TOSHIBA

64-Bit TX System RISC
TX49 Family

TMPR4955A

TOSHIBA CORPORATION

R4000/R4400/R5000 are a trademark of MIPS Technologies, Inc.

Preface

Thank you for your new or continued patronage of Toshiba semiconductor products. This is the 2001 edition of the user's manual for the TMPR4955A 64-bit RISC microprocessor.

This databook is written so as to be accessible to engineers who may be designing a Toshiba microprocessor into their products for the first time. No prior knowledge of this device is assumed. What we offer here is basic information about the microprocessor, a discussion of the application fields in which the microprocessor is utilized, and an overview of design methods. On the other hand, the more experienced designer will find complete technical specifications for this product.

Toshiba continually updates its technical information. Your comments and suggestions concerning this and other Toshiba documents are sincerely appreciated and may be utilized in subsequent editions. For updating of the data in this manual, or for additional information about the product appearing in it, please contact your nearest Toshiba office or authorized Toshiba dealer.

Dec 2000

Contents

Handling Precautions

TMPR4955A

Handling Precautions

1.   Using Toshiba Semiconductors Safely

TOSHIBA are continually working to improve the quality and the reliability of their products.

Nevertheless, semiconductor devices in general can malfunction or fail due to their inherent electrical sensitivity and vulnerability to physical stress. It is the responsibility of the buyer, when utilizing TOSHIBA products, to observe standards of safety, and to avoid situations in which a malfunction or failure of a TOSHIBA product could cause loss of human life, bodily injury or damage to property.

In developing your designs, please ensure that TOSHIBA products are used within specified operating ranges as set forth in the most recent products specifications. Also, please keep in mind the precautions and conditions set forth in the TOSHIBA Semiconductor Reliability Handbook.

2.   Safety Precautions

This section lists important precautions which users of semiconductor devices (and anyone else) should observe in order to avoid injury and damage to property, and to ensure safe and correct use of devices.

Please be sure that you understand the meanings of the labels and the graphic symbol described below before you move on to the detailed descriptions of the precautions.

[Explanation of labels]

⚠ DANGER	Indicates an imminently hazardous situation which will result in death or serious injury if you do not follow instructions.
⚠ WARNING	Indicates a potentially hazardous situation which could result in death or serious injury if you do not follow instructions.
⚠ CAUTION	Indicates a potentially hazardous situation which if not avoided, may result in minor injury or moderate injury.

[Explanation of graphic symbol]

Graphic symbol	Meaning
⚠ (laser)	Indicates that caution is required (laser beam is dangerous to eyes).

2.1   General Precautions regarding Semiconductor Devices

⚠CAUTION

Do not use devices under conditions exceeding their absolute maximum ratings (e.g. current, voltage, power dissipation or temperature).
This may cause the device to break down, degrade its performance, or cause it to catch fire or explode resulting in injury.

Do not insert devices in the wrong orientation.
Make sure that the positive and negative terminals of power supplies are connected correctly. Otherwise the rated maximum current or power dissipation may be exceeded and the device may break down or undergo performance degradation, causing it to catch fire or explode and resulting in injury.

When power to a device is on, do not touch the device's heat sink.
Heat sinks become hot, so you may burn your hand.

Do not touch the tips of device leads.
Because some types of device have leads with pointed tips, you may prick your finger.

When conducting any kind of evaluation, inspection or testing, be sure to connect the testing equipment's electrodes or probes to the pins of the device under test before powering it on.
Otherwise, you may receive an electric shock causing injury.

Before grounding an item of measuring equipment or a soldering iron, check that there is no electrical leakage from it.
Electrical leakage may cause the device which you are testing or soldering to break down, or could give you an electric shock.

Always wear protective glasses when cutting the leads of a device with clippers or a similar tool.
If you do not, small bits of metal flying off the cut ends may damage your eyes.

2.2   Precautions Specific to Each Product Group

2.2.1   Optical semiconductor devices

⚠ DANGER

When a visible semiconductor laser is operating, do not look directly into the laser beam or look through the optical system.
This is highly likely to impair vision, and in the worst case may cause blindness.
If it is necessary to examine the laser apparatus, for example to inspect its optical characteristics, always wear the appropriate type of laser protective glasses as stipulated by IEC standard IEC825-1.

⚠ WARNING

Ensure that the current flowing in an LED device does not exceed the device's maximum rated current.
This is particularly important for resin-packaged LED devices, as excessive current may cause the package resin to blow up, scattering resin fragments and causing injury.

When testing the dielectric strength of a photocoupler, use testing equipment which can shut off the supply voltage to the photocoupler. If you detect a leakage current of more than 100 μA, use the testing equipment to shut off the photocoupler's supply voltage; otherwise a large short-circuit current will flow continuously, and the device may break down or burst into flames, resulting in fire or injury.

When incorporating a visible semiconductor laser into a design, use the device's internal photodetector or a separate photodetector to stabilize the laser's radiant power so as to ensure that laser beams exceeding the laser's rated radiant power cannot be emitted.
If this stabilizing mechanism does not work and the rated radiant power is exceeded, the device may break down or the excessively powerful laser beams may cause injury.

2.2.2   Power devices

⚠ DANGER

Never touch a power device while it is powered on. Also, after turning off a power device, do not touch it until it has thoroughly discharged all remaining electrical charge.
Touching a power device while it is powered on or still charged could cause a severe electric shock, resulting in death or serious injury.

When conducting any kind of evaluation, inspection or testing, be sure to connect the testing equipment's electrodes or probes to the device under test before powering it on.
When you have finished, discharge any electrical charge remaining in the device.
Connecting the electrodes or probes of testing equipment to a device while it is powered on may result in electric shock, causing injury.

⚠WARNING

Do not use devices under conditions which exceed their absolute maximum ratings (current, voltage, power dissipation, temperature etc.).
This may cause the device to break down, causing a large short-circuit current to flow, which may in turn cause it to catch fire or explode, resulting in fire or injury.

Use a unit which can detect short-circuit currents and which will shut off the power supply if a short-circuit occurs.
If the power supply is not shut off, a large short-circuit current will flow continuously, which may in turn cause the device to catch fire or explode, resulting in fire or injury.

When designing a case for enclosing your system, consider how best to protect the user from shrapnel in the event of the device catching fire or exploding.
Flying shrapnel can cause injury.

When conducting any kind of evaluation, inspection or testing, always use protective safety tools such as a cover for the device.
Otherwise you may sustain injury caused by the device catching fire or exploding.

Make sure that all metal casings in your design are grounded to earth.
Even in modules where a device's electrodes and metal casing are insulated, capacitance in the module may cause the electrostatic potential in the casing to rise.
Dielectric breakdown may cause a high voltage to be applied to the casing, causing electric shock and injury to anyone touching it.

When designing the heat radiation and safety features of a system incorporating high-speed rectifiers, remember to take the device's forward and reverse losses into account.
The leakage current in these devices is greater than that in ordinary rectifiers; as a result, if a high-speed rectifier is used in an extreme environment (e.g. at high temperature or high voltage), its reverse loss may increase, causing thermal runaway to occur.
This may in turn cause the device to explode and scatter shrapnel, resulting in injury to the user.

A design should ensure that, except when the main circuit of the device is active, reverse bias is applied to the device gate while electricity is conducted to control circuits, so that the main circuit will become inactive.
Malfunction of the device may cause serious accidents or injuries.

⚠CAUTION

When conducting any kind of evaluation, inspection or testing, either wear protective gloves or wait until the device has cooled properly before handling it.
Devices become hot when they are operated. Even after the power has been turned off, the device will retain residual heat which may cause a burn to anyone touching it.

2.2.3 Bipolar ICs (for use in automobiles)

⚠CAUTION

If your design includes an inductive load such as a motor coil, incorporate diodes or similar devices into the design to prevent negative current from flowing in.
The load current generated by powering the device on and off may cause it to function erratically or to break down, which could in turn cause injury.

Ensure that the power supply to any device which incorporates protective functions is stable.
If the power supply is unstable, the device may operate erratically, preventing the protective functions from working correctly. If protective functions fail, the device may break down causing injury to the user.

3.   General Safety Precautions and Usage Considerations

This section is designed to help you gain a better understanding of semiconductor devices, so as to ensure the safety, quality and reliability of the devices which you incorporate into your designs.

3.1   From Incoming to Shipping

3.1.1   Electrostatic discharge (ESD)

When handling individual devices (which are not yet mounted on a printed circuit board), be sure that the environment is protected against electrostatic electricity. Operators should wear anti-static clothing, and containers and other objects which come into direct contact with devices should be made of anti-static materials and should be grounded to earth via an 0.5- to 1.0-MΩ protective resistor.

Please follow the precautions described below; this is particularly important for devices which are marked "Be careful of static.".

(1)   Work environment

• When humidity in the working environment decreases, the human body and other insulators can easily become charged with static electricity due to friction. Maintain the recommended humidity of 40% to 60% in the work environment, while also taking into account the fact that moisture-proof-packed products may absorb moisture after unpacking.

• Be sure that all equipment, jigs and tools in the working area are grounded to earth.

• Place a conductive mat over the floor of the work area, or take other appropriate measures, so that the floor surface is protected against static electricity and is grounded to earth. The surface resistivity should be 10^4 to 10^8 Ω/sq and the resistance between surface and ground, 7.5×10^5 to 10^8 Ω

• Cover the workbench surface also with a conductive mat (with a surface resistivity of 10^4 to 10^8 Ω/sq, for a resistance between surface and ground of 7.5×10^5 to 10^8 Ω) . The purpose of this is to disperse static electricity on the surface (through resistive components) and ground it to earth. Workbench surfaces must not be constructed of low-resistance metallic materials that allow rapid static discharge when a charged device touches them directly.

• Pay attention to the following points when using automatic equipment in your workplace:

  (a)   When picking up ICs with a vacuum unit, use a conductive rubber fitting on the end of the pick-up wand to protect against electrostatic charge.

  (b)   Minimize friction on IC package surfaces. If some rubbing is unavoidable due to the device's mechanical structure, minimize the friction plane or use material with a small friction coefficient and low electrical resistance. Also, consider the use of an ionizer.

  (c)   In sections which come into contact with device lead terminals, use a material which dissipates static electricity.

  (d)   Ensure that no statically charged bodies (such as work clothes or the human body) touch the devices.

(e)   Make sure that sections of the tape carrier which come into contact with installation devices or other electrical machinery are made of a low-resistance material.

(f)   Make sure that jigs and tools used in the assembly process do not touch devices.

(g)   In processes in which packages may retain an electrostatic charge, use an ionizer to neutralize the ions.

- Make sure that CRT displays in the working area are protected against static charge, for example by a VDT filter. As much as possible, avoid turning displays on and off. Doing so can cause electrostatic induction in devices.

- Keep track of charged potential in the working area by taking periodic measurements.

- Ensure that work chairs are protected by an anti-static textile cover and are grounded to the floor surface by a grounding chain. (Suggested resistance between the seat surface and grounding chain is 7.5×10^5 to $10^{12}\Omega$.)

- Install anti-static mats on storage shelf surfaces. (Suggested surface resistivity is 10^4 to 10^8 Ω/sq; suggested resistance between surface and ground is 7.5×10^5 to 10^8 Ω.)

- For transport and temporary storage of devices, use containers (boxes, jigs or bags) that are made of anti-static materials or materials which dissipate electrostatic charge.

- Make sure that cart surfaces which come into contact with device packaging are made of materials which will conduct static electricity, and verify that they are grounded to the floor surface via a grounding chain.

- In any location where the level of static electricity is to be closely controlled, the ground resistance level should be Class 3 or above. Use different ground wires for all items of equipment which may come into physical contact with devices.

(2)   Operating environment

- Operators must wear anti-static clothing and conductive shoes (or a leg or heel strap).

- Operators must wear a wrist strap grounded to earth via a resistor of about 1 MΩ.

- Soldering irons must be grounded from iron tip to earth, and must be used only at low voltages (6 V to 24 V).

- If the tweezers you use are likely to touch the device terminals, use anti-static tweezers and in particular avoid metallic tweezers. If a charged device touches a low-resistance tool, rapid discharge can occur. When using vacuum tweezers, attach a conductive chucking pat to the tip, and connect it to a dedicated ground used especially for anti-static purposes (suggested resistance value: 10^4 to 10^8 Ω).

- Do not place devices or their containers near sources of strong electrical fields (such as above a CRT).
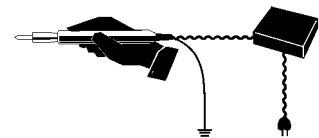
- When storing printed circuit boards which have devices mounted on them, use a board container or bag that is protected against static charge. To avoid the occurrence of static charge or discharge due to friction, keep the boards separate from one other and do not stack them directly on top of one another.

- Ensure, if possible, that any articles (such as clipboards) which are brought to any location where the level of static electricity must be closely controlled are constructed of anti-static materials.

- In cases where the human body comes into direct contact with a device, be sure to wear anti-static finger covers or gloves (suggested resistance value: 10^8 Ω or less).

- Equipment safety covers installed near devices should have resistance ratings of 10^9 Ω or less.

- If a wrist strap cannot be used for some reason, and there is a possibility of imparting friction to devices, use an ionizer.

- The transport film used in TCP products is manufactured from materials in which static charges tend to build up. When using these products, install an ionizer to prevent the film from being charged with static electricity. Also, ensure that no static electricity will be applied to the product's copper foils by taking measures to prevent static occuring in the peripheral equipment.

3.1.2   Vibration, impact and stress

Handle devices and packaging materials with care. To avoid damage to devices, do not toss or drop packages. Ensure that devices are not subjected to mechanical vibration or shock during transportation. Ceramic package devices and devices in canister-type packages which have empty space inside them are subject to damage from vibration and shock because the bonding wires are secured only at their ends.
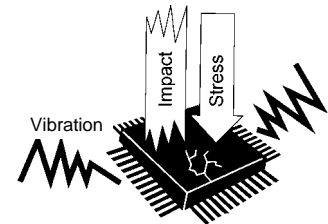
Plastic molded devices, on the other hand, have a relatively high level of resistance to vibration and mechanical shock because their bonding wires are enveloped and fixed in resin. However, when any device or package type is installed in target equipment, it is to some extent susceptible to wiring disconnections and other damage from vibration, shock and stressed solder junctions. Therefore when devices are incorporated into the design of equipment which will be subject to vibration, the structural design of the equipment must be thought out carefully.

If a device is subjected to especially strong vibration, mechanical shock or stress, the package or the chip itself may crack. In products such as CCDs which incorporate window glass, this could cause surface flaws in the glass or cause the connection between the glass and the ceramic to separate.

Furthermore, it is known that stress applied to a semiconductor device through the package changes the resistance characteristics of the chip because of piezoelectric effects. In analog circuit design attention must be paid to the problem of package stress as well as to the dangers of vibration and shock as described above.

3.2   Storage

3.2.1   General storage

- Avoid storage locations where devices will be exposed to moisture or direct sunlight.

- Follow the instructions printed on the device cartons regarding transportation and storage.

- The storage area temperature should be kept within a temperature range of 5°C to 35°C, and relative humidity should be maintained at between 45% and 75%.

- Do not store devices in the presence of harmful (especially corrosive) gases, or in dusty conditions.

- Use storage areas where there is minimal temperature fluctuation. Rapid temperature changes can cause moisture to form on stored devices, resulting in lead oxidation or corrosion. As a result, the solderability of the leads will be degraded.

- When repacking devices, use anti-static containers.

- Do not allow external forces or loads to be applied to devices while they are in storage.

- If devices have been stored for more than two years, their electrical characteristics should be tested and their leads should be tested for ease of soldering before they are used.
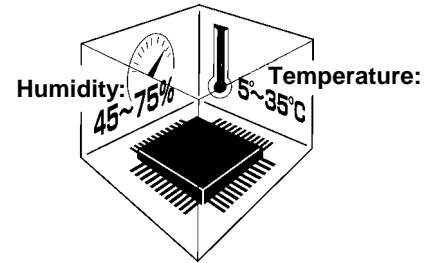
3.2.2   Moisture-proof packing

Moisture-proof packing should be handled with care. The handling procedure specified for each packing type should be followed scrupulously. If the proper procedures are not followed, the quality and reliability of devices may be degraded. This section describes general precautions for handling moisture-proof packing. Since the details may differ from device to device, refer also to the relevant individual datasheets or databook.

(1)   General precautions
    Follow the instructions printed on the device cartons regarding transportation and storage.

- Do not drop or toss device packing. The laminated aluminum material in it can be rendered ineffective by rough handling.

