). If the contents are a system interface command, then SysCmd(8) must be set to "0." If the contents are a system interface data identifier, then SysCmd(8) must be set to "1."

5.10.2 Syntax of system interface commands

Following is an explanation of the SysCmd bus structure in the case of a system interface command. Figure 10-1 illustrates the structure common to all system interface commands.

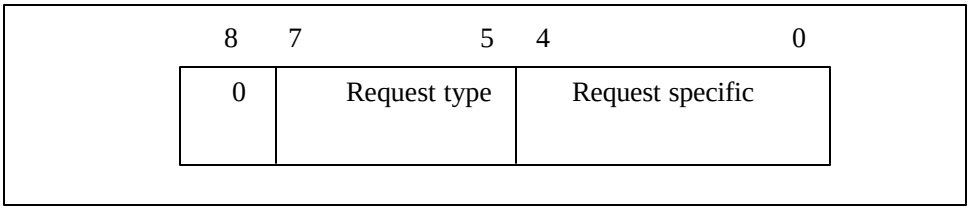


Figure 10-1 Syntax Bit Structure of System Interface Command

SysCmd(8) must always be set to "0" in the case of system interface commands.
SysCmd(7:5) specifies the type of system interface request (read, write, null). Table 10-1 indicates the SysCmd(7:5) specification method.

SysCmd(7:5)	Command
0	Read request
1	Reserve
2	Write request
3	Null request
4 – 7	Reserve

Table 10-1 SysCmd(7:5) Specification Method for System Interface Commands

SysCmd(4:0) varies depending on the type of request. Each specification type is indicated below.

5.10.3 Read requests

Figure 10-2 illustrates the SysCmd format in the case of read requests.

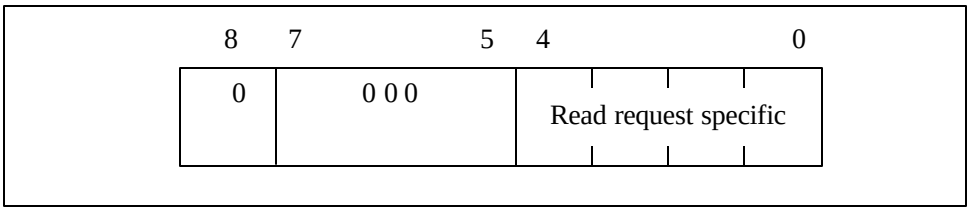


Figure 10-2 Bit Definition of SysCmd Bus for Read Requests

Tables 10-2, 10-3, and 10-4 indicate the methods for specifying SysCmd(4:0) for read requests.

SysCmd(4:3)	Read Properties
0 – 1	Reserved
2	Non-coherent block read
3	64-bit mode Double word, partial double word, word, or partial word read 32-bit mode Word or partial word read

Table 10-2 Specification Method of SysCmd(4:3) for Read Requests

SysCmd(2)	Reserved
SysCmd(1:0)	Read Block Size
0	Reserved
1	8 words (64-bit or 32-bit bus mode)
2 – 3	Reserved

Table 10-3 Specification Method of SysCmd(2:0) for Block Read Requests

SysCmd(2:0)	Read Data Size
0	1 byte valid (byte)
1	2 bytes valid (half-word)
2	3 bytes valid (tri-byte)
3	4 bytes valid (word)
4	5 bytes valid (quinti-byte)
5	6 bytes valid (sexi-byte)
6	7 bytes valid (septi-byte)
7	8 bytes valid (double word)

Note: 4 – 7 of SysCmd(2:0) is only valid when in the 64-bit bus mode.

Table 10-4 Data Size Expressed by SysCmd(2:0) for Double Word, Word, or Partial Word Read Requests

5.10.4 Write requests

Figure 10-3 illustrates the SysCmd format for write requests.

Table 10-5 indicates the methods of specifying write properties using SysCmd(4:3). Table 10-6 indicates the methods of specifying replacements properties using SysCmd(2:0) for block write requests. Table 10-7 indicates the methods of specifying the data size using SysCmd(2:0) for write requests.

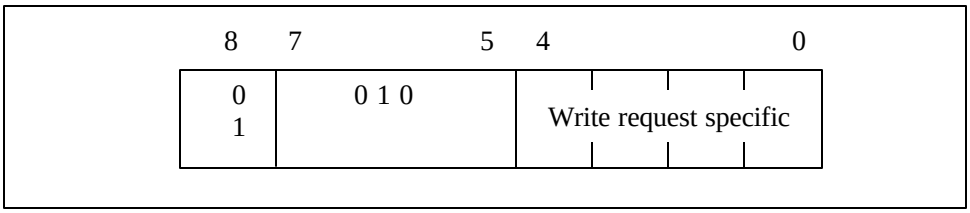


Figure 10-3 SysCmd Bus Bit Specification for Write Requests

SysCmd(4:3)	Write Properties
0 – 1	Reserved
2	Block write
3	64-bit mode Double word, partial double word, word, or partial word write 32-bit mode Word or partial word write

Table 10-5 Methods of Specifying SysCmd(4:3) for Write Requests

SysCmd(2)	Reserved
SysCmd(1:0)	Read Block Size
0	Reserved
1	8 words
2 – 3	Reserved

Table 10-6 Specification Method of SysCmd(2:0) for Block Write Requests

SysCmd(2:0)	Write Data Size
0	1 byte valid (byte)
1	2 bytes valid (half-word)
2	3 bytes valid (tri-byte)
3	4 bytes valid (word)
4	5 bytes valid (quinti-byte)
5	6 bytes valid (sexi-byte)
6	7 bytes valid (septi-byte)
7	8 bytes valid (double word)

Note: 4 – 7 of SysCmd(2:0) is only valid when in the 64-bit bus mode.

Table 10-7 Methods for Specifying SysCmd(2:0) for Block Write Requests

5.10.5 Null requests

Figure 10-4 illustrates the SysCmd format for null requests.