- The storage area temperature should be kept within a temperature range of 5°C to 30°C, and relative humidity should be maintained at 90% (max). Use devices within 12 months of the date marked on the package seal.

- If the 12-month storage period has expired, or if the 30% humidity indicator shown in Figure 1 is pink when the packing is opened, it may be advisable, depending on the device and packing type, to back the devices at high temperature to remove any moisture. Please refer to the table below. After the pack has been opened, use the devices in a 5°C to 30°C. 60% RH environment and within the effective usage period listed on the moisture-proof package. If the effective usage period has expired, or if the packing has been stored in a high-humidity environment, bake the devices at high temperature.

Packing	Moisture removal
Tray	If the packing bears the "Heatproof" marking or indicates the maximum temperature which it can withstand, bake at 125°C for 20 hours. (Some devices require a different procedure.)
Tube	Transfer devices to trays bearing the "Heatproof" marking or indicating the temperature which they can withstand, or to aluminum tubes before baking at 125°C for 20 hours.
Tape	Deviced packed on tape cannot be baked and must be used within the effective usage period after unpacking, as specified on the packing.

- When baking devices, protect the devices from static electricity.

- Moisture indicators can detect the approximate humidity level at a standard temperature of 25°C. 6-point indicators and 3-point indicators are currently in use, but eventually all indicators will be 3-point indicators.

(a) 6-point indicator   (b) 3-point indicator
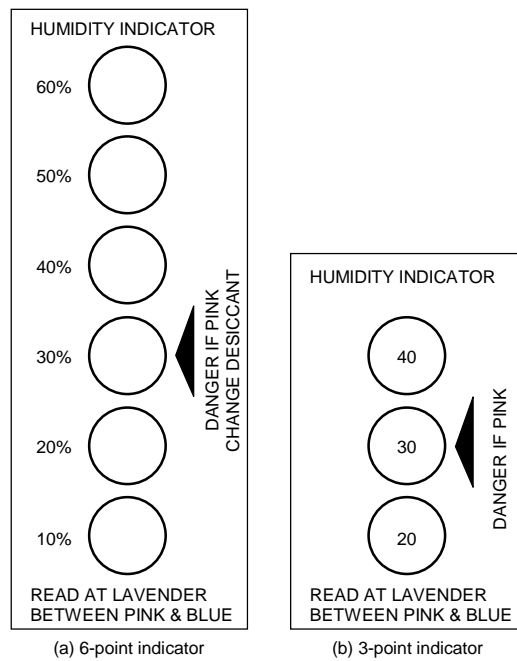
Figure 1   Humidity indicator

3.3   Design

Care must be exercised in the design of electronic equipment to achieve the desired reliability. It is important not only to adhere to specifications concerning absolute maximum ratings and recommended operating conditions, it is also important to consider the overall environment in which equipment will be used, including factors such as the ambient temperature, transient noise and voltage and current surges, as well as mounting conditions which affect device reliability. This section describes some general precautions which you should observe when designing circuits and when mounting devices on printed circuit boards.

For more detailed information about each product family, refer to the relevant individual technical datasheets available from Toshiba.

3.3.1   Absolute maximum ratings

⚠CAUTION   Do not use devices under conditions in which their absolute maximum ratings (e.g. current, voltage, power dissipation or temperature) will be exceeded. A device may break down or its performance may be degraded, causing it to catch fire or explode resulting in injury to the user.

The absolute maximum ratings are rated values which must not be exceeded during operation, even for an instant. Although absolute maximum ratings differ from product to product, they essentially concern the voltage and current at each pin, the allowable power dissipation, and the junction and storage temperatures.
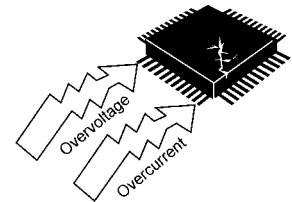
If the voltage or current on any pin exceeds the absolute maximum rating, the device's internal circuitry can become degraded. In the worst case, heat generated in internal circuitry can fuse wiring or cause the semiconductor chip to break down.

If storage or operating temperatures exceed rated values, the package seal can deteriorate or the wires can become disconnected due to the differences between the thermal expansion coefficients of the materials from which the device is constructed.

3.3.2   Recommended operating conditions

The recommended operating conditions for each device are those necessary to guarantee that the device will operate as specified in the datasheet.
If greater reliability is required, derate the device's absolute maximum ratings for voltage, current, power and temperature before using it.

3.3.3   Derating

When incorporating a device into your design, reduce its rated absolute maximum voltage, current, power dissipation and operating temperature in order to ensure high reliability.
Since derating differs from application to application, refer to the technical datasheets available for the various devices used in your design.

3.3.4   Unused pins

If unused pins are left open, some devices can exhibit input instability problems, resulting in malfunctions such as abrupt increase in current flow. Similarly, if the unused output pins on a device are connected to the power supply pin, the ground pin or to other output pins, the IC may malfunction or break down.

Since the details regarding the handling of unused pins differ from device to device and from pin to pin, please follow the instructions given in the relevant individual datasheets or databook.

CMOS logic IC inputs, for example, have extremely high impedance. If an input pin is left open, it can easily pick up extraneous noise and become unstable. In this case, if the input voltage level reaches an intermediate level, it is possible that both the P-channel and N-channel transistors will be turned on, allowing unwanted supply current to flow. Therefore, ensure that the unused input pins of a device are connected to the power supply (Vcc) pin or ground (GND) pin of the same device. For details of what to do with the pins of heat sinks, refer to the relevant technical datasheet and databook.

3.3.5   Latch-up

Latch-up is an abnormal condition inherent in CMOS devices, in which Vcc gets shorted to ground. This happens when a parasitic PN-PN junction (thyristor structure) internal to the CMOS chip is turned on, causing a large current of the order of several hundred mA or more to flow between Vcc and GND, eventually causing the device to break down.

Latch-up occurs when the input or output voltage exceeds the rated value, causing a large current to flow in the internal chip, or when the voltage on the Vcc (Vdd) pin exceeds its rated value, forcing the internal chip into a breakdown condition. Once the chip falls into the latch-up state, even though the excess voltage may have been applied only for an instant, the large current continues to flow between Vcc (Vdd) and GND (Vss). This causes the device to heat up and, in extreme cases, to emit gas fumes as well. To avoid this problem, observe the following precautions:

(1)   Do not allow voltage levels on the input and output pins either to rise above Vcc (Vdd) or to fall below GND (Vss). Also, follow any prescribed power-on sequence, so that power is applied gradually or in steps rather than abruptly.

(2)   Do not allow any abnormal noise signals to be applied to the device.

(3)   Set the voltage levels of unused input pins to Vcc (Vdd) or GND (Vss).

(4)   Do not connect output pins to one another.

3.3.6   Input/Output protection

Wired-AND configurations, in which outputs are connected together, cannot be used, since this short-circuits the outputs. Outputs should, of course, never be connected to Vcc (Vdd) or GND (Vss).

Furthermore, ICs with tri-state outputs can undergo performance degradation if a shorted output current is allowed to flow for an extended period of time. Therefore, when designing circuits, make sure that tri-state outputs will not be enabled simultaneously.

3.3.7   Load capacitance

Some devices display increased delay times if the load capacitance is large. Also, large charging and discharging currents will flow in the device, causing noise. Furthermore, since outputs are shorted for a relatively long time, wiring can become fused.

Consult the technical information for the device being used to determine the recommended load capacitance.

3.3.8   Thermal design

The failure rate of semiconductor devices is greatly increased as operating temperatures increase. As shown in Figure 2, the internal thermal stress on a device is the sum of the ambient temperature and the temperature rise due to power dissipation in the device. Therefore, to achieve optimum reliability, observe the following precautions concerning thermal design:

(1) Keep the ambient temperature (Ta) as low as possible.

(2) If the device's dynamic power dissipation is relatively large, select the most appropriate circuit board material, and consider the use of heat sinks or of forced air cooling. Such measures will help lower the thermal resistance of the package.

(3) Derate the device's absolute maximum ratings to minimize thermal stress from power dissipation.

$\theta ja = \theta jc + \theta ca$

$\theta ja = (Tj-Ta) / P$

$\theta jc = (Tj-Tc) / P$

$\theta ca = (Tc-Ta) / P$

in which θja = thermal resistance between junction and surrounding air (°C/W)

θjc = thermal resistance between junction and package surface, or internal thermal resistance (°C/W)

θca = thermal resistance between package surface and surrounding air, or external thermal resistance (°C/W)

Tj = junction temperature or chip temperature (°C)

Tc = package surface temperature or case temperature (°C)

Ta = ambient temperature (°C)

P = power dissipation (W)
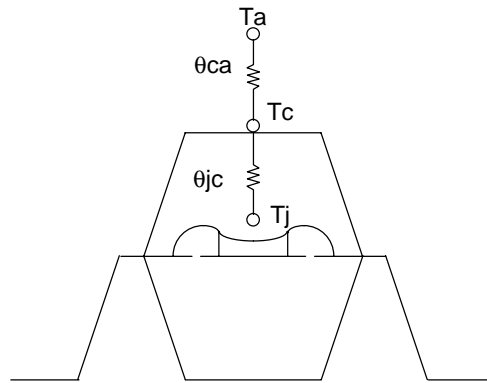
Figure 2   Thermal resistance of package

3.3.9   Interfacing

When connecting inputs and outputs between devices, make sure input voltage (V$_{IL}$/V$_{IH}$) and output voltage (V$_{OL}$/V$_{OH}$) levels are matched. Otherwise, the devices may malfunction. When connecting devices operating at different supply voltages, such as in a dual-power-supply system, be aware that erroneous power-on and power-off sequences can result in device breakdown. For details of how to interface particular devices, consult the relevant technical datasheets and databooks. If you have any questions or doubts about interfacing, contact your nearest Toshiba office or distributor.

3.3.10   Decoupling

Spike currents generated during switching can cause Vcc (Vdd) and GND (Vss) voltage levels to fluctuate, causing ringing in the output waveform or a delay in response speed. (The power supply and GND wiring impedance is normally 50 Ω to 100 Ω.) For this reason, the impedance of power supply lines with respect to high frequencies must be kept low. This can be accomplished by using thick and short wiring for the Vcc (Vdd) and GND (Vss) lines and by installing decoupling capacitors (of approximately 0.01 μF to 1 μF capacitance) as high-frequency filters between Vcc (Vdd) and GND (Vss) at strategic locations on the printed circuit board.

For low-frequency filtering, it is a good idea to install a 10- to 100-μF capacitor on the printed circuit board (one capacitor will suffice). If the capacitance is excessively large, however, (e.g. several thousand μF) latch-up can be a problem. Be sure to choose an appropriate capacitance value.

An important point about wiring is that, in the case of high-speed logic ICs, noise is caused mainly by reflection and crosstalk, or by the power supply impedance. Reflections cause increased signal delay, ringing, overshoot and undershoot, thereby reducing the device's safety margins with respect to noise. To prevent reflections, reduce the wiring length by increasing the device mounting density so as to lower the inductance (L) and capacitance (C) in the wiring. Extreme care must be taken, however, when taking this corrective measure, since it tends to cause crosstalk between the wires. In practice, there must be a trade-off between these two factors.

3.3.11   External noise

Printed circuit boards with long I/O or signal pattern lines are vulnerable to induced noise or surges from outside sources. Consequently, malfunctions or breakdowns can result from overcurrent or overvoltage, depending on the types of device used. To protect against noise, lower the impedance of the pattern line or insert a noise-canceling circuit. Protective measures must also be taken against surges.
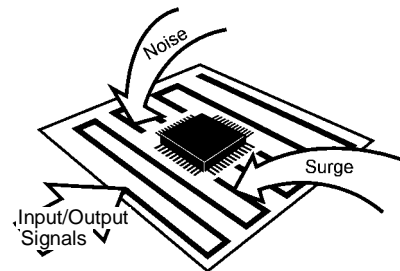
For details of the appropriate protective measures for a particular device, consult the relevant databook.

3.3.12   Electromagnetic interference

Widespread use of electrical and electronic equipment in recent years has brought with it radio and TV reception problems due to electromagnetic interference. To use the radio spectrum effectively and to maintain radio communications quality, each country has formulated regulations limiting the amount of electromagnetic interference which can be generated by individual products.

Electromagnetic interference includes conduction noise propagated through power supply and telephone lines, and noise from direct electromagnetic waves radiated by equipment. Different measurement methods and corrective measures are used to assess and counteract each specific type of noise.

Difficulties in controlling electromagnetic interference derive from the fact that there is no method available which allows designers to calculate, at the design stage, the strength of the electromagnetic waves which will emanate from each component in a piece of equipment. For this reason, it is only after the prototype equipment has been completed that the designer can take measurements using a dedicated instrument to determine the strength of electromagnetic interference waves. Yet it is possible during system design to incorporate some measures for the prevention of electromagnetic interference, which can facilitate taking corrective measures once the design has been completed. These include installing shields and noise filters, and increasing

the thickness of the power supply wiring patterns on the printed circuit board. One effective method, for example, is to devise several shielding options during design, and then select the most suitable shielding method based on the results of measurements taken after the prototype has been completed.

3.3.13   Peripheral circuits

In most cases semiconductor devices are used with peripheral circuits and components. The input and output signal voltages and currents in these circuits must be chosen to match the semiconductor device's specifications. The following factors must be taken into account.

(1)   Inappropriate voltages or currents applied to a device's input pins may cause it to operate erratically. Some devices contain pull-up or pull-down resistors. When designing your system, remember to take the effect of this on the voltage and current levels into account.

(2)   The output pins on a device have a predetermined external circuit drive capability. If this drive capability is greater than that required, either incorporate a compensating circuit into your design or carefully select suitable components for use in external circuits.

3.3.14   Safety standards

Each country has safety standards which must be observed. These safety standards include requirements for quality assurance systems and design of device insulation. Such requirements must be fully taken into account to ensure that your design conforms to the applicable safety standards.

3.3.15   Other precautions

(1)   When designing a system, be sure to incorporate fail-safe and other appropriate measures according to the intended purpose of your system. Also, be sure to debug your system under actual board-mounted conditions.

(2)   If a plastic-package device is placed in a strong electric field, surface leakage may occur due to the charge-up phenomenon, resulting in device malfunction. In such cases take appropriate measures to prevent this problem, for example by protecting the package surface with a conductive shield.

(3)   With some microcomputers and MOS memory devices, caution is required when powering on or resetting the device. To ensure that your design does not violate device specifications, consult the relevant databook for each constituent device.

(4)   Ensure that no conductive material or object (such as a metal pin) can drop onto and short the leads of a device mounted on a printed circuit board.

3.4   Inspection, Testing and Evaluation

3.4.1   Grounding

⚠CAUTION   Ground all measuring instruments, jigs, tools and soldering irons to earth. Electrical leakage may cause a device to break down or may result in electric shock.

3.4.2 Inspection Sequence

⚠CAUTION

① Do not insert devices in the wrong orientation. Make sure that the positive and negative electrodes of the power supply are correctly connected. Otherwise, the rated maximum current or maximum power dissipation may be exceeded and the device may break down or undergo performance degradation, causing it to catch fire or explode, resulting in injury to the user.

② When conducting any kind of evaluation, inspection or testing using AC power with a peak voltage of 42.4 V or DC power exceeding 60 V, be sure to connect the electrodes or probes of the testing equipment to the device under test before powering it on. Connecting the electrodes or probes of testing equipment to a device while it is powered on may result in electric shock, causing injury.

(1) Apply voltage to the test jig only after inserting the device securely into it. When applying or removing power, observe the relevant precautions, if any.

(2) Make sure that the voltage applied to the device is off before removing the device from the test jig. Otherwise, the device may undergo performance degradation or be destroyed.

(3) Make sure that no surge voltages from the measuring equipment are applied to the device.

(4) The chips housed in tape carrier packages (TCPs) are bare chips and are therefore exposed. During inspection take care not to crack the chip or cause any flaws in it.
Electrical contact may also cause a chip to become faulty. Therefore make sure that nothing comes into electrical contact with the chip.

3.5 Mounting

There are essentially two main types of semiconductor device package: lead insertion and surface mount. During mounting on printed circuit boards, devices can become contaminated by flux or damaged by thermal stress from the soldering process. With surface-mount devices in particular, the most significant problem is thermal stress from solder reflow, when the entire package is subjected to heat. This section describes a recommended temperature profile for each mounting method, as well as general precautions which you should take when mounting devices on printed circuit boards. Note, however, that even for devices with the same package type, the appropriate mounting method varies according to the size of the chip and the size and shape of the lead frame. Therefore, please consult the relevant technical datasheet and databook.