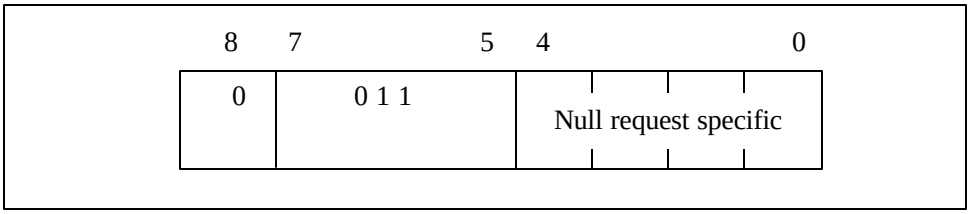


Figure 10-4 Bit Definition of the SysCmd Bus for Null Requests

Null request commands are always used for system interface release external null requests. Table 10-8 indicates methods of specifying SysCmd(4:3) for system interface release external null requests. SysCmd(2:0) is reserved for null requests.

SysCmd(4:3)	Null Properties
0	Release system interface
1 – 3	Reserved

Table 10-8 Method of Specifying SysCmd(4:3) for External Null Requests

5.10.6 Syntax of system interface data identifiers

This section defines methods of specifying the SysCmd bus for system interface identifiers. The bit structure illustrated in Figure 10-5 is common to all system interface data identifiers.

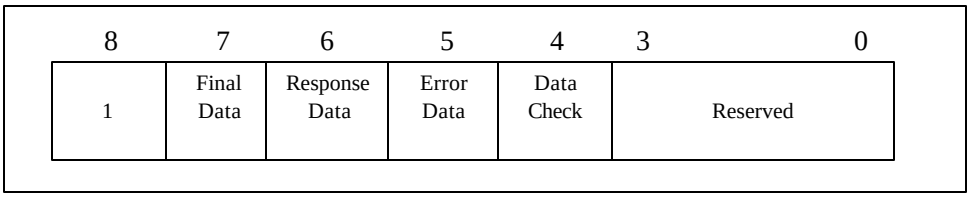


Figure 10-5 Bit Definition of SysCmd Bus for Data Identifiers

SysCmd(8) must always be set to "1" for system interface data identifiers. System interface data identifiers are in the non-coherent data format.

5.10.7 Non-coherent data

Non-coherent data are data such as the following.

- Data that are the subject of a processor block write request or a processor double word/partial double word/word/partial word write request.
- Data that are returned to a processor non-coherent block read request or a processor double word/ partial double word/word/partial word read request.
- Data that are the subject of an external write request.
- Data that are returned to an external read request as a response.

5.10.8 Bit definition of data identifiers

In the case of processor or external coherent data identifiers and processor or external non-coherent data identifiers, SysCmd(7) indicates that they are the final data element, and SysCmd(6) indicates whether they are response data. Response data are the data that are returned to a read request as a response.

SysCmd(5) indicates whether there is an error in a data element. Uncorrectable errors are included in the error data. When such an error is returned to the processor, a bus error is generated. If a primary parity error is detected in the data items to be transferred, the processor deasserts the good data bits and sends data.

SysCmd(4) indicates to the processor whether the data bits and check bits of a data element should be searched.

SysCmd(3) is reserved in the case of external data identifiers.

SysCmd(4:3) is reserved in the case of non-coherent processor data identifiers.

SysCmd(2:0) is reserved in the case of non-coherent data identifiers.

Table 10-9 indicates methods of specifying SysCmd(7:3) for processor data identifiers. Table 10-10 indicates methods of specifying SysCmd(7:3) for external data identifiers.

SysCmd(7)	Final Data Element Indication
0	Final data element
1	Is not final data element
SysCmd(6)	Response Data Indication
0	Response data
1	Is not response data
SysCmd(5)	Good Data Indication
0	No errors
1	Is error data
SysCmd(4:3)	Reserved

Table 10-9 Methods of Specifying SysCmd(7:3) for Processor Data Identifiers

SysCmd(7)	Final Data Element Indication
0	Final data element
1	Is not final data element
SysCmd(6)	Response Data Indication
0	Response data
1	Is not response data
SysCmd(5)	Good Data Indication
0	No errors
1	Is error data
SysCmd(4)	Enable Data Check
0	Check data bits and check bits
1	Do not check data bits and check bits
SysCmd(3)	Reserved

Table 10-10 Method of Specifying SysCmd(7:3) for External Data Identifiers

5.11 System Interface Addresses

System interface addresses in the 64-bit bus mode are complete 36-bit physical addresses. They are sent to the lower 36 bits (bits 35 – 0) of the SysAD bus during the address cycle. The remaining bits of the SysAD bus are not used by the address cycle.

System interface addresses in the 32-bit bus mode are complete 32-bit physical addresses. They are sent to the lower 32 bits of the SysAD bus during the address cycle.

5.11.1 Addressing rules in the 64-bit bus mode

Addresses that are to be used in double word, partial double word, word or partial word transactions are arranged to match the size of the data element. The following rules are used by this system.

- Target addresses of the block request are aligned to the double word boundaries. In other words, the lower 3 bits of the address become "0."
- For double word requests, the lower 3 bits of the address are set to "0."
- Word requests set the lower 2 bits of the address to "0."
- Half word requests set the least significant bit of the address to "0."
- Byte requests, 3-byte requests, 5-byte requests, 6-byte requests and 7-byte requests use the byte address.

5.11.2 Addressing rules in the 32-bit bus mode

Addresses that are to be used in word or partial word transactions are arranged to match the size of the data element. The following rules are used by this system.

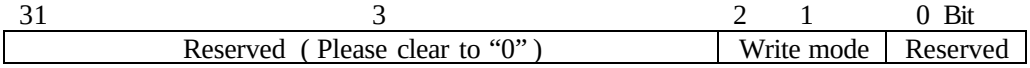
- Target addresses of the block request are aligned to the word boundaries. In other words, the lower 2 bits of the address become "0."
- Word requests set the lower 2 bits of the address to "0."
- Half-word requests set the least significant bit of the address to "0."
- Byte requests and 3-byte requests use the byte address.

5.12 Mode Register of System Interface (G2Sconfig)

The Mode Register of System Interface (G2Sconfig) is a write only register. This register is only Word-Access.

Address	Field	Description
0xF_FF10_0000	G2Sconfig	Mode Register of System Interface

Table 12-1 G2Sconfig



Bit	Field	Description	ColReset	Read/Write
31..3	0	Reserved	Undefined	Write only
2..1	Write mode	Processor write protocols set 00 : R4000 compatible write 01 : Reserved 10 : Pipeline write 11 : Reissue write (single write and block write are same protocols)	00	Write only
0	0	Reserved	Undefined	Write only

Note: Please the Bit 31..3 and Bit0 clear to 0.

Figure 12-1 G2Sconfig Register Formats

6. ELECTRICAL CHARACTERISTICS

6.1 ABSOLUTE MAXIMUM RATINGS (TX4955)

TMPR4955F-167/200

V_{SS} = 0 V (GND)

PARAMETER	SYMBOL	RATINGS	UNIT
Supply voltage (for I/O)	VccIOMax	-0.5 T.B.D.	V
Supply voltage (for internal)	VccIntMax	-0.5 T.B.D.	V
Input voltage(*1)	V_{IN}	-0.5 to V_{CC} + 0.5	V
Storage Temperature	T_{STG}	-65 to +150	°C

Note) If LSI is used above the maximum ratings, permanent destruction of LSI can result. In addition, it is desirable to use LSI for normal operation under the recommended condition. If these conditions are exceeded, reliability of LSI may be adversely affected.

(*1) VIN Min. = -1.5V for pulse width less than 10 ns.

6.2 RECOMMENDED OPERATING CONDITIONS (TX4955)

TMPR4955F-167/200

V_{SS} = 0 V (GND)

PARAMETER	SYMBOL	CONDITIONS	MIN.	MAX.	Unit
Supply Voltage (for I/O)	VccIO		3.135	3.465	V
Supply Voltage (for internal)	VccInt		2.375	2.625	V
Operating Case Temperature	T_C		0	+70	°C

6.3 DC CHARACTERISTICS (TX4955)

TMPR4955F-167/200

T_C = 0°C to 70°C, VccInt = 2.5V± 5%,VccIO = 3.3 V ± 5%

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNITS
Output High Voltage	V_{OH}	VccIO = 3.3V,Vss=0V I_{OH} = -4 mA	2.4			V
Output Low Voltage	V_{OL}	VccIO = 3.3V,Vss=0V I_{OL} = 4 mA			0.4	V
Input High Voltage (*2)	V_{IH}		2.0		VccIO+ 0.3	V
Input Low Voltage (*1,2)	V_{IL}		-0.5 (*1)		0.8	V
Operating Current	I_{CC}	VccIO = 3.465V, VccInt = 2.625V, MasterClock=83.5MHz			T.B.D.	A
Standby Current	Iccs	VccIO = 3.465V, VccInt = 2.625V,			T.B.D.	A
Input Leakage	I_{LI}	Except (*3)port			±10	µA
Pull-up (*3)	Rinu			100	T.B.D.	K
Pull-down (*4)	Rind			100	T.B.D.	K
Output Leakage	I_{LO}				±20	µA
Input Capacitance	C_{IN}				10	pF
Output Capacitance	C_{OUT}				10	pF

(*1) V_{IL} Min. = -1.5V for pulse width less than 10 ns.

(*2) Except for MasterClock input

(*3) Applies to Int(5:0)*,NMI*, RESET*,JTMS, JTCK, JTDI, DivMode(1:0),TPC1 inputs

(*4) Applies to TRST*,RdRdy*,TPC3,TPC2 inputs

6.4 AC CHARACTERISTICS (TX4955)

6.4.1 CLOCK TIMING

TMPR4955F-167/200

T_C = 0°C to 70°C, VccInt = 2.5V±5%,VccIO = 3.3 V ± 5%

PARAMETER	SYMBOL	CONDITIONS	MIN.	MAX.	UNITS
MasterClock High	t_{MCH}	Transition 5 ns	3.0		ns
MasterClock Low	t_{MCL}	Transition 5 ns	3.0		ns
MasterClock Frequency (*1)	f_{MCK}		20	83.5/100	MHz
Internal Operation Frequency	f_{PCK}		50	167/200	MHz
MasterClock Period	t_{MCP}		10	50	ns
MasterClock Rise Time	t_{MCR}			2.0	ns
MasterClock Fall Time	t_{MCF}			2.0	ns

(*1) Operation of TMPR495F is only guaranteed with the Phase Lock Loop enabled.

(*2) All output timings assume a 50 pF capacitive load. Output timings should be derated where appropriate.

6.4.2 SYSTEM INTERFACE

TMPR4955F-167/200

T_C = 0°C to 70°C, VccInt = 2.5V±5%,VccIO = 3.3 V ± 5%

PARAMETER	SYMBOL	MIN.	MAX.	UNITS
Data Output (*1,2,3)	t_{DO}	1.0	5.0	ns
Data Setup (*3)	t_{DS}	2.5		ns
Data Hold (*3)	t_{DH}	1.0		ns

(*1) Timings are measured from 1.5V of the SClock to 1.5V of signal.

(*2) Capacitive load for all output timings is 50 pF.

(*3) Data Output, Data Setup and Data Hold apply to all logic signals driven out of or driven into the TMPR4955F on the system interface. Clocks are specified separately.