3.5.1 Lead forming

⚠CAUTION

① Always wear protective glasses when cutting the leads of a device with clippers or a similar tool. If you do not, small bits of metal flying off the cut ends may damage your eyes.

② Do not touch the tips of device leads. Because some types of device have leads with pointed tips, you may prick your finger.

Semiconductor devices must undergo a process in which the leads are cut and formed before the devices can be mounted on a printed circuit board. If undue stress is applied to the interior of a device during this process, mechanical breakdown or performance degradation can result. This is attributable primarily to differences between the stress on the device's external leads and the stress on the internal leads. If the relative difference is great enough, the device's internal leads, adhesive properties or sealant can be damaged. Observe these precautions during the lead-forming process (this does not apply to surface-mount devices):

(1)   Lead insertion hole intervals on the printed circuit board should match the lead pitch of the device precisely.

(2)   If lead insertion hole intervals on the printed circuit board do not precisely match the lead pitch of the device, do not attempt to forcibly insert devices by pressing on them or by pulling on their leads.

(3)   For the minimum clearance specification between a device and a printed circuit board, refer to the relevant device's datasheet and databook. If necessary, achieve the required clearance by forming the device's leads appropriately. Do not use the spacers which are used to raise devices above the surface of the printed circuit board during soldering to achieve clearance. These spacers normally continue to expand due to heat, even after the solder has begun to solidify; this applies severe stress to the device.

(4)   Observe the following precautions when forming the leads of a device prior to mounting.

• Use a tool or jig to secure the lead at its base (where the lead meets the device package) while bending so as to avoid mechanical stress to the device. Also avoid bending or stretching device leads repeatedly.

• Be careful not to damage the lead during lead forming.

• Follow any other precautions described in the individual datasheets and databooks for each device and package type.
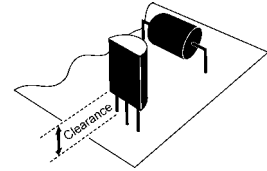
3.5.2   Socket mounting

(1)   When socket mounting devices on a printed circuit board, use sockets which match the inserted device's package.

(2)   Use sockets whose contacts have the appropriate contact pressure. If the contact pressure is insufficient, the socket may not make a perfect contact when the device is repeatedly inserted and removed; if the pressure is excessively high, the device leads may be bent or damaged when they are inserted into or removed from the socket.

(3)   When soldering sockets to the printed circuit board, use sockets whose construction prevents flux from penetrating into the contacts or which allows flux to be completely cleaned off.

(4)   Make sure the coating agent applied to the printed circuit board for moisture-proofing purposes does not stick to the socket contacts.

(5)   If the device leads are severely bent by a socket as it is inserted or removed and you wish to repair the leads so as to continue using the device, make sure that this lead correction is only performed once. Do not use devices whose leads have been corrected more than once.

(6)   If the printed circuit board with the devices mounted on it will be subjected to vibration from external sources, use sockets which have a strong contact pressure so as to prevent the sockets and devices from vibrating relative to one another.

3.5.3   Soldering temperature profile

The soldering temperature and heating time vary from device to device. Therefore, when specifying the mounting conditions, refer to the individual datasheets and databooks for the devices used.

(1)   Using a soldering iron

Complete soldering within ten seconds for lead temperatures of up to 260°C, or within three seconds for lead temperatures of up to 350°C.

(2)   Using medium infrared ray reflow

• Heating top and bottom with long or medium infrared rays is recommended (see Figure 3).
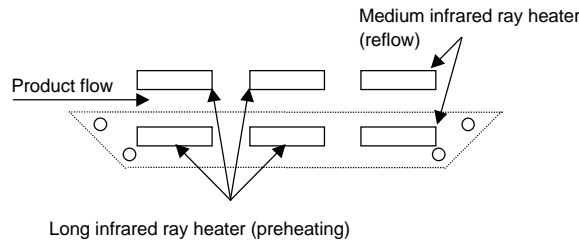
Figure 3   Heating top and bottom with long or medium infrared rays

• Complete the infrared ray reflow process within 30 seconds at a package surface temperature of between 210°C and 240°C.

• Refer to Figure 4 for an example of a good temperature profile for infrared or hot air reflow.
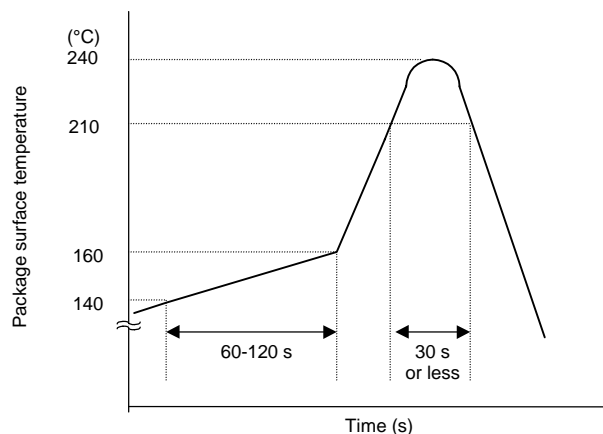
Figure 4   Sample temperature profile for infrared or hot air reflow

(3)   Using hot air reflow

• Complete hot air reflow within 30 seconds at a package surface temperature of between 210°C and 240°C.

• For an example of a recommended temperature profile, refer to Figure 4 above.

(4)   Using solder flow

• Apply preheating for 60 to 120 seconds at a temperature of 150°C.

• For lead insertion-type packages, complete solder flow within 10 seconds with the temperature at the stopper (or, if there is no stopper, at a location more than 1.5 mm from the body) which does not exceed 260°C.

- For surface-mount packages, complete soldering within 5 seconds at a temperature of 250°C or less in order to prevent thermal stress in the device.

- Figure 5 shows an example of a recommended temperature profile for surface-mount packages using solder flow.
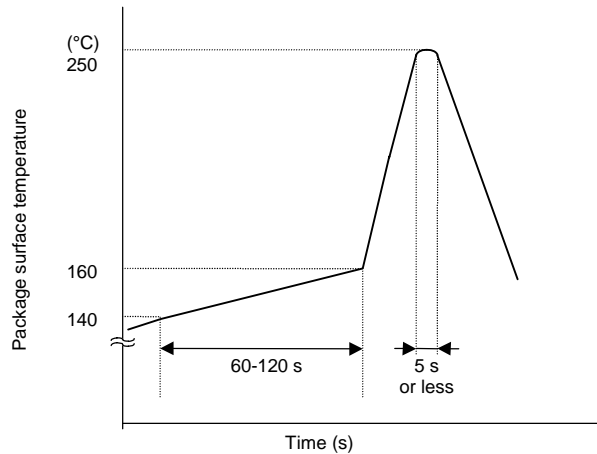
Figure 5   Sample temperature profile for solder flow

3.5.4   Flux cleaning and ultrasonic cleaning

(1) When cleaning circuit boards to remove flux, make sure that no residual reactive ions such as Na or Cl remain. Note that organic solvents react with water to generate hydrogen chloride and other corrosive gases which can degrade device performance.

(2) Washing devices with water will not cause any problems. However, make sure that no reactive ions such as sodium and chlorine are left as a residue. Also, be sure to dry devices sufficiently after washing.

(3) Do not rub device markings with a brush or with your hand during cleaning or while the devices are still wet from the cleaning agent. Doing so can rub off the markings.

(4) The dip cleaning, shower cleaning and steam cleaning processes all involve the chemical action of a solvent. Use only recommended solvents for these cleaning methods. When immersing devices in a solvent or steam bath, make sure that the temperature of the liquid is 50°C or below, and that the circuit board is removed from the bath within one minute.

(5) Ultrasonic cleaning should not be used with hermetically-sealed ceramic packages such as a leadless chip carrier (LCC), pin grid array (PGA) or charge-coupled device (CCD), because the bonding wires can become disconnected due to resonance during the cleaning process. Even if a device package allows ultrasonic cleaning, limit the duration of ultrasonic cleaning to as short a time as possible, since long hours of ultrasonic cleaning degrade the adhesion between the mold resin and the frame material. The following ultrasonic cleaning conditions are recommended:

Frequency: 27 kHz ~ 29 kHz

Ultrasonic output power: 300 W or less (0.25 W/cm^2 or less)

Cleaning time: 30 seconds or less

Suspend the circuit board in the solvent bath during ultrasonic cleaning in such a way that the ultrasonic vibrator does not come into direct contact with the circuit board or the device.

3.5.5   No cleaning

If analog devices or high-speed devices are used without being cleaned, flux residues may cause minute amounts of leakage between pins. Similarly, dew condensation, which occurs in environments containing residual chlorine when power to the device is on, may cause between-lead leakage or migration. Therefore, Toshiba recommends that these devices be cleaned. However, if the flux used contains only a small amount of halogen (0.05W% or less), the devices may be used without cleaning without any problems.

3.5.6   Mounting tape carrier packages (TCPs)

(1)   When tape carrier packages (TCPs) are mounted, measures must be taken to prevent electrostatic breakdown of the devices.

(2)   If devices are being picked up from tape, or outer lead bonding (OLB) mounting is being carried out, consult the manufacturer of the insertion machine which is being used, in order to establish the optimum mounting conditions in advance and to avoid any possible hazards.

(3)   The base film, which is made of polyimide, is hard and thin. Be careful not to cut or scratch your hands or any objects while handling the tape.

(4)   When punching tape, try not to scatter broken pieces of tape too much.

(5)   Treat the extra film, reels and spacers left after punching as industrial waste, taking care not to destroy or pollute the environment.

(6)   Chips housed in tape carrier packages (TCPs) are bare chips and therefore have their reverse side exposed. To ensure that the chip will not be cracked during mounting, ensure that no mechanical shock is applied to the reverse side of the chip. Electrical contact may also cause a chip to fail. Therefore, when mounting devices, make sure that nothing comes into electrical contact with the reverse side of the chip.
If your design requires connecting the reverse side of the chip to the circuit board, please consult Toshiba or a Toshiba distributor beforehand.

3.5.7   Mounting chips

Devices delivered in chip form tend to degrade or break under external forces much more easily than plastic-packaged devices. Therefore, caution is required when handling this type of device.

(1)   Mount devices in a properly prepared environment so that chip surfaces will not be exposed to polluted ambient air or other polluted substances.

(2)   When handling chips, be careful not to expose them to static electricity.
In particular, measures must be taken to prevent static damage during the mounting of chips. With this in mind, Toshiba recommend mounting all peripheral parts first and then mounting chips last (after all other components have been mounted).

(3)   Make sure that PCBs (or any other kind of circuit board) on which chips are being mounted do not have any chemical residues on them (such as the chemicals which were used for etching the PCBs).

(4)   When mounting chips on a board, use the method of assembly that is most suitable for maintaining the appropriate electrical, thermal and mechanical properties of the semiconductor devices used.

* For details of devices in chip form, refer to the relevant device's individual datasheets.

3.5.8   Circuit board coating

When devices are to be used in equipment requiring a high degree of reliability or in extreme environments (where moisture, corrosive gas or dust is present), circuit boards may be coated for protection. However, before doing so, you must carefully consider the possible stress and contamination effects that may result and then choose the coating resin which results in the minimum level of stress to the device.

3.5.9   Heat sinks

(1)   When attaching a heat sink to a device, be careful not to apply excessive force to the device in the process.

(2)   When attaching a device to a heat sink by fixing it at two or more locations, evenly tighten all the screws in stages (i.e. do not fully tighten one screw while the rest are still only loosely tightened). Finally, fully tighten all the screws up to the specified torque.

(3)   Drill holes for screws in the heat sink exactly as specified. Smooth the surface by removing burrs and protrusions or indentations which might interfere with the installation of any part of the device.

(4)   A coating of silicone compound can be applied between the heat sink and the device to improve heat conductivity. Be sure to apply the coating thinly and evenly; do not use too much. Also, be sure to use a non-volatile compound, as volatile compounds can crack after a time, causing the heat radiation properties of the heat sink to deteriorate.

(5)   If the device is housed in a plastic package, use caution when selecting the type of silicone compound to be applied between the heat sink and the device. With some types, the base oil separates and penetrates the plastic package, significantly reducing the useful life of the device.
Two recommended silicone compounds in which base oil separation is not a problem are YG6260 from Toshiba Silicone.

(6)   Heat-sink-equipped devices can become very hot during operation. Do not touch them, or you may sustain a burn.
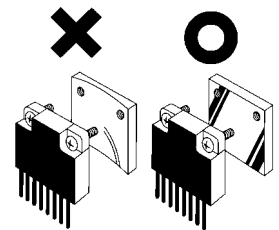
3.5.10   Tightening torque

(1)   Make sure the screws are tightened with fastening torques not exceeding the torque values stipulated in individual datasheets and databooks for the devices used.

(2)   Do not allow a power screwdriver (electrical or air-driven) to touch devices.

3.5.11   Repeated device mounting and usage

Do not remount or re-use devices which fall into the categories listed below; these devices may cause significant problems relating to performance and reliability.

(1)   Devices which have been removed from the board after soldering

(2)   Devices which have been inserted in the wrong orientation or which have had reverse current applied

(3)   Devices which have undergone lead forming more than once

3.6   Protecting Devices in the Field

3.6.1   Temperature

Semiconductor devices are generally more sensitive to temperature than are other electronic components. The various electrical characteristics of a semiconductor device are dependent on the ambient temperature at which the device is used. It is therefore necessary to understand the temperature characteristics of a device and to incorporate device derating into circuit design. Note also that if a device is used above its maximum temperature rating, device deterioration is more rapid and it will reach the end of its usable life sooner than expected.

3.6.2   Humidity

Resin-molded devices are sometimes improperly sealed. When these devices are used for an extended period of time in a high-humidity environment, moisture can penetrate into the device and cause chip degradation or malfunction. Furthermore, when devices are mounted on a regular printed circuit board, the impedance between wiring components can decrease under high-humidity conditions. In systems which require a high signal-source impedance, circuit board leakage or leakage between device lead pins can cause malfunctions. The application of a moisture-proof treatment to the device surface should be considered in this case. On the other hand, operation under low-humidity conditions can damage a device due to the occurrence of electrostatic discharge. Unless damp-proofing measures have been specifically taken, use devices only in environments with appropriate ambient moisture levels (i.e. within a relative humidity range of 40% to 60%).

3.6.3   Corrosive gases

Corrosive gases can cause chemical reactions in devices, degrading device characteristics. For example, sulphur-bearing corrosive gases emanating from rubber placed near a device (accompanied by condensation under high-humidity conditions) can corrode a device's leads. The resulting chemical reaction between leads forms foreign particles which can cause electrical leakage.

3.6.4   Radioactive and cosmic rays

Most industrial and consumer semiconductor devices are not designed with protection against radioactive and cosmic rays. Devices used in aerospace equipment or in radioactive environments must therefore be shielded.

3.6.5   Strong electrical and magnetic fields

Devices exposed to strong magnetic fields can undergo a polarization phenomenon in their plastic material, or within the chip, which gives rise to abnormal symptoms such as impedance changes or increased leakage current. Failures have been reported in LSIs mounted near malfunctioning deflection yokes in TV sets. In such cases the device's installation location must be changed or the device must be shielded against the electrical or magnetic field. Shielding against magnetism is especially necessary for devices used in an alternating magnetic field because of the electromotive forces generated in this type of environment.

3.6.6   Interference from light (ultraviolet rays, sunlight, fluorescent lamps and incandescent lamps)

Light striking a semiconductor device generates electromotive force due to photoelectric effects. In some cases the device can malfunction. This is especially true for devices in which the internal chip is exposed. When designing circuits, make sure that devices are protected against incident light from external sources. This problem is not limited to optical semiconductors and EPROMs. All types of device can be affected by light.

3.6.7   Dust and oil

Just like corrosive gases, dust and oil can cause chemical reactions in devices, which will adversely affect a device's electrical characteristics. To avoid this problem, do not use devices in dusty or oily environments. This is especially important for optical devices because dust and oil can affect a device's optical characteristics as well as its physical integrity and the electrical performance factors mentioned above.

3.6.8   Fire

Semiconductor devices are combustible; they can emit smoke and catch fire if heated sufficiently. When this happens, some devices may generate poisonous gases. Devices should therefore never be used in close proximity to an open flame or a heat-generating body, or near flammable or combustible materials.

3.7   Disposal of Devices and Packing Materials

When discarding unused devices and packing materials, follow all procedures specified by local regulations in order to protect the environment against contamination.

4.   Precautions and Usage Considerations

This section describes matters specific to each product group which need to be taken into consideration when using devices. If the same item is described in Sections 3 and 4, the description in Section 4 takes precedence.

4.1   Microcontrollers

4.1.1   Design

(1)   Using resonators which are not specifically recommended for use

Resonators recommended for use with Toshiba products in microcontroller oscillator applications are listed in Toshiba databooks along with information about oscillation conditions. If you use a resonator not included in this list, please consult Toshiba or the resonator manufacturer concerning the suitability of the device for your application.

(2)   Undefined functions

In some microcontrollers certain instruction code values do not constitute valid processor instructions. Also, it is possible that the values of bits in registers will become undefined. Take care in your applications not to use invalid instructions or to let register bit values become undefined.

4-2

TMPR4955A

Rev 2.2

Chapter 1.   Introduction

1.1   Overview

The TMPR4955A (to be called "TX4955A" hereinafter) is a standard microcontroller of 64-bit RISC Microprocessor TX49 family.

The TX4955A uses the TX49/H2 Processor Core as the CPU. The TX49/H2 Processor Core is a 64-bit RISC CPU core Toshiba developed based on the R4000 architecture of MIPS Technologies, Inc ("MIPS"). In addition to its TX49/H2 core with Floating Point Unit ("FPU"), the TX4955A also incorporates peripheral circuits such as SysAD Bus Interface.

R4000 is a trademark of MIPS Technologies, Inc.

1.2   Notation used in this manual

1.2.1   Numerical notation

- Hexadecimal numbers in this manual are expressed as follows:0x2A (example shown for decimal number 42)

- KB (kilobyte) 2^{10} = 1,024 bytes
  MB (megabyte) 2^{20} = 1,024 × 1,024 = 1,048,576 byte
  GB (gigabyte) 2^{30} = 1,024 × 1,024 × 1,024 = 1,073,741,824 bytes

1.2.2   Data notation

- Byte: Eight bits

- Half word: Two contiguous bytes (16 bits)

- Word: Four contiguous bytes (32 bits)

- Double word: Eight contiguous bytes (64 bits)

1.2.3   Signal notation

- Active-low signals are indicated by adding an asterisk(*) at the end of the signal name (Example: RESET*)

- When a signal is driven to the active voltage level, the signal is said to be "asserted." When the signal is driven to an inactive voltage level, it is said to be "deasserted."

1.2.4   Register notation

- The following nomenclature is used for access attributes.

  R: Read only. Cannot be written.

  W: Write only. The bit value is undefined if read.

  R/W: Read/Write

Chapter 2. Features

- **TX49/H2 Processor Core**
  TX49/H2 Processor Core is a 64-bit RISC CPU core Toshiba developed based on the architecture of MIPS for interactive consumer applications including Printer, Network and set-top terminals.

- **IEEE754 compatible single and double precision FPU**
  The floating-point operations fully conform to the requirements of ANSI/IEEE Standard 754-1985, IEEE Standard for Binary Floating-Point Arithmetic.
  As 32 general purpose register (32 FGRs), each of which is 64 bits wide and 2 control of FPU registers (2 FCRs), each of which is 32bit wide. All floating-point instructions, as defined in the MIPS ISA for the floating-point coprocessor, CP1, are processed by the other hardware unit that executes integer instruction.

- **Internal bus width is 64-bit, External bus width is 32-bit**
  Core and Cache are connect with 64-bit Internal bus. External bus is SysAD-bus I/F. This interface is compatible with the TX4300, R4000, and R5000 system interfaces.

- **Power management**
  Internal: 1.5 V   I/O: 3.3 V
  The TX49/H2 Processor support Power management mode (Halt, Doze)

- **Maximum operating frequency**
  The TX49/H2 Processor's maximum operating frequency is 200 MHz.
  The SysAD-bus I/F and TX49/H2 Processor's maximum operating frequency is set by External pin (DivMode (1:0)).

  Ex. Core's operating frequency is 167 MHz/200 MHz

Div Mode (1:0)	Master Clock	CPU Clock
0	41.8 MHz/50 MHz	167 MHz/200 MHz
1	66.7 MHz/80 MHz	167 MHz/200 MHz
2	83.5 MHz/100 MHz	167 MHz/200 MHz
3	55.7 MHz/66.7 MHz	167 MHz/200 MHz

- **Package**
  TX4955A: 160-pin QFP

- **Part Number**
  TMPR4955AF-167
  TMPR4955AF-200

2.1   Block Diagram
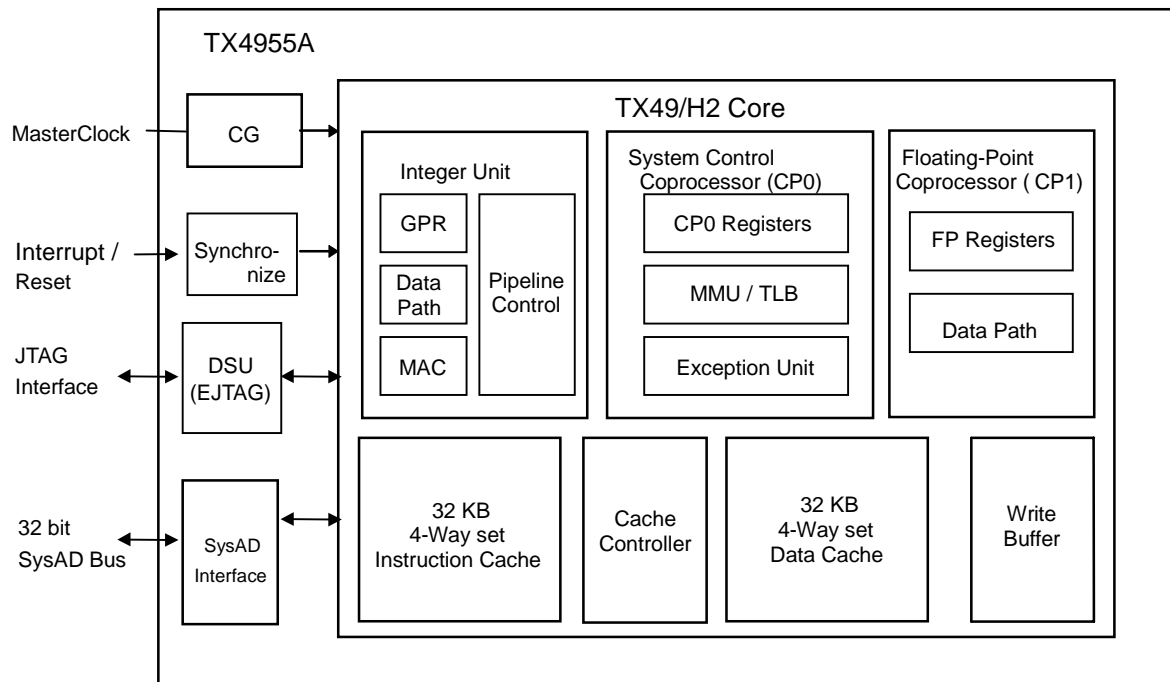
Figure 2.1.1 TX4955A Processor Signals

2.2 Pin Description

2.2.1 TX4955A pin out (160-pin QFP)

Table 2.2.1 TX4955A Pin Out

1	Vss	41	Vss	81	VccInt	121	SysAD28
2	BufSel1	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	BufSel1	75	VccIO	115	SysAD27	155	VssPLL
36	PCST3	76	Vss	116	VccIO	156	PLLCAP
37	PCST2	77	TPC3	117	MODE43*	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.

Table 2.2.2 System Interface

PIN NAME	I / O	FUNCTION
SysAD(31:0)	I / O	**System addres/data bus** A 32-bit address and data bus for communication between the processor and an external agent.
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(3:0)	I / O	**System command/data check bus** A 4-bit bus containing parity check bits for the SysAD bus during data cycle.
SysCmdP	I / O	**Reserved for system command/data identifier bus parity** For the TX4955A this signal is unused on input and zero on output.
ValidIn*	I	**Valid input** The external agent asserts ValidIn* when it is driving a calid address or data on the SysAD bus and 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.

Table 2.2.3 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 00 41.8 MHz/50 MHz 167 MHz/200 MHz 01 66.7 MHz/80 MHz 167 MHz/200 MHz 10 83.5 MHz/100 MHz 167 MHz/200 MHz 11 55.7 MHz/66.7 MHz 167 MHz/200 MHz
TintDis	I	**Timer-Interrupt disable input** 0 enable Timer-Interrupt (Can not use Int 5*) 1 disable Timer-Interrupt (Can use Int 5*)
HALT/DOZE	O	**HALT/DOZE mode output** This signal output the status of HALT or DOZE mode. This signal indicates that the TX4955A is in the HALT or DOZE mode when this signal is "H".
PLLReset*	I	**PLL reset input** A signal to halt the PLL oscillation or the TX4955A 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

Table 2.2.4 Interrupt Interface

PIN NAME	I / O	FUNCTION
Int(5:0)∗	I	**Interrupt** Six lines of general-purpose processor interrupt inputs, which are sampled at rising edges of the Master Clock. Note: Int(5)∗ can use when TintDis signal is "1".
NMI∗	I	**Nonmaskable interrupt** Nonmaskable interrupt input, which is sampled at rising edges of the Master Clock.

Table 2.2.5 JTAG Interface

PIN NAME	I / O	FUNCTION
JTDI	I	**JTAG data input / Debug interrupt input** Run-time mode: Input serial data to JTAG data/instruction registers. Real-time mode: Interrupt input to change the debug unit state from real-time mode to run-time mode.
JTCK	I	**JTAG clock input** Clock input for JTAG. The JTDI and JTMS data are latched on rising edges of this clock.
JTDO/TPC(0)	O	**JTAG data output / Trace PC output** Data is serially shifted out from this pin. / Outputs a non-sequential program counter value synchronously with DCLK.
JTMS	I	**JTAG command** Controls mainly the status transition of the TAP controller state machine. When the serial input data is a JTAG command, apply a high signal (= 1) to this pin.
DCLK	O	**Debug clock (1/3 CPU clock)** Clock output for a real-time debug system. Timings of the serial monitor bus and PC trace interface signals all are defined by this debug clock DCLK. The TX4955A's operating clock frequency is 1/3 that of Pclock. (Pclock is Internal CPU Clock)
PCST(8:0)	O	**PC trace status** Outputs PC trace status information and serial monitor bus mode.
TPC(3:1) Note 1	O	**Trace PC output** Outputs a non-sequential program counter value synchronously with DCLK.
TRST∗	I	**Test reset input** Reset input for a real-time debug system. When TRST∗ is asserted (= 0), the debug support unit (DSU) is initialized. TRST∗ should be asserted when DSU is not used.

Note1: Leave TPC (3-1) pins open when not using them as PC trace outputs for debugging.

Table 2.2.6 Initialization Interface

PIN NAME	I / O	FUNCTION
Reset*	I	**Reset** Assert this signal for all reset sequences. Also, make sure it is applied synchronously with the Master Clock.
ColdReset*	I	**Cold reset** Assert this signal at power-on and for a cold reset. SClock starts operating synchronously with this signal.
MODE43*	I	**SysAD bus protocol selection** The high or low level of this input signal at power-on or cold reset selects the SysAD bus protocol. 　　Low level: TX4300 type protocol 　　High level: R5000 type protocol
BufSel(1:0)	I	**Output Buffer Select** Select the output buffer type of data bus and control signals. BufSel[1:0]　　Buffer output rate 　　00　　　　　50 % 　　01　　　　　reserved 　　10　　　　　150 % 　　11　　　　　100 % (Same as TX4955)
PLLCAP	I	**PLL adjusting capacitor** Reserved

Table 2.2.7 Power Supply

PIN NAME	I / O	FUNCTION
VccPLL	－	**Vcc for the PLL** This is a static Vcc for the internal Phase Locked Loop. (Apply a 1.5 V power supply.)
VssPLL	－	**Vss for the PLL** This is a static Vss for the internal Phase Locked Loop.
VccIO	－	**VccIO** This is a 3.3 V power supply pin.
VccInt	－	**VccInt** This is a 1.5 V power supply pin.
Vss	－	**Vss** This is the ground pin.

Chapter 3.   Initialization Interface

This chapter describes the TX4955A Initialization interface, and the processor modes. This includes the reset signal description and types, and initialization sequence, with signals and timing dependencies, and the user-selectable TX4955A processor modes.

Signal names are listed in bold letters—for instance the signal **MasterClock** indicates the processor clock. Low-active signals are indicated by a trailing asterisk, such as **ColdReset***, the power-on/cold reset signal.

3.1   Functional Overview

The TX4955A processor has the following two types of resets; they use the **ColdReset*** and **Reset*** input signals.

- **Cold Reset** is asserted after the power supply is stable and then restarts all clocks. A cold reset completely reinitializes the internal state machine of the processor without saving any state information.

- **Warm Reset** restarts processor, but does not affect clocks. A warm reset preserves the processor internal state.

After reset, the processor is bus master and drives the *SysAD* bus.

For reset vector address, use Oxbfc0 0000.

In the TX49/H2 processor core, the reset vector is located in address space without caching. Therefore, the cache and TLB need not be initialized at reset processing.

3.1.1   System Coordination

Care must be taken to coordinate system reset with other system elements. In general, bus errors immediately before, during, or after a reset may result in unpredicted behavior. Also, a small amount of processor state is guaranteed as stable after a reset of the TX4955A processor, so extreme care must be taken to correctly initialize the processor through software.

The operation of each type of reset is described in sections that follow. Refer to Figure 3.2.1 and Figure 3.2.2 later in this chapter for timing diagrams of the cold and warm resets.

3.2 Reset Signal Description

This section describes the two reset signals, **ColdReset*** and **Reset***.

- **ColdReset***: the **ColdReset*** signal must be asserted[Note] (low) to reset the processor. Internal clock begins to cycle and is synchronized with the deasserted edge (high) of **ColdReset***. **ColdReset*** can be asserted and deasserted asynchronously with the rising edge of **MasterClock**.

- **Reset***: the **Reset*** signal is asserted synchronously to initiate a warm reset. The **Reset*** signal must be deasserted synchronously with **MasterClock**.

Note: Asserted means the signal is true, or in its valid state. For example, the low-active Reset* signal is said to be asserted when it is in a low (true) state.

3.2.1 Cold Reset

A cold reset is used to completely reset the processor, including processor clocks. Information about saving processor states is given below.

Once power to the processor is established, the **ColdReset*** signal is asserted for 64,000 **MasterClock** cycles to ensure time for the processor clocks to lock to the input **MasterClock**. **ColdReset*** can be asserted and deasserted asynchronously with the rising edge of **MasterClock**.

Reset* can be asserted/deasserted when **ColdReset*** is asserted, but its value must not be changed during the reset sequence.

Upon reset, the processor becomes bus master and drives the **SysAD** bus. After **Reset*** is deasserted, the processor branches to the Reset exception vector and begins executing the Reset exception code.

For information on the bit states during a cold reset, see the section on Cold Reset exceptions in CPU Exception Processing.

3.2.2 Warm Reset

A warm reset is used to reset the processor without affecting the clocks; in other words, a warm reset is a logic reset.

Asserting the **Reset*** signal resets the processor without disrupting the clocks and allows the processor to retain as much of its state as possible for debugging. (For information on saving the processor states, see the section on Soft Reset exception in CPU Exception Processing.) Because asserting the **Reset*** signal results in an immediate warm reset, multicycle instructions such as cache misses or floating-point instructions may be aborted and some data lost as a result.

A warm reset is started by assertion of the **Reset*** pin. **Reset*** must be asserted for a minimum of 16 cycles, and must be asserted and deasserted synchronously with **MasterClock**. In general, data in the processor is preserved for debugging purposes.

Upon reset, the processor becomes bus master and drives the **SysAD** bus. After **Reset*** is deasserted, the processor branches to the Reset exception vector and begins executing the reset exception code.

If **Reset*** is asserted in the middle of a **SysAD** transaction, care must be taken to reset all external agents to avoid **SysAD** bus contention.