6.4.3 CAPACITIVE LOAD DURATION

T_C = 0°C to 70°C, VccInt = 2.5V±5%,VccIO = 3.3 V ± 5%

PARAMETER	SYMBOL	MIN.	MAX.	UNITS
Load Duration	C_{LD}		2.0	ns / 25 pF

6.5 ABSOLUTE MAXIMUM RATINGS (TX4956)

TMPR4956F-167/200

V_{SS} = 0 V (GND)

PARAMETER	SYMBOL	RATINGS	UNIT
Supply voltage (for I/O)	VccIOMax	-0.5 T.B.D.	V
Supply voltage (for internal)	VccIntMax	-0.5 T.B.D.	V
Input voltage(*1)	V_{IN}	-0.5 to V_{CC} + 0.5	V
Storage Temperature	T_{STG}	-65 to +150	°C

Note) If LSI is used above the maximum ratings, permanent destruction of LSI can result. In addition, it is desirable to use LSI for normal operation under the recommended condition. If these conditions are exceeded, reliability of LSI may be adversely affected.

(*1) VIN Min. = -1.5V for pulse width less than 10 ns.

6.6 RECOMMENDED OPERATING CONDITIONS (TX4956)

TMPR4956F-167/200

V_{SS} = 0 V (GND)

PARAMETER	SYMBOL	CONDITIONS	MIN.	MAX.	Unit
Supply Voltage (for I/O)	VccIO		3.135	3.465	V
Supply Voltage (for internal)	VccInt		2.375	2.625	V
Operating Case Temperature	T_C		0	+70	°C

6.7 DC CHARACTERISTICS (TX4956)

TMPR4956F-167/200

T_C = 0°C to 70°C, VccInt = 2.5V±5%,VccIO = 3.3 V ± 5%

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNITS
Output High Voltage	V_{OH}	VccIO = 3.3V,Vss=0V I_{OH} = -4 mA	2.4			V
Output Low Voltage	V_{OL}	VccIO = 3.3V,Vss=0V I_{OL} = 4 mA			0.4	V
Input High Voltage (*2)	V_{IH}		2.0		VccIO+ 0.3	V
Input Low Voltage (*1,2)	V_{IL}		-0.5 (*1)		0.8	V
Operating Current	I_{CC}	VccIO = 3.465V, VccInt = 2.625V, MasterClock=83.5MHz			T.B.D.	A
Standby Current	Iccs	VccIO = 3.465V, VccInt = 2.625V,			T.B.D.	A
Input Leakage	I_{LI}	Except (*3)port			±10	µA
Pull-up (*3)	Rinu			100	T.B.D.	K
Pull-down (*4)	Rind			100	T.B.D.	K
Output Leakage	I_{LO}				±20	µA
Input Capacitance	C_{IN}				10	pF
Output Capacitance	C_{OUT}				10	pF

(*1) V_{IL} Min. = -1.5V for pulse width less than 10 ns.

(*2) Except for MasterClock input .

(*3) Applies to Int(5:0)*,NMI*, RESET*,JTMS, JTCK, JTDI, DivMode(1:0),TPC1 inputs .

(*4) Applies to TRST*,RdRdy*,TPC3,TPC2 inputs .

6.8 AC CHARACTERISTICS (TX4956)

6.8.1 CLOCK TIMING

TMPR4956F-167/200

T_C = 0°C to 70°C, VccInt = 2.5V±5%,VccIO = 3.3 V ± 5%

PARAMETER	SYMBOL	CONDITIONS	MIN.	MAX.	UNITS
MasterClock High	t_{MCH}	Transition 5 ns	3.0		ns
MasterClock Low	t_{MCL}	Transition 5 ns	3.0		ns
MasterClock Frequency (*1)	f_{MCK}		20	83.5/100	MHz
Internal Operation Frequency	f_{PCK}		50	167/200	MHz
MasterClock Period	t_{MCP}		10	50	ns
MasterClock Rise Time	t_{MCR}			2.0	ns
MasterClock Fall Time	t_{MCF}			2.0	ns

(*1) Operation of TMPR495F is only guaranteed with the Phase Lock Loop enabled.

(*2) All output timings assume a 50 pF capacitive load. Output timings should be derated where appropriate.

6.8.2 SYSTEM INTERFACE

TMPR4956F-167/200

T_C = 0°C to 70°C, VccInt = 2.5V±5%,VccIO = 3.3 V ± 5%

PARAMETER	SYMBOL	MIN.	MAX.	UNITS
Data Output (*1,2,3)	t_{DO}	1.0	5.0	ns
Data Setup (*3)	t_{DS}	2.5		ns
Data Hold (*3)	t_{DH}	1.0		ns

(*1) Timings are measured from 1.5V of the SClock to 1.5V of signal.

(*2) Capacitive load for all output timings is 50 pF.

(*3) Data Output, Data Setup and Data Hold apply to all logic signals driven out of or driven into the TMPR4956F on the system interface. Clocks are specified separately.