Figure 3.2.1 and Figure 3.2.2 show the timing diagrams for the cold and warm resets.
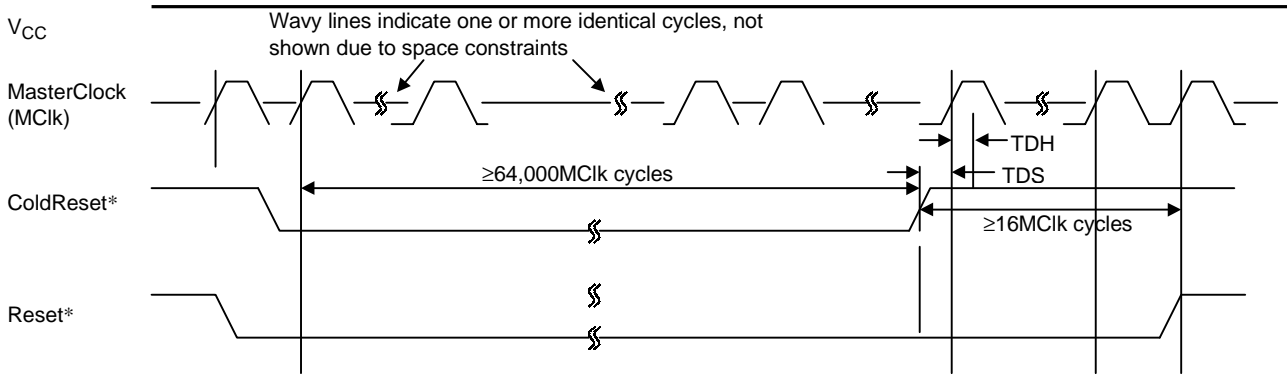
Figure 3.2.1 Cold Reset
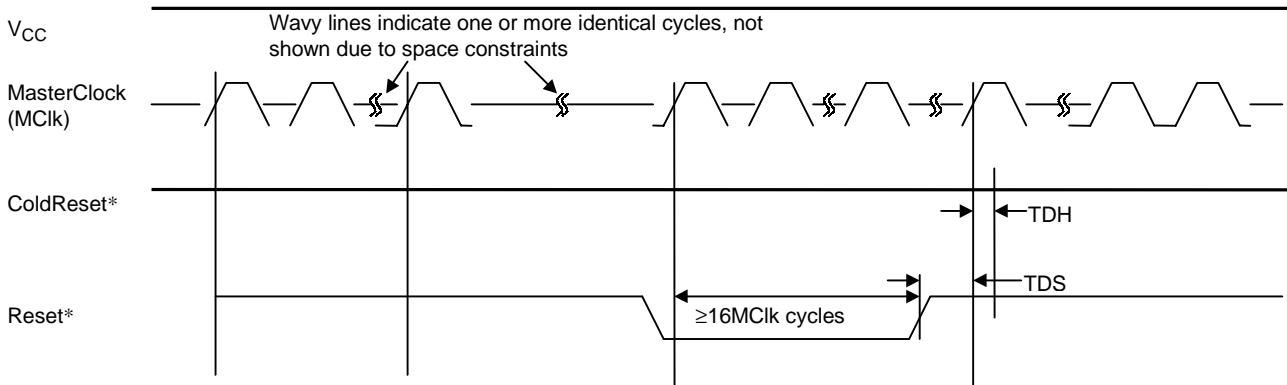
Figure 3.2.2 Warm Reset
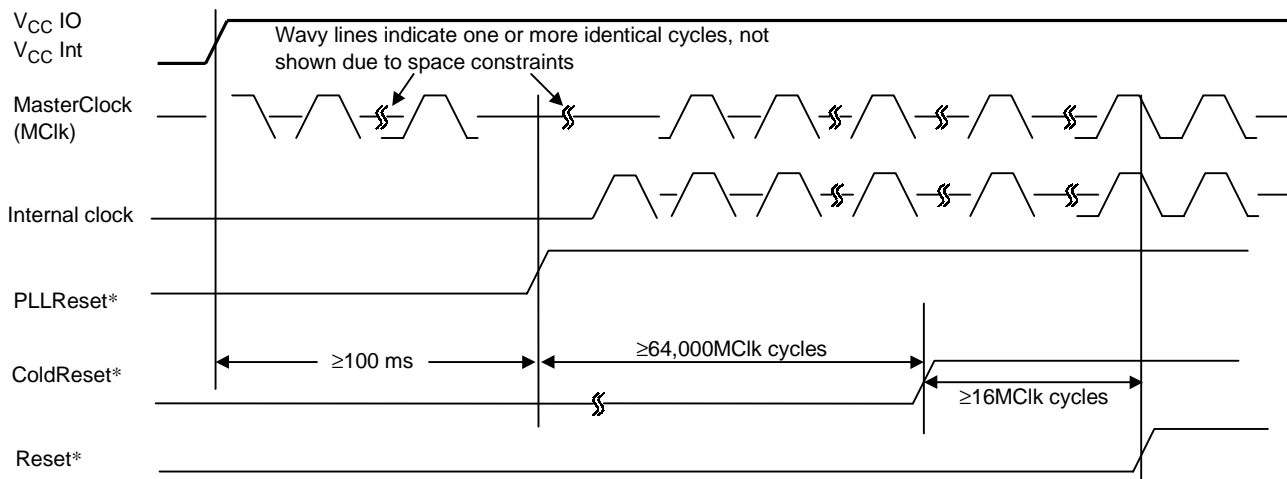
Note: As for the PLLReset∗, there is not need to take the same period with Master Clock.

Figure 3.2.3 Power on Reset

3.3   TX4955A

The TX4955A processor supports several user-selectable modes. All modes except **DivMode** ratios are set/reset by writing to the *Status* and *Config* registers.

3.3.1   Power Modes (Doze/Halt Mode)

The TX4955A supports three power modes: normal operation, Doze, and Halt. This section describes these three modes. In TX4955A, there is a terminal showing that the status is Halt mode or Doze mode state.

Figure 3.3.1 show the status shifts in the operation mode.
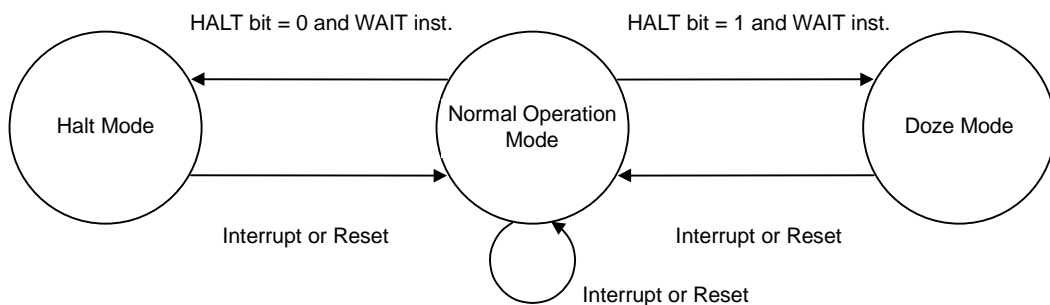
Figure 3.3.1 Status Shift Among Normal Operation Mode and Low Power Consumption Modes

- Normal Operation Mode

    Normally the TX4955A processor clock (**PClock**) operates at a multiple of the **MasterClock** speed (as designated by the **DivMode** setting). The System Interface Clock (**SClock**) operates at the same frequency as **MasterClock**.

- Doze Mode

    The user can specify Doze mode by setting the Halt bit in the Config Register to 1 after excuting a Wait instruction. Consumption power is bigger than later Halt mode so that a circuit maintaining pipeline status works in Doze mode.

    To recover from Doze mode, assert one of the following signal:

    Int[5:0]*, NMI*, Reset*, or ColdReset*.

    When recovering from Doze mode by asserting a non-enabled Int[5:0]* signal, the value of the Int Mask field in the Status Register is not affected.

    When recovering from Doze mode via the ColdReset* signal, NMI* signal, or an enabled Int[5:0]* signal, the top instruction of the exception handler corresponding to the signal is executed. On execution of the top instruction, the EPC points to instruction following the Wait instruction. When recovering via a non-enabled Int[5:0]* signal, the processor resumes execution from the instruction following the instruction used to change to Doze mode.

- Halt Mode

    The user can change to Halt mode by setting the Halt bit in the Config resister to 0 after excuting a Wait instruction.

    In Halt mode, operations are suspended with the pipeline state retained.

    Even requests for bus mastership are ignored. When an instruction that clears the

wait instruction is executed during a bus operation, the Halt mode is after the bus operation completes.

Write buffer operations do not stop during Halt mode: when the write buffer contain date, write operations continue until the write buffer empties.

To recover from Halt mode, assert one of the following signal:

Int[5:0]*, NMI*, Reset*, or ColdReset*.

When recovering from Halt mode by asserting a non-enabled Int[5:0]* signal, the value of the Int Mask field in the Status Register is not affected.

When recovering from Halt mode via the ColdReset* signal, NMI* signal, or an enabled Int[5:0]* signal, the top instruction of the exception handler corresponding to the signal is executed. On execution of the top instruction, the EPC points to instruction following the Wait instruction. When recovering via a non-enabled Int[5:0]* signal, the processor resumes execution from the instruction following the instruction used to change to Halt mode.

3.3.2 Operating Modes

The TX4955A has three operating modes, User mode, Supervisor mode and Kernel mode, that function in both 32- and 64-bit operations. The KSU, EXL and ERL bits in the Status register select User, Supervisor or Kernel mode. The UX, SX and KX bits in the Status register select 32-bit or 64-bit addressing for User, Kernel and Supervisor operating modes.

KSU	EXL	ERL	UX	SX	KX	Mode
10	0	0	0	—	—	32-bit addressing for User mode
10	0	0	1	—	—	64-bit addressing for User mode
01	0	0	—	0	—	32-bit addressing for Supervisor mode
01	0	0	—	1	—	64-bit addressing for Supervisor mode
00	—	—	—	—	0	32-bit addressing for Kernel mode
—	1	—	—	—	0	32-bit addressing for Kernel mode
—	—	1	—	—	0	32-bit addressing for Kernel mode
00	—	—	—	—	1	64-bit addressing for Kernel mode
—	1	—	—	—	1	64-bit addressing for Kernel mode
—	—	1	—	—	1	64-bit addressing for Kernel mode

3.3.3 System Endianness

By setting the External pin (Endian), the state of system endianness is provide to the processor.

Indicateds the initial setting of the endian during a reset.
Endian = 0: Little Endian
Endian = 1: Big Endian

Note: Be bit in the Config register is read only.

3.3.4   Reverse Endianness

When the *RE* bit in the *Status* register is set, endianess as seen by user software is reversed from its previous state.

3.3.5   Instruction Trace Support

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

3.3.6   Bootstrap Exception Vector

This bit is used when diagnostic tests cause exceptions to occur prior to verifying proper operation of the cache and main memory system.

When set, the Bootstrap Exception Vector (*BEV*) bit in the *Status* register causes the TLB refill exception vector to be relocated to a virtual address of 0xFFFF FFFF BFC0 0200 and the general exception vector relocated to address 0xFFFF FFFF BFC0 0380.

When *BEV* is cleared, these vectors are located at 0xFFFF FFFF 8000 0000 (TLB refill) and 0xFFFF FFFF 8000 0180 (general).

3.3.7   Interrupt Enable

When IE bit is clear, interrupts are not allowed, with the exception of reset and the nonmaskable interrupt.

3.3.8   Floating-Point Reigsters

By setting the FR bit in the Status register, the user accesses the full set of 32, 64-bit floating point registers as defined in the MIPS III ISA. When reset, the processor accesses the registers as defined in the MIPS II architecture.

Chapter 4. Clock Interface

This chapter describes the clock signals ("clocks") used in the TX4955A processor.

The subject matter includes basic system clocks, system timing parameters, operating the TX4955A processor in reduced power (RP) mode, connecting clocks to a phase-locked system, and connecting clocks to a system without phase locking.

4.1 Signal Terminology

The following terminology is used in this chapter (and book) when describing signals:

- *Rising edge* indicates a low-to-high transition.
- *Falling edge* indicates a high-to-low transition.
- *Clock-to-Q delay* is the amount of time it takes for a signal to move from the input of a device (*clock*) to the output of the device (*Q*).

Figure 4.1.1 and Figure 4.1.2 illustrate these terms.
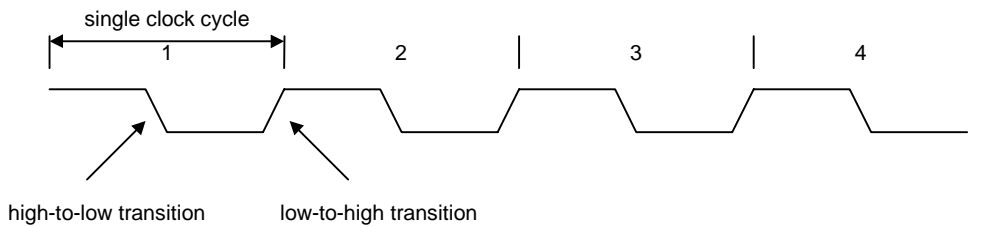
Figure 4.1.1 Signal Transitions
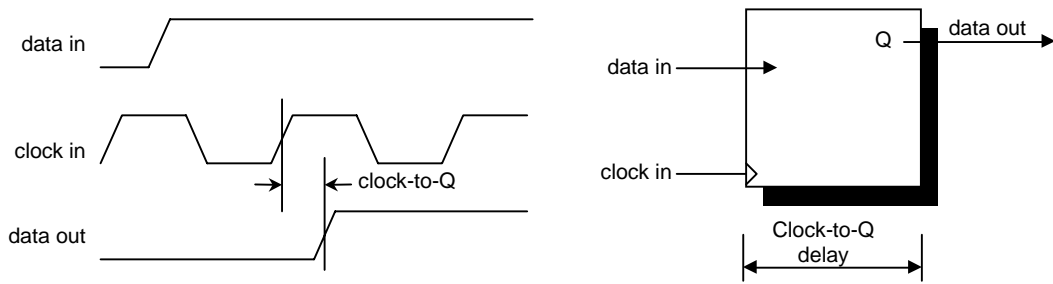
Figure 4.1.2 Clock-to-Q Delay

4.2   Basic System Clocks

The various clock signals used in the TX4955A processor are described below, starting with **MasterClock**, upon which the processor bases all internal and external clocking.

The clocks on the TX4955A processor are controlled by an on-processor Phase-locked Loop (PLL) circuit. This circuit keeps the TX4955A processor's internal clock edges aligned with the clock edges of the **MasterClock** signal, which itself acts as the master system clock.

Inside the TX4955A processor, the **MasterClock** signal can be multiplied by a factor set by the **DivMode (1:0)** inputs to the processor. All internal clocks are then derived from this clock. The TX4955A processor has two primary internal clocks, the pipeline (also referred to as *processor*) clock, **PClock**, and the system interface clock, **SClock**. **SClock** has the same frequency and phase as **MasterClock**.

4.2.1   MasterClock

The Processor bases all internal and external clocking on the single **MasterClock** input signal. **MasterClock** specifications are shown in Figure 4.2.1.
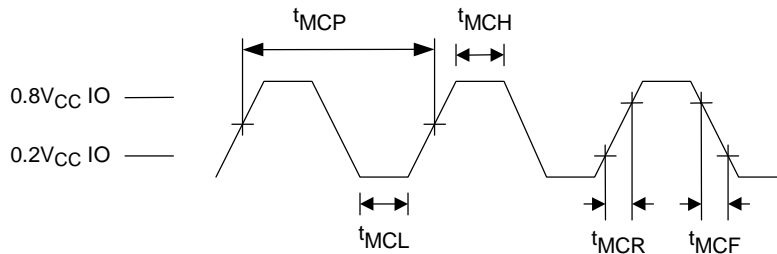
Figure 4.2.1 MasterClock

4.2.2   PClock

The pipeline (or *processor*) clock, **PClock**, can be 2, 2.5, 3 or 4 times the **MasterClock** frequency. This multiplication factor is determined by **DivMode (1:0)** pins, which are static signal inputs to TX4955A.

All internal registers and latches use **PClock**.

4.2.3   SClock

The system interface clock, **SClock**, is the same as the **MasterClock** frequency. **SClock** is always derived from **Pclock.** The TX4955A processor drives its outputs on this clock edge.

The first rising edge of **SClock**, after **ColdReset*** is deasserted, is aligned with the first rising edge of **MasterClock**.

4.2.4   PClock-to-SClock Division

Figure 4.2.2 shows the clocks for a PClock-to-SClock division by 2.
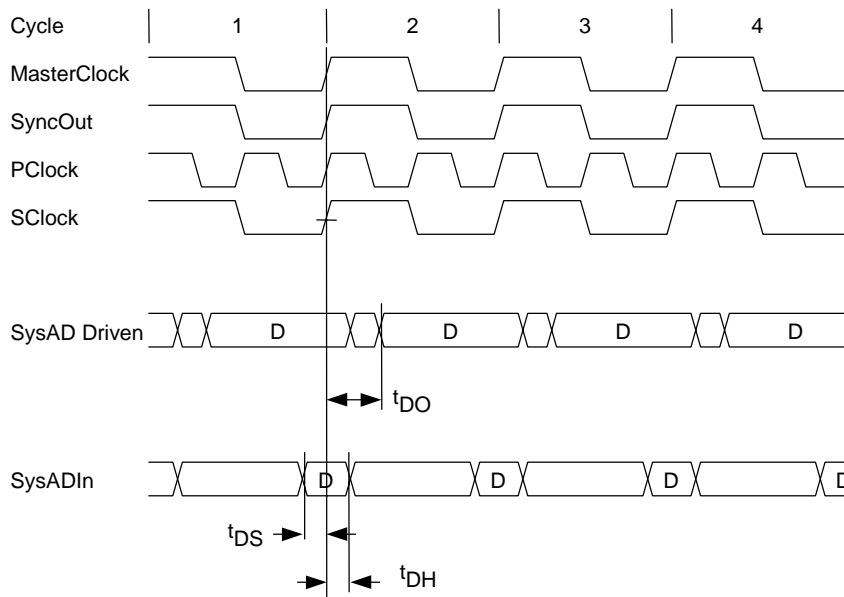
Figure 4.2.2 Processor Clock, PClock-to-SClock Divisor of 2

4.2.5   Phase-Locked Loop (PLL)

The TX4955A clocks are controlled by a Phase-locked Loop circuit (PLL).

4.3   Connecting Clocks to a Phase-Locked System

When the processor is used in a phase-locked system, the external agent must phase lock its operation to a common **MasterClock**. In such a system, the delivery of data and data sampling have common characteristics, even if the components have different delay values. For example, *transmission time* (the amount of time a signal takes to move from one component to another along a trace on the board) between any two components A and B of a phase-locked system can be calculated from the following equation:

$$\text{Transmission Time} = (\text{SClock period}) - (t_{DO} \text{ for A}) - (t_{DS} \text{ for B}) -$$
$$(\text{Clock Jitter for A Max}) - (\text{Clock Jitter for B Max})$$

Figure 4.3.1 shows a block-level diagram of a phase-locked system using the TX4955A processor.
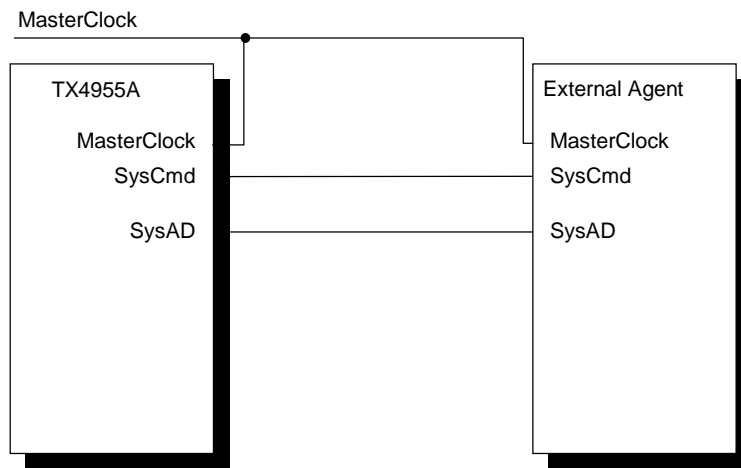
Figure 4.3.1 The TX4955A Processor Phase-Locked System

Chapter 5   Cache Organization

This chapter describes in detail the cache memory: its place in the TX4955A memory organization, and individual organization of the caches.

This chapter uses the following terminology:

- The data cache may also be referred to as the D-cache.

- The instruction cache may also be referred to as the I-cache.

These terms are used interchangeably throughout this book.

5.1   Memory Organization

Figure 5.1.1 shows the TX4955A system memory hierarchy. In the logical memory hierarchy, both primary and secondary caches lie between the CPU and main memory. They are designed to make the speedup of memory accesses transparent to the user.

Each functional block in Figure 5.1.1 has the capacity to hold more data than the block above it. For instance, physical main memory has a larger capacity than the caches. At the same time, each functional block takes longer to access than any block above it. For instance, it takes longer to access data in main memory than in the CPU on-chip registers.
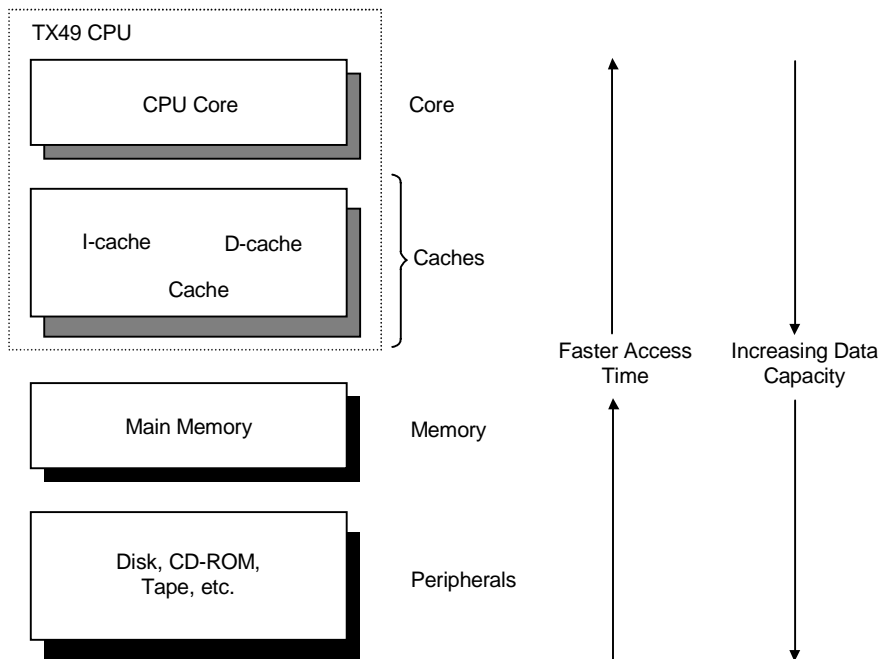
Figure 5.1.1 Logical Hierarchy of Memory

The TX4955A processor has two on-chip caches: one holds instructions (the instruction cache), the other holds data (the data cache). The instruction and data caches can be read in one **PClock** cycle.

Data writes are pipelined and can complete at a rate of one per **PClock** cycle. In the first stage of the cycle, the store address is translated and the tag is checked; in the second stage, the data is written into the data RAM.

5.2 Cache Organization

This section describes the organization of the on-chip data and instruction caches. Figure 5.2.1 provides a block diagram of the TX4955A cache and memory model.

I-cache: Instruction cache
D-cache: data cache
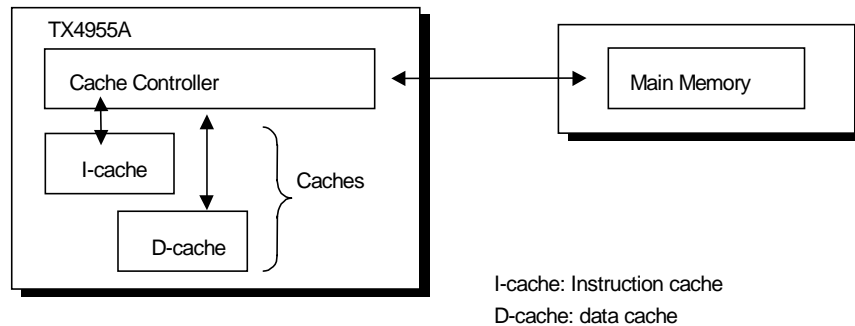
Figure 5.2.1 TX4955A Cache Support

5.2.1 Cache Sizes

The TX4955A instruction cache is 32 Kbytes; the data cache is 32 Kbytes.

5.2.2 Cache Line Lengths

A cache line is the smallest unit of information that can be fetched from main memory for the cache, and that is represented by a single tag.(Note)

The line size for the instruction cache is 8 words (32 bytes) and the line size for the data cache is 8 words (32 bytes).

Note: Cache tags are described in the following sections.

5.2.3 Organization of the Instruction Cache (I-Cache)

Each line of I-cache data (although it is actually an instruction, it is referred to as data to distinguish it from its tag) has an associated 24-bit tag.

The TX4955A processor I-cache has the following characteristics:

- Cache size: 32 KB
- Four-way set associative
- FIFO replacement
- Indexed with a virtual address
- Checked with a physical tag
- Block (line) size: 8 words (32 bytes)
- Burst refill size: 8 words (32 bytes)
- Lockable on a per-line basis (way1 ~ way3)
- All valid bits, lock and FIFO bits are cleared by a Reset exception

5.2.4   Instruction cache address field

Figure 5.2.2 shows the instruction cache address field. When 4-KB page size is in 32 KB I-cache, the bit 12 of the physical Address and the Virtual Address must be same value.

35	12	11	5 4	3 2	0
Physical Tag (24 bits)		Cache Tag Index (8 bits)	Word (2 bits)	Byte (3 bits)	
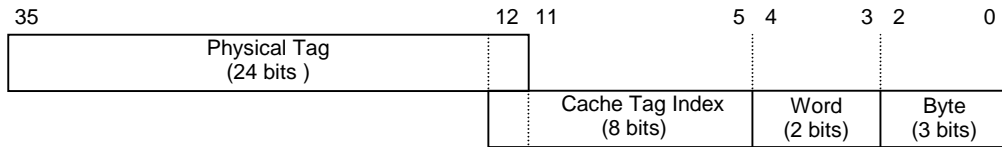
Figure 5.2.2 Instruction cache address field

5.2.5   Instruction cache configuration

Each line in the 4 ways of the instruction cache share F1, F0 replacement bits. Figure 5.2.3 shows the format of replacement bits. These bits are shared by way0, way1, way2 and way3 for 32 KB cache, and indicate next set to which replacement will be directed; when lock bit is set to 1, indicate this set is not locked.

Each line of instruction cache data has an associated 26-bit tag that contains a 24-bit physical address, a single Lock bit and a single valid bit, except for the line in way0, which has an 25-bit tag that excludes a lock bit. Figure 5.2.4 shows the formats of tag and data pair.

1	0
F1	F2
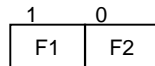
F0:   FIFO replace bit 0

F1:   FIFO replace bit 1

Figure 5.2.3 Format of replacement bits

24 23 0	63 0	63 0	63 0	63 0
V	P Tag	Data	Data	Data	Data

Format for way0

25 24 23 0	63 0	63 0	63 0	63 0
L	V	P Tag	Data	Data	Data	Data

Format for way1, 2 and 3
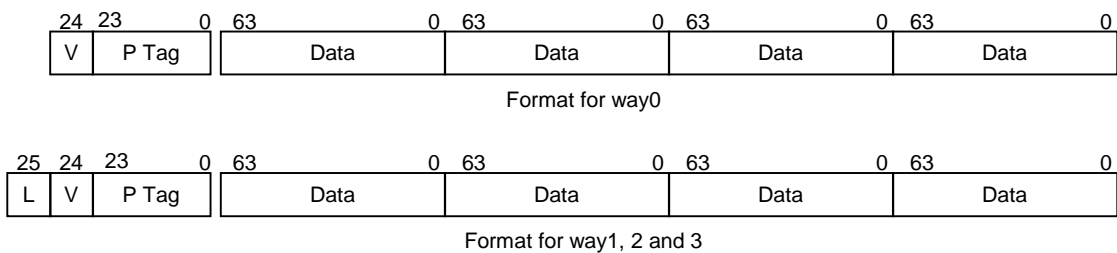
L: Lock bit (1: enable, 0: disable)

V: Valid bit (1: valid, 0: invalid)

PTag: Physical tag (bit 35~12 of the physical address)

Data: Instruction cache data

Figure 5.2.4 Format of tag and data pair for I-cache

5.2.6   Organization of the Data Cache (D-Cache)

Each line of D-cache data has an associated 24-bit tag.

The TX4955A processor D-cache has the following characteristics:

- Cache size: 32 KB
- Four-way set associative
- FIFO replacement
- Indexed with a virtual address
- Checked with a physical tag
- Block (line) size: 8 words (32 bytes)
- Burst size: 8 words (32 bytes)
- Store buffer
- Lockable on a per-line basis (way1 ~ way3)
- Write-back or write-through on a per-page basis
- All write-back, CS, FIFO and lock bits are cleared by a Reset exception

5.2.7   Data cache address field

Figure 5.2.5 shows the data cache address field.   When 4-KB page size is used in 32KB D-cache, the bit 12 of the Physical Address and the Virtual Address must be same value.
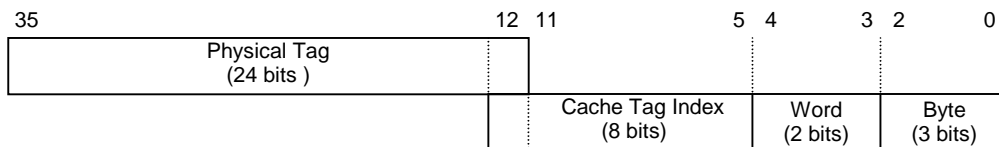
Figure 5.2.5 Data cache address field

5.2.8   Data cache configuration

Each line in the 4 ways of the data cache share F1, F0 replacement bits. Figure 5.2.6 shows the format of replacement bits. These bits are shared by way0, way1, way2 and way3 for 32 KB cache, and indicate next set to which replacement will be directed; when lock bit is set to 1, indicate this set is not locked.

Each line of data cache data has an associated 28-bit tag that contains a 24-bit physical address, a single Lock bit, a single write-back bit and a 2-bit cache state, except for the line in way0, which has an 27-bit tag that excludes a Lock bit. Figure 5.2.7 shows the formats of tag and data pair.
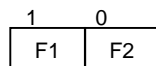
F0:   FIFO replace bit 0

F1:   FIFO replace bit 1

Figure 5.2.6 Format of replacement bits

26	25 24	23 0	63 0	63 0	63 0	63 0
W	CS	PTag	DATA	DATA	DATA	DATA

Format for way0

27	26	25 24	23 0	63 0	63 0	63 0	63 0
L	W	CS	PTag	DATA	DATA	DATA	DATA

Format for way1, 2 and 3
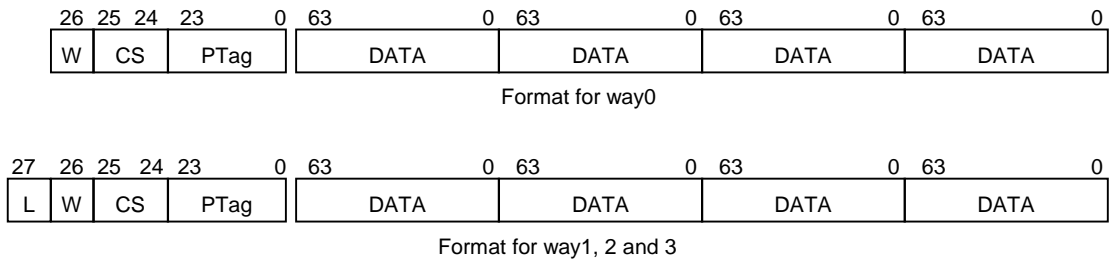
L: Lock bit (1: enable, 0: disable)
W: Write-back bit (set if cache line has written)
CS: Primary cache state (0: Invalid, 1: Reserved, 2: Reserved, 3: Valid)
PTag: Physical tag (bit 35~12 of the physical address)
Data: Data cache data

Figure 5.2.7 Format of tag and data pair for D-cache

In the TX4955A, the W (write-back) bit, not the cache state, indicates when the primary cache contents modified data that must be written back to memory. The states Invalid and Dirty Exclusive are used to describe the cache line. That is, there is no hardware support for cache coherency.

5.3   Lock function

The lock function can be used to locate critical instruction/data in one instruction/data cache set and they are not replaced when the lock bit is set.

5.3.1   Lock function

Setting the Lock bit in each line cache enable the instruction/data cache lock function. When the lock function is enabled, the instruction/data in the valid line is locked and never be replaced. The set to be locked is pointed by FIFO bit. Refilled instruction/data during the lock function is enabled is locked. When a store miss occurs for the write-through data cache without write allocate, the store data is not written to the cache and will therefore not be locked.

The lock function is disabled by clearing the Lock bit in each line.

In order to clear or set the Lock bit in the cache, Cache instructions (Index store I-cache /D-cache Tag) can be used, and in order to load the instruction/data to cache from memory, another Cache instructions (Fill I-cache/D-cache) can be used (refer to Cache instruction).

Clear the lock bit as follows when data written to a locked line should be stored in main memory.