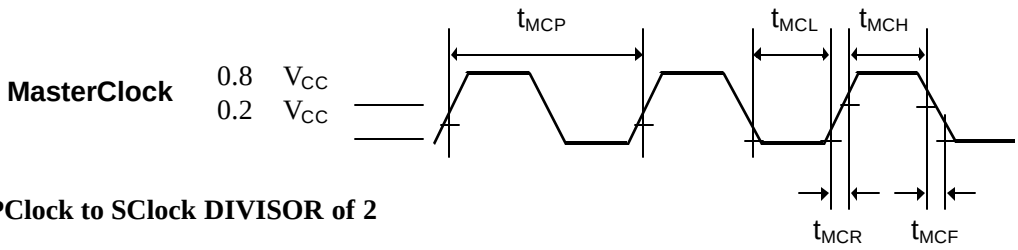
6.8.3 CAPACITIVE LOAD DURATION

T_C = 0°C to 70°C, VccInt = 2.5V±5%,VccIO = 3.3 V ± 5%

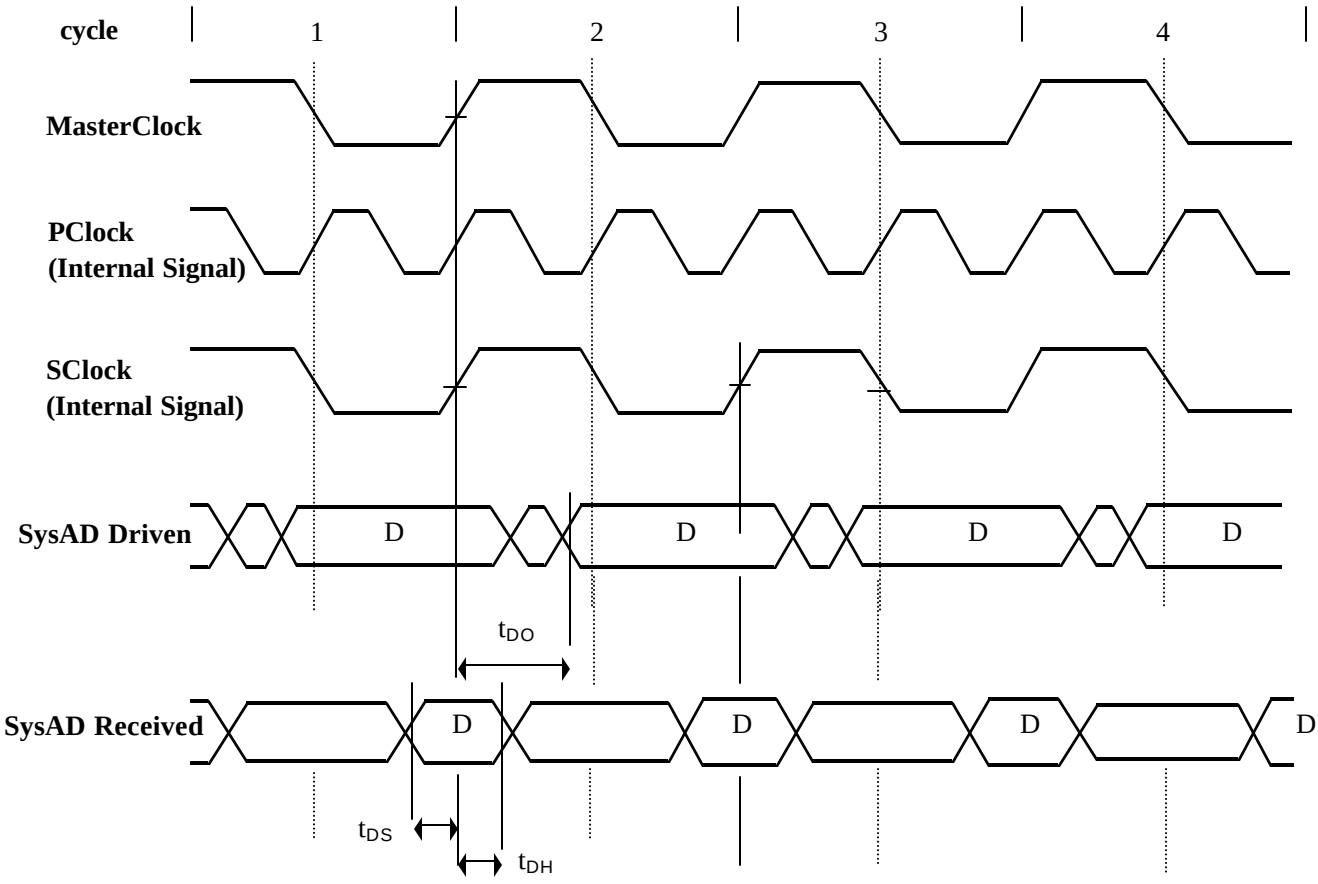
PARAMETER	SYMBOL	MIN.	MAX.	UNITS
Load Duration	C_{LD}		2.0	ns / 25 pF

6.9 TIMING DIAGRAMS

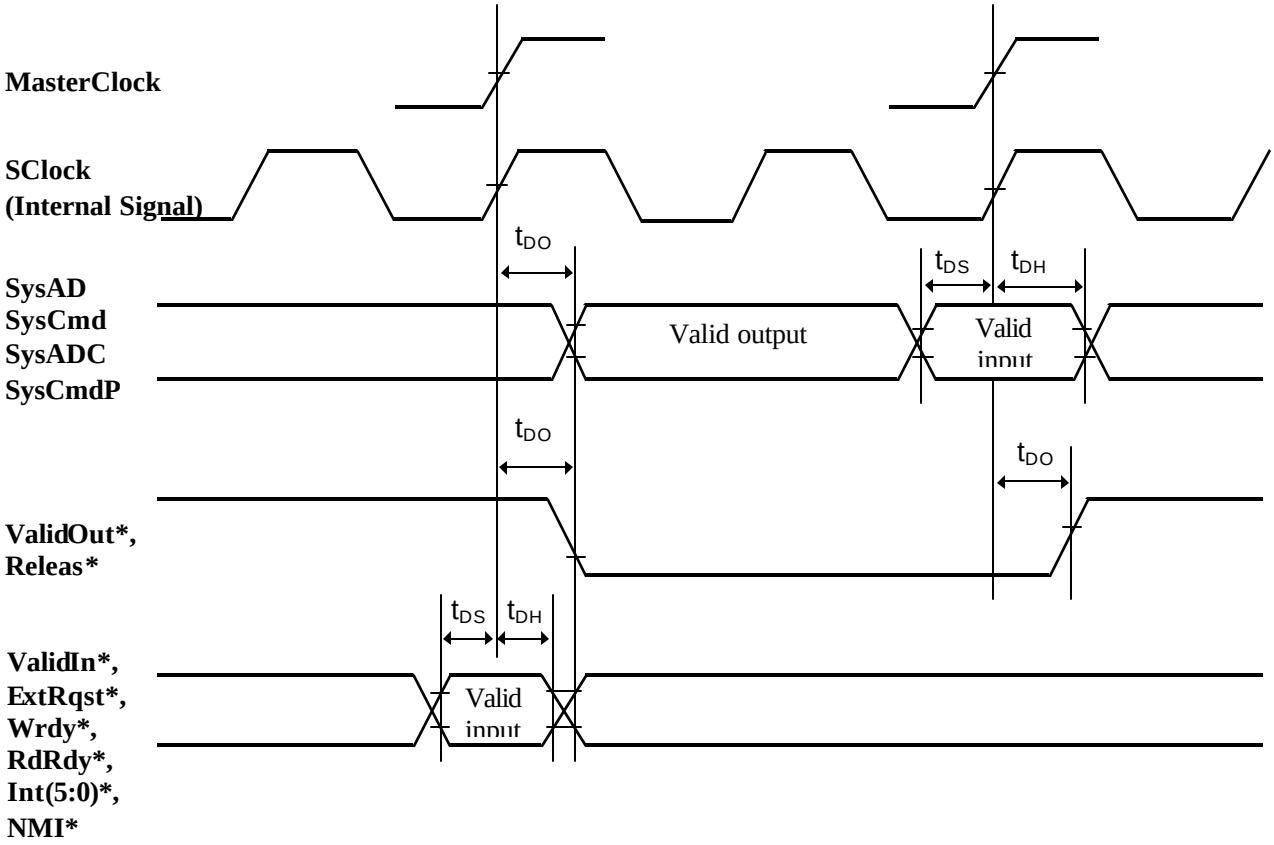
6.9.1 CLOCK TIMING



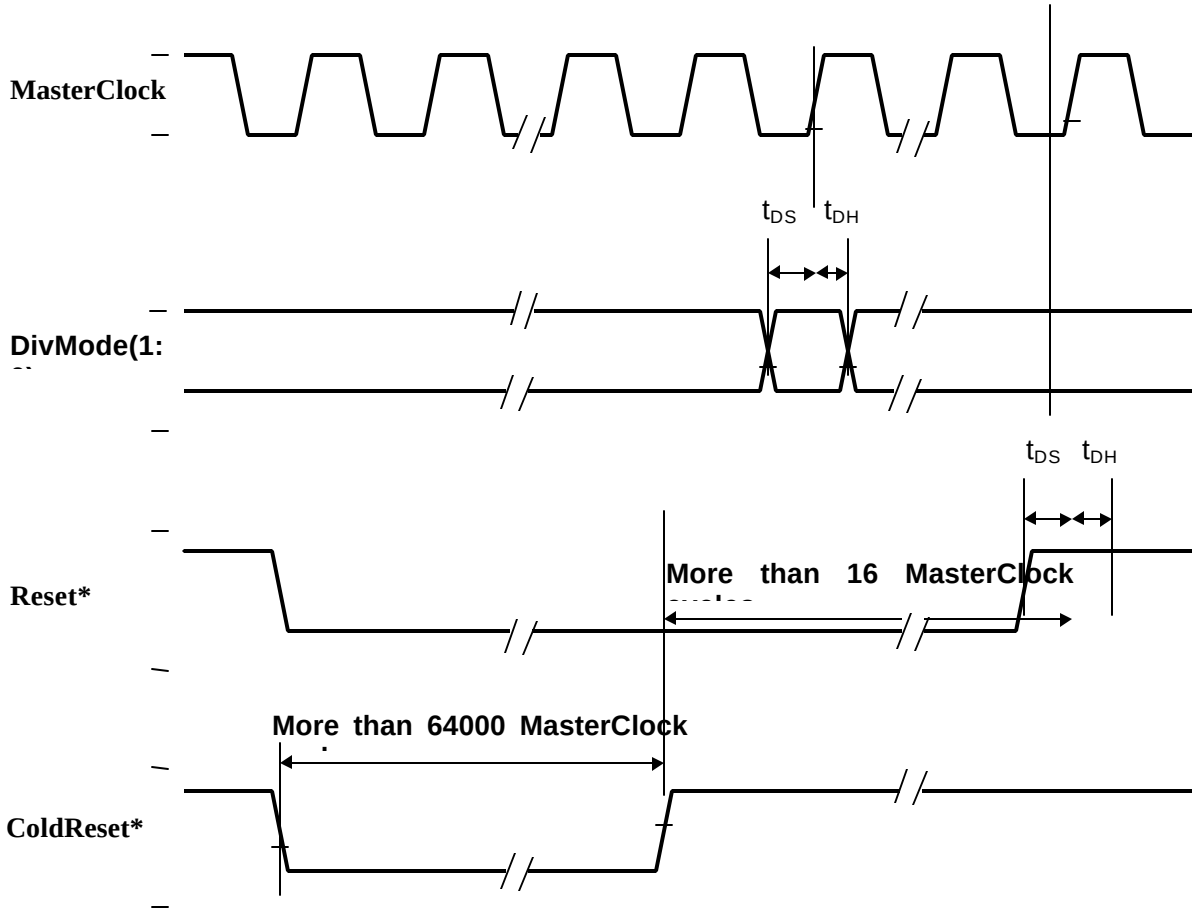
6.9.2 PClock to SClock DIVISOR of 2



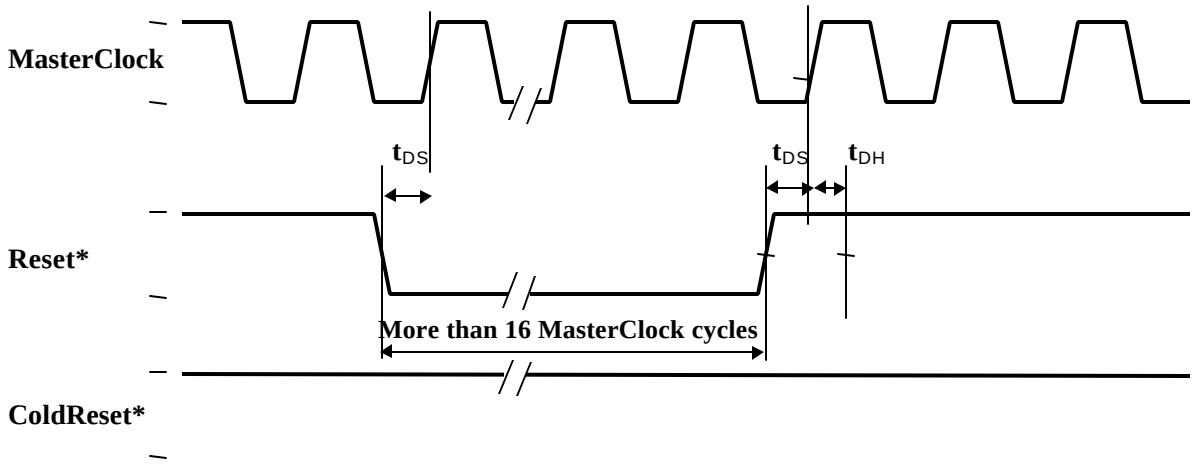
6.9.3 SYSTEM INTERFACE TIMING



6.9.4 COLD RESET TIMING

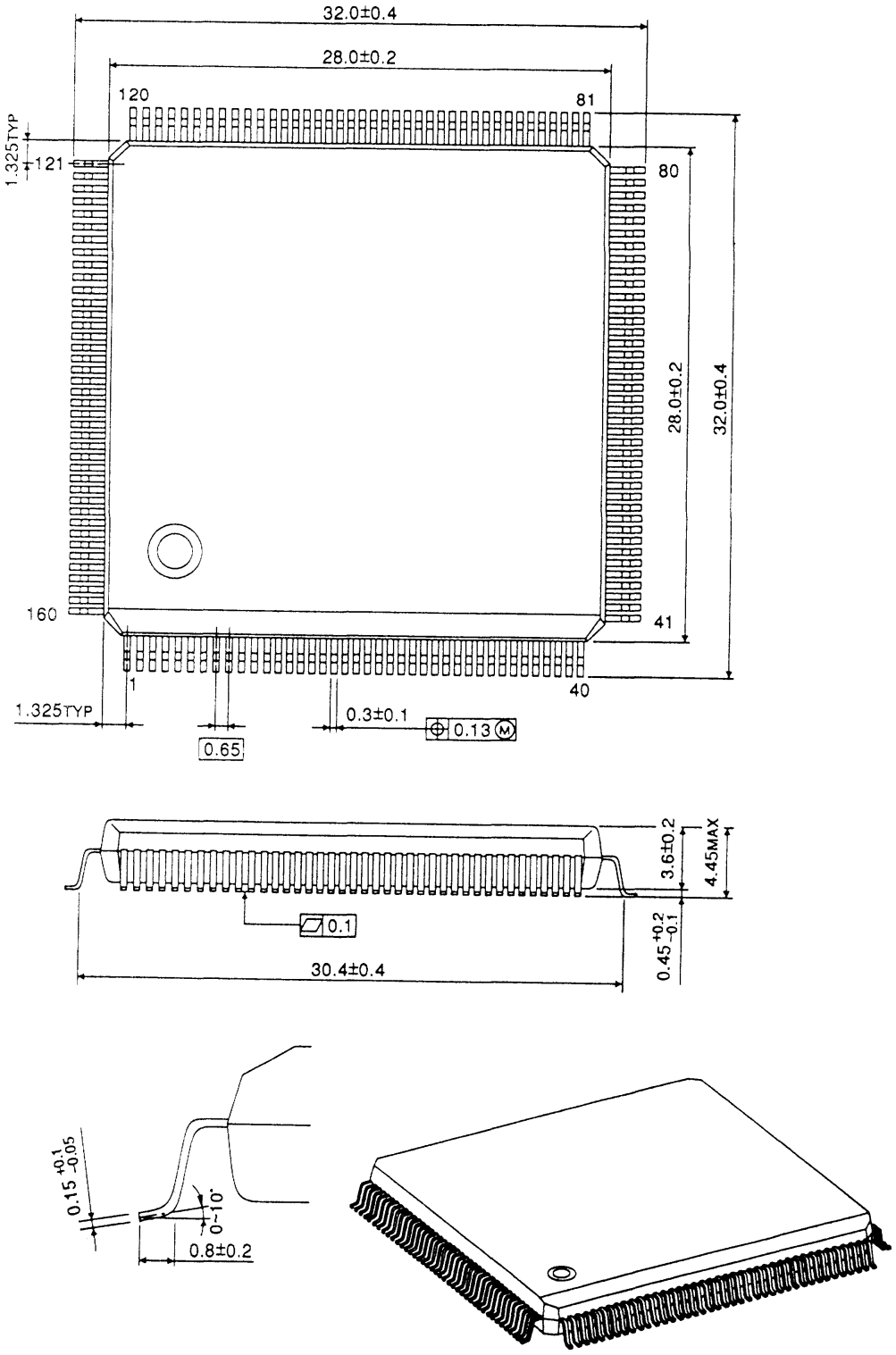


6.9.5 WARM RESET TIMING

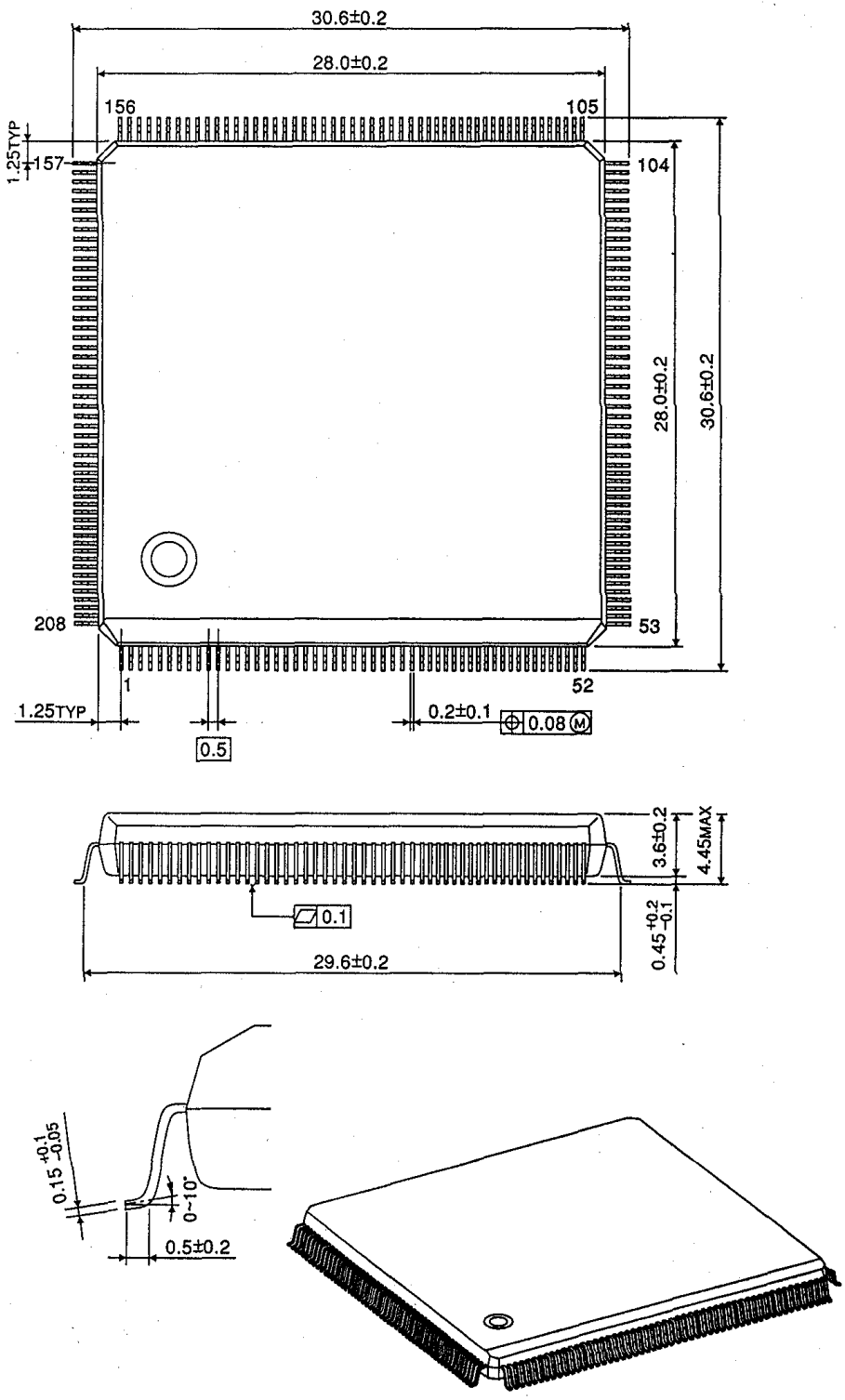