(1) Read the locked data from cache memory
(2) Clear the lock bit
(3) Store the data that was read

5.3.2   Operation during lock

After the lock bit is set for a line, the line can be replaced only when it's line state is invalid. The locked valid line can never be replaced. FIFO bit should point only to the set of locked invalid line or unlocked line.

A write access to a locked valid line takes place only to the cache not to the memory at Write Back mode. Both of the cache and the memory are replaced at Write Through mode.

5.3.3   Example of Data cache locking

During the load operation to the locked line of the cache, any interrupt should be disabled in order to avoid to lock the wrong data.

To lock data cache lines, the following sequence of codes could be used.

```
......................          /* Disable the interrupt */
mtc0  t0, TagLo                 /* Load data into TagLo reg */
cache 2 (D), offset (base)      /* Invalidate and lock line in desired set using
                                        Index_Store_Tag cache instruction */
cache 7 (D), offset (base)      /* Fill the cache line from desired memory location */
......................          /* Enable the interrupt */
```

5.3.4   Example of Instruction cache locking

To lock instruction cache lines, the following sequence of codes could be used:

```
......................          /* Disable the interrupt */
mtc0  t0, TagLo                 /* Load data into TagLo reg */
cache 2 (I), offset (base)      /* Invalidate and lock line in desired set using
                                        Index_Store_Tag cache instruction */
cache 5 (I), offset (base)      /* Fill the cache line from desired memory location */
......................          /* Enable the interrupt */
```

5.4   The primary cache accessing

Figure 5.4.1 shows the virtual address (VA) index to the primary cache. Each instruction and data cache size is 32 KB. The virtual address bits be used to index into the primary cache decided by the cache size.
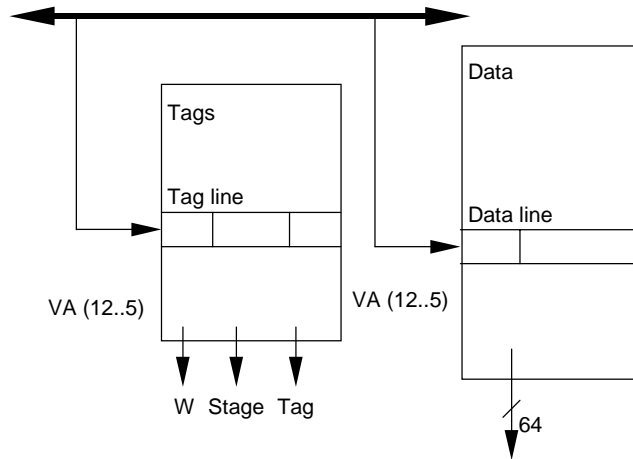
Figure 5.4.1 Primary Cache Data and Tag Organization

5.5   Cache States

The section describes about the state of a cache line. The cache line in the TX4955A are in one of states described in Table 5.5.1.

The I-Cache line is in one of the following states:
- invalid
- valid

The D-Cache line is in one of the following states:
- invalid
- valid

Table 5.5.1 Cache States

Cache line State	Description
Invalid	A cache line that does not contain valid information must be marked invalid, and cannot be used. A cache line in any other state than invalid is assumed to contain valid information.
Valid	A cache line contains valid information. The cache line may or not be consistent with memory and is owned by the processor (see Cache Line Ownership in this chapter).

5.6   Cache Line Ownership

The TX4955A becomes the owner of a cache line after it writes to that cache line (that is, by entering the dirty exclusive), and is responsible for providing the contents of that line on a read request. There can only be one owner for each cache line.

5.7   Cache Multi-Hit Operation

The TX4955A is not guaranteed the operation for the multi-hit of primary cache.

Thus, in case of locking the specified program/data in the primary cache, the program/data must be used after locked in the cache by Fill instruction.

Such as the previous description the cache multi hit does not guarantee in the TX4955A, however, if it occurs, the TX4955A will do the operation as follows.

- Load: read from the higher way of priority (way0 > way1 > way2 > way3).
- Store: write to all way that are multi-hits.

5.8   FIFO Replacement Algorithm

The instruction and data caches in the TX4955A use the FIFO replacement algorithm.

- Usually, cache elements are replaced in this order: Way0, Way1, Way2, Way3.
- The FIFO[1:0] replacement bits do not point to a locked, valid cache line.
- Data is first written to a cache line marked invalid, if any.
- The FIFO replacement bits change every time memory data is written to the cache or a CACHE instruction is executed.

Figure 5.8.1 shows several examples of how the FIFO replacement bits change.
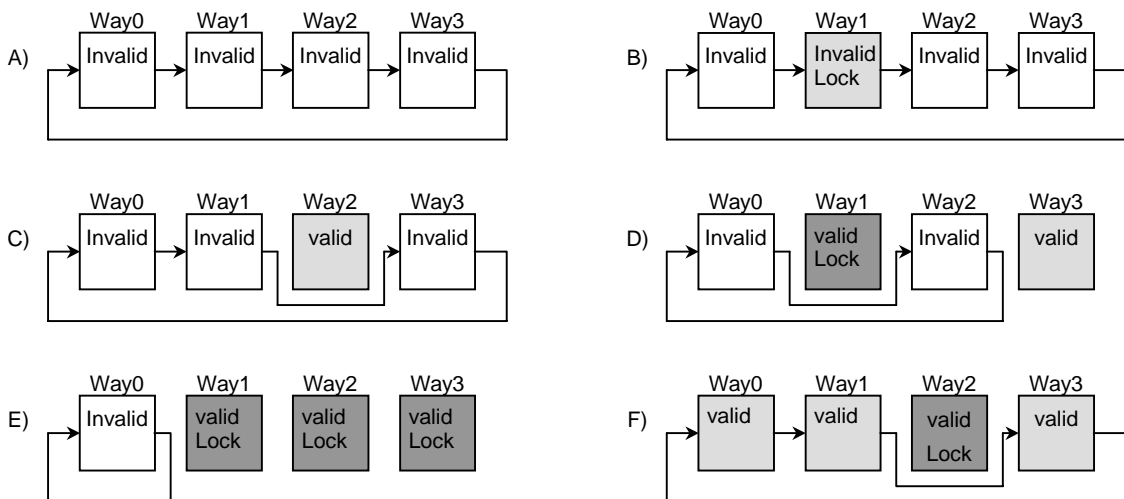
Figure 5.8.1 Examples of Cache State Transitions by the FIFO Replacement Algorithm

5.9   Cache Testing

5.9.1   Cache disabling

The ICE# and DCE# bits in the Config register enable and disable the instruction and data caches respectively.

When the cache is disabled, any attempt to access the cache causes a cache miss; therefore, a cache refill does not occur. (A burst bus cycle does not occur, either, as is the case with an access to a uncached memory space.) With the cache disabled, the Valid (V) and Cache State (CS) bits for each entry remain unchanged.

Note: When the instruction cache is disabled

All instruction fetches cause an instruction cache miss. External memory accesses will occur as a single-read operation.

Instruction cache operations by the CACHE instruction are valid.

Note: When the data cache is disabled

All memory accesses by the load and store instructions cause a data cache miss. At this time, no cache refill occurs. External memory accesses will occur as a single-read or single-write operation.

Data cache operations by the CACHE instruction are valid.

Note: How to disable the instruction cache reliably

To disable the instruction cache, stop instruction streaming by following the MTC0 instruction with a jump instruction, as shown below:

Example: MTC0 Rn, Config   (Set ICE# bit.)
            J   L1   (Jump to L1 and stop streaming.)
            NOP   (Jump delay slot)
        L1: CACHE IndexInvalidate, offset (base)

5.9.2   Cache Flushing

Both the instruction and data caches are flushed by a Cold Reset or Warm Reset exception. i:e., all the Valid and CS bits are cleared to zeroes.

The instruction cache is also flushed by the Index Invalidate and Hit Invalidate operations with a CACHE instruction. The data cache is flushed by the Hit Invalidate operation with a CACHE instruction.

Data is written back to the main memory when an Index Writeback Invalidate or Hit Writeback Invalidate operation is performed, when a Hit Writeback operation is performed, and when a cache line is replaced. When the write-back policy is employed, it is required to consciously maintain cache coherency when flushing the cache.

5.10 Cache Operations

As described earlier, caches provide fast temporary data storage, and they make the speedup of memory accesses transparent to the user. In general, the processor accesses cache-resident instructions or data through the following procedure:

1. The processor, through the on-chip cache controller, attempts to access the next instruction or data in the appropriate cache.

2. The cache controller checks to see if this instruction or data is present in the cache.

   - If the instruction/data is present, the processor retrieves it. This is called a cache *hit*.

   - If the instruction/data is not present in the cache, the cache controller must retrieve it from memory. This is called a cache *miss*.

3. The processor retrieves the instruction/data from the cache and operation continues.

It is possible for the same data to be in two places simultaneously: main memory and cache. This data is kept consistent through the use of a *write-back* methodology; that is, modified data is not written back to memory until the cache line is to be replaced.

Instruction and data cache line replacement operations are listed in described as the following sections.

Table 5.10.1 Cache Instruction

Name	Caches	Operation
Index Invalidate	Instruction	Sets the cache state of cache block to invalid.
Index Write Back Invalidate	Data	Examines cache state, if Valid Dirty, then that block is written back to main memory. Then the cache block is set to invalid.
Index Load Tag	Instruction & Data	Read the tag for the cache block at the specified index and place it into TagLo.
Index Store Tag	Instruction & Data	Write the tag for the cache block at the specified index from the TagLo register.
Create Dirty Exclusive	Data	If the cache does not contain the specified address, and the block is Valid Dirty the block will be written back to main memory. Then the tag will be set to the specified physical address and will be marked valid.
Hit Invalidate	Instruction & Data	If the cache block contains the specified address, cache block will be marked invalid.
Hit Write Back Invalidate	Data	If the cache block contains the specified address, and it is Valid Dirty, the data will be written back to main memory. Then, the cache block is marked invalid.
Fill	Instruction	Fill the Instruction cache block from main memory.
Fill	Data	Fill the Data cache block from memory.
Hit Write Back	Data	If the cache block contains the specified address, and it is marked Valid Dirty, the block will be written back to main memory, and marked Valid Clean.

5.10.1 Cache Write Policy

The TX4955A processor manages its data cache by using a write-back and a write-through policy. A write-back stores write data into the cache, instead of writing it directly to memory. Some time later this data is independently written into memory. In the TX4955A implementation, a modified cache line is not written back to memory until the cache line is to be replaced either in the course of satisfying a cache miss, or during the execution of a write-back CACHE instruction.

When the processor writes a cache line back to memory, it does not ordinarily retain a copy of the cache line, and the state of the cache line is changed to invalid.

A write-through is written simultaneously to cache and memory.

5.10.2 Data Cache Line Replacement

Since the data cache uses a write-back and a write-through methodology, a cache line load is issued to main memory on a load or store miss, as described below. After the data from memory is written to the data cache, the pipeline resumes execution.

TX4955A does not support "Critical Data Word First". Always it transfer the data of first address.

Rules for replacement on data load and data store misses are given below.

- **Data Load Miss**
  If the missed line is not dirty, it is replaced with the requested line.
  If the missed line is dirty, it is moved to the write buffer. The requested line replaces the missed line, and the data in the write buffer is written to memory.

- **Data Store Miss**
  If the missed line is not dirty, the requested line is merged with the store data and written to memory.
  If the missed line is dirty, it is moved to the write buffer. The requested line is merged with the store data and written to cache, and data in the write buffer is written to memory.
  The data cache miss penalties, in number of SClock cycles, are given in Table 5.10.2.

Table 5.10.2 Data Cache Refill Penalty Cycle Count

Number of PClock Cycles	Action
1	Stall the DC stage.
1	Transfer address to the write buffer and wait for the pipeline start signal
1 ~ 2	Transfer address to the internal SysAD bus on the SClock.
2	Transfer to the external SysAD bus.
M	Time needed to access memory, measured in PClock cycles.
4	Transfer the cache line form memory to the SysAD bus.
2	Transfer the cache line from the external bus to the internal bus.
0	Restart the DC stage.

5.10.3   Instruction Cache Line Replacement

For an instruction cache miss, refill is done using sequential ordering, starting from the first word of the retrieved cache line.

During an instruction cache miss, a memory read is issued. The requested line is returned from memory and written to the instruction cache. At this time the pipeline resumes execution, and the instruction cache is reaccessed.

The replacement sequence for an instruction cache miss is:

1. Move the instruction physical address to the processor pads.
2. Wait for a PClock cycle, aligned with an SClock boundary, to occur.
3. Read the line from memory and write it out to the instruction cache array.
4. Restart the processor pipe.

The instruction cache miss penalties, in number of PClock, is given in Table 5.10.3.

Table 5.10.3 Instruction Cache Refill Penalty Cycle Count

Number of PClock	Action
1	Stall the RF stage.
1	Transfer address to the write buffer and wait for the pipeline start signal.
1 ~ 2	Transfer to the external SysAD bus.
2	Transfer to the external SysAD bus.
M	Time needsd to access memory, measured in PClock cycles.
8	Transfer the cache line from memory to the SysAD bus.
2	Transfer the cache line from the external bus to the internal bus.
0	Restart the RF stage.

5.11   Manipulation of the Caches by an External Agent

The TX4955A does not provide any mechanisms for an external agent to examine and manipulate the state and contents of the caches.

Chapter 6. TX4955A System Interface

6.1 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.).

6.2 Explanation of System Interface of R5000 type protocol mode

In TX4955A, it is built in system interface function corresponding to TX4300 type protocol. A selection of above-mentioned R5000 type protocol mode or TX4300 type protocol mode increases by external pin (MODE43* : 117 pin).

MODE43* = 0: TX4300 type protocol

MODE43* = 1: R5000 type protocol

The TX4955A processor supports 32-bit address/data interfaces. This processor makes it possible to construct a processor system by processors and main memory. System interfaces consist of the following components:

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

The TX4955A processor accesses external resources using the system interface in order to correct cache misses, uncached operation, and other problems.

6.2.1   Interface bus

Figure 6.2.1 illustrates the 32-bit address/data bus SysAD (31:0), 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.
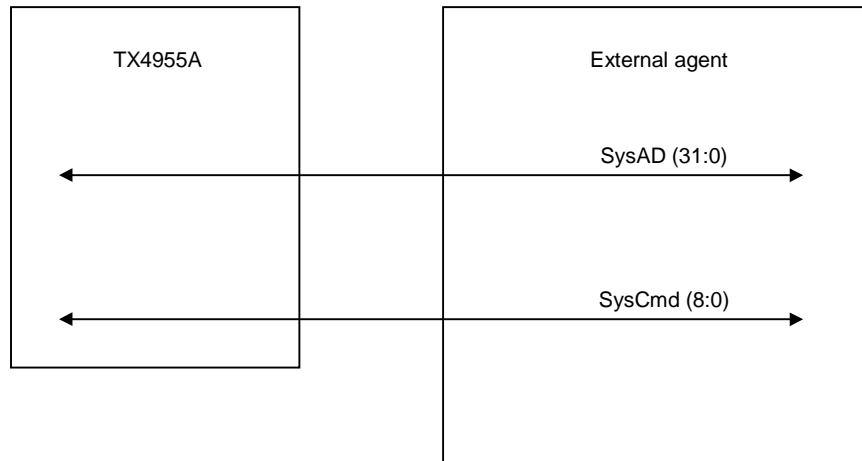
Figure 6.2.1 System Interface Bus

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.

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

6.2.3   Issue cycle

Two types of processor issue cycles exist with the TX4955A.