7. PACKAGE DIMENSION

7.1 TX4955 QFP160-P-2828-0.65A UNIT : mm

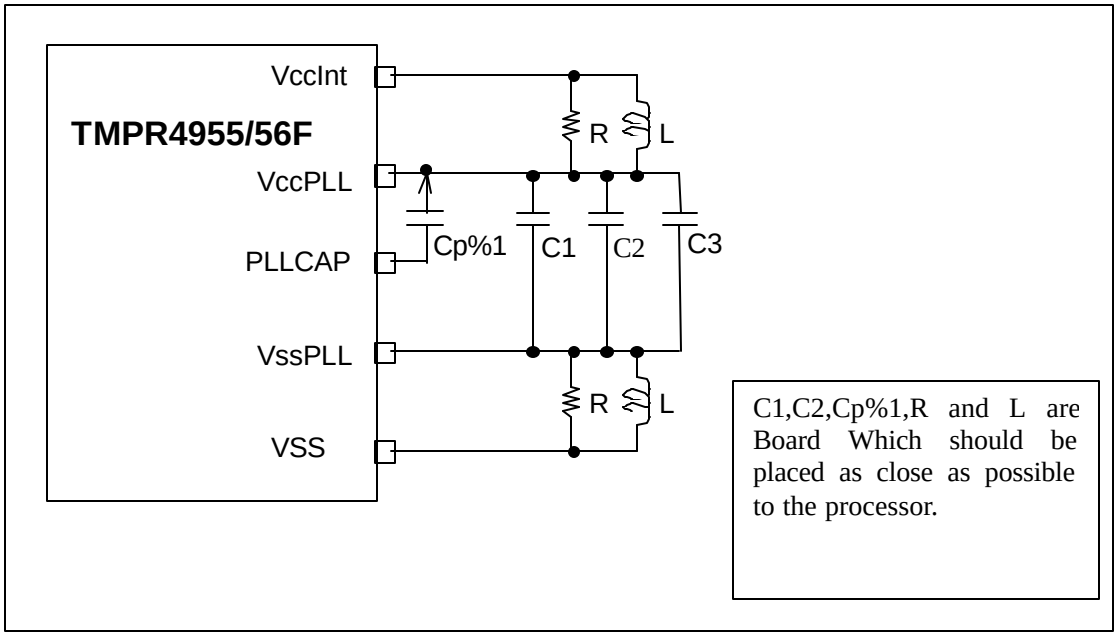


7.2 TX4956 QFP208-P-2828-0.50 **UNIT : mm**



Appendix A. Passive Components

The Phase Locked Loop circuit requires several components for proper operation, which are connected to PLLCAP,VccPLL, and VssPLL, as illustrated in Figure C-1.
In addition, the capacitors for PLLCAP (Cp%1) can be connected to VccPLL.



Values (TENTATIVE) :

R = 5 ohm

C1 = 1 nF

C2 = 82 nF

C3 = 10 μF

Cp%1 = 900 pF

VccInt = 2.5v ± 5%

The inductors (L) can be used as alternatives to the (R) to filter the power supply.

It is essential to isolate the analog power and ground for the PLL circuit (VccPLL/VssPLL) from the regular power ground (VccIO/VccINT/Vss).