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

The TX4955A 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 6.2.2, RdRdy* must be asserted two cycles before the processor read request address cycle in order to define the address cycle as an issue cycle.
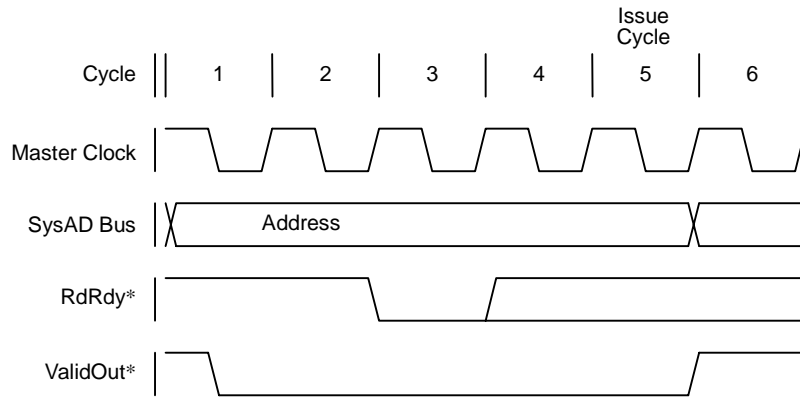
Figure 6.2.2 RdRdy* Signal Status in case of Read Request

As illustrated in Figure 6.2.3, 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.
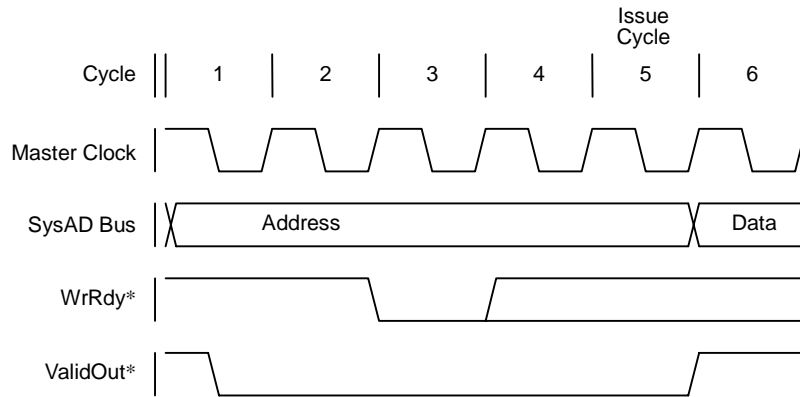
Figure 6.2.3 WrRdy* Signal Status in case of Write Request

The TX4955A 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 TX4955A 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 TX4955A 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 TX4955A issues processor requests after external requests are complete. Rules relating to issuing are also provided to processor requests.

6.2.4   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 ready to accept a new read or write transaction.

- 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 TX4955A processor uses ValidOut* and the external agent ValidIn* signals to indicate the valid command/data on the SysCmd/SysAD bus.

6.2.5   System Interface Protocol of R5000 type

Figure 6.2.4 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.
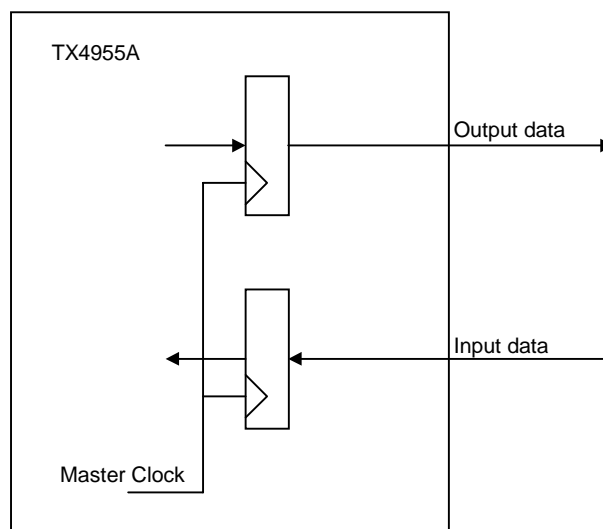
Figure 6.2.4 Operation of the System Interface Between Registers

6.2.5.1 Master state and slave state

The system interface is placed in the master state when the TX4955A 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.

6.2.5.2 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.

6.2.5.3 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.

6.2.5.4 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.

6.2.6   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 6.2.5 illustrates the system event and request cycles.
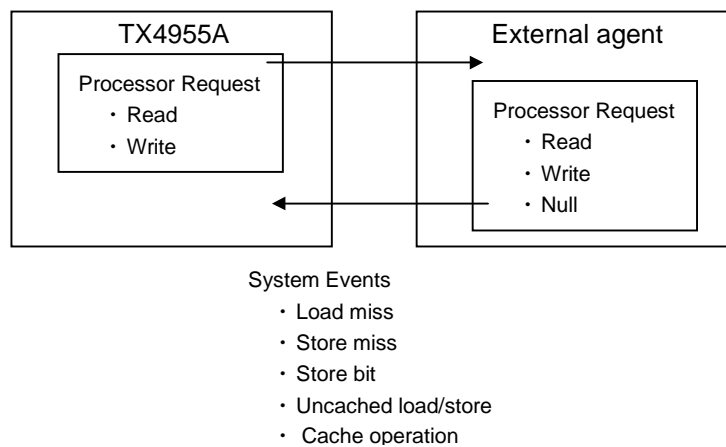
Figure 6.2.5 Requests and System Event

6.2.6.1   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 6.2.6 by consecutive write requests of single data cycles.
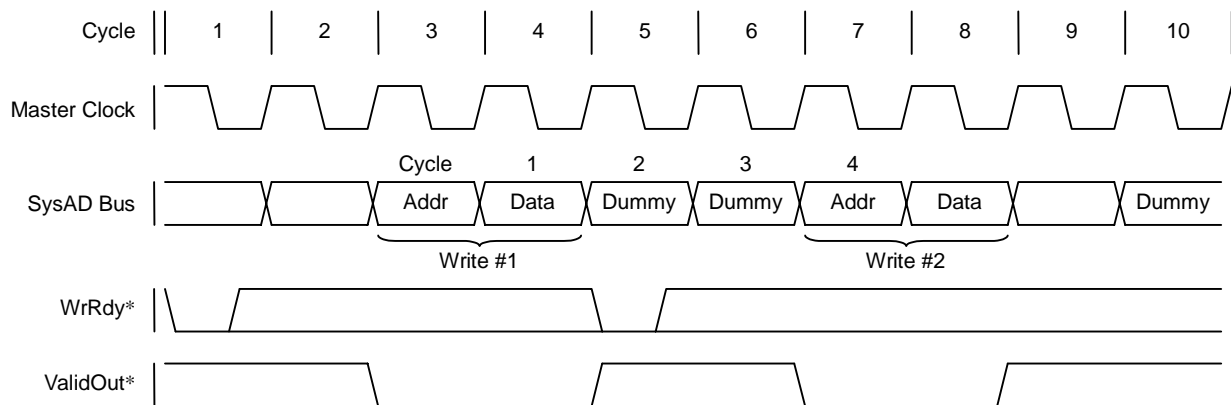
Figure 6.2.6 Timing of Consecutive Write Cycles

6.2.6.2 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 6.2.7, there are two types of processor request: read and write.
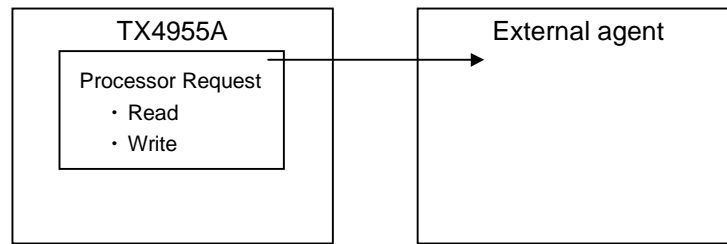
Figure 6.2.7 Processor Requests

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 TX4955A 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 6.2.8 illustrates the sequence of the processor request cycle.
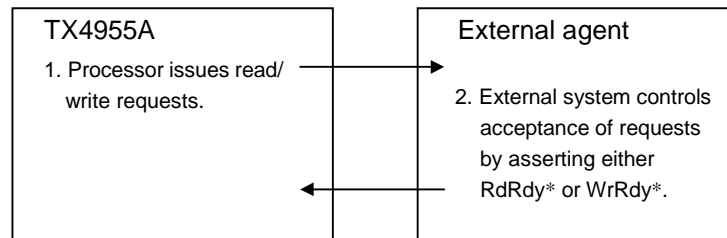
Figure 6.2.8 Processor Requests

6.2.6.3 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.

6.2.6.4 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.

6.2.6.5 External requests

As illustrated in Figure 6.2.9, there are three types of external request: read, write, and null. This section will explain read responses, which are special external request cases.
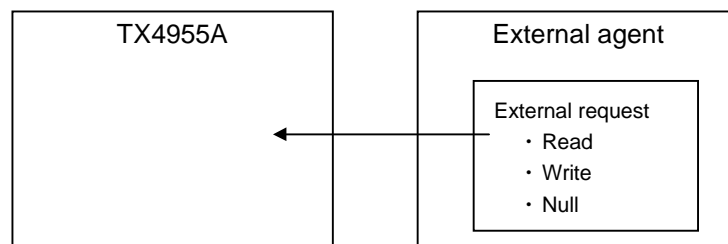
Figure 6.2.9 External Requests

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 6.2.10, 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.
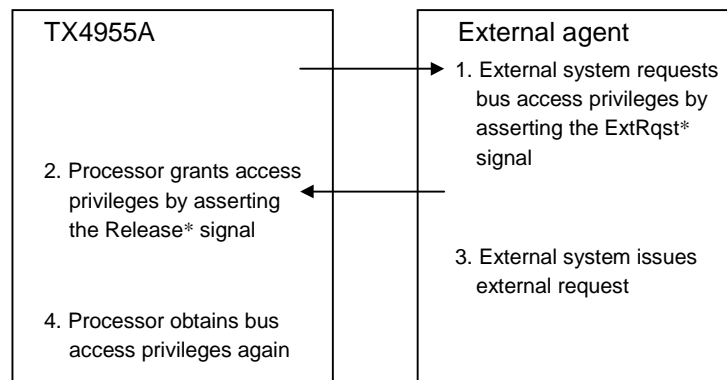
Figure 6.2.10 External Requests

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.

6.2.6.6   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 TX4955A 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.

6.2.6.7   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.

6.2.6.8   Read responses

As illustrated in Figure 6.2.11, 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.

TX4955A	External agent
1. Read request	
	2. Read response
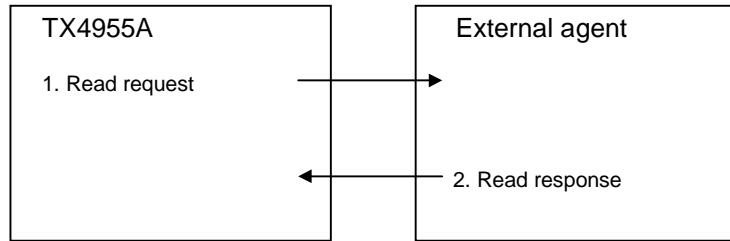
Figure 6.2.11 Read Response

6.2.7   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

6.2.7.1   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 6.2.1 indicates the measures that can be taken when a load miss occurs in the primary cache.

Table 6.2.1 Load Miss to 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

6.2.7.2   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 6.2.2 indicates the measures taken when there is a store miss to the primary cache.

Table 6.2.2 Store Miss to 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

6.2.7.3 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.

6.2.7.4 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.

The TX4955A 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 TX4955A 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.

6.2.7.5 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.

6.2.8   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 6.2.3 below describes the abbreviations used in the following timing diagram of the bus.

Table 6.2.3 System Interface Request

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

6.2.8.1   Processor request protocol

Processor request protocol is as follows.

- Read
- Write
- Null write

6.2.8.2   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 6.2.12.

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 6.2.12), 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 6.2.12 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.
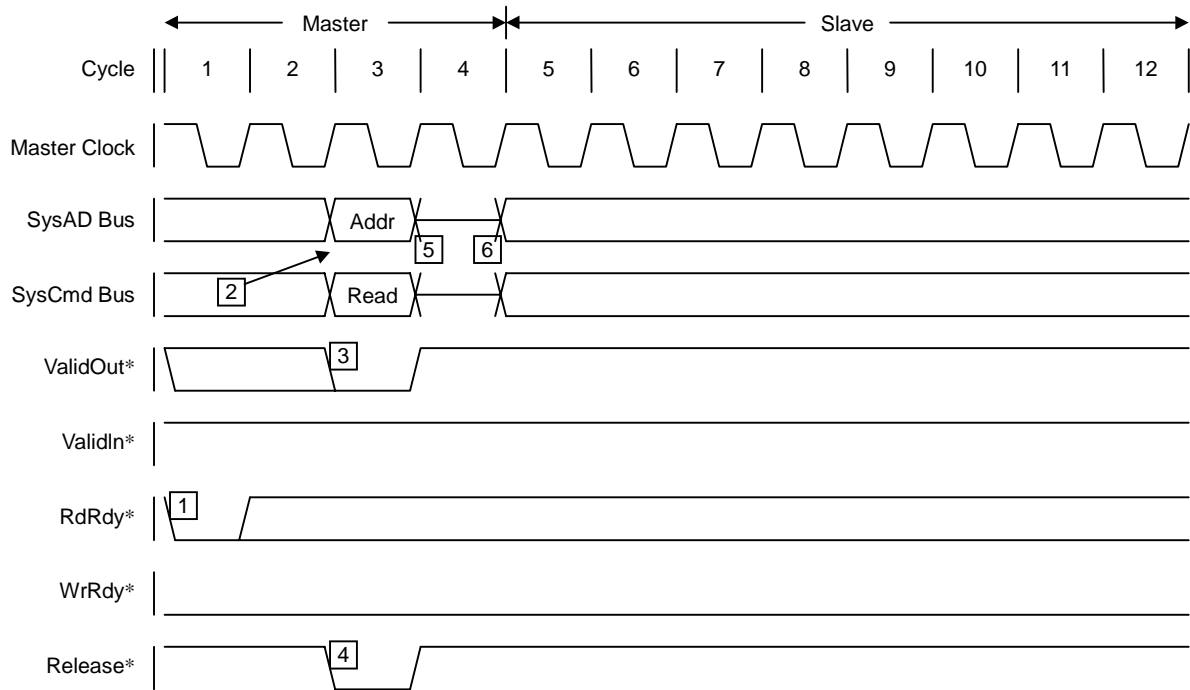
Figure 6.2.12 Processor Read Request Protocol

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.

6.2.8.3 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 6.2.13 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.
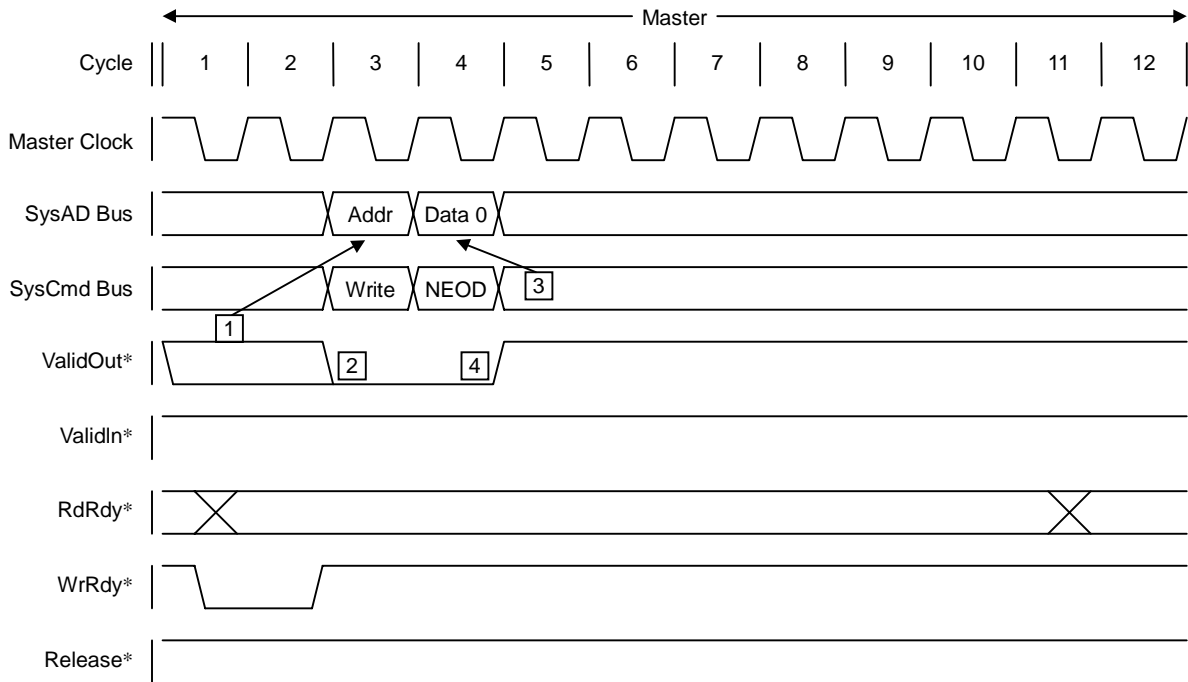
Figure 6.2.13 Processor Non-coherent Single Word Write Request Protocol

6.2.8.4   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

Table 6.2.4 Data Transfer Rate, Data Pattern and Setting at single write requests

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.

1. R4000 compatible write
When in the R4000 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. Figure 6.2.14 illustrates its basic operation.

In the case of the TX4955A, 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.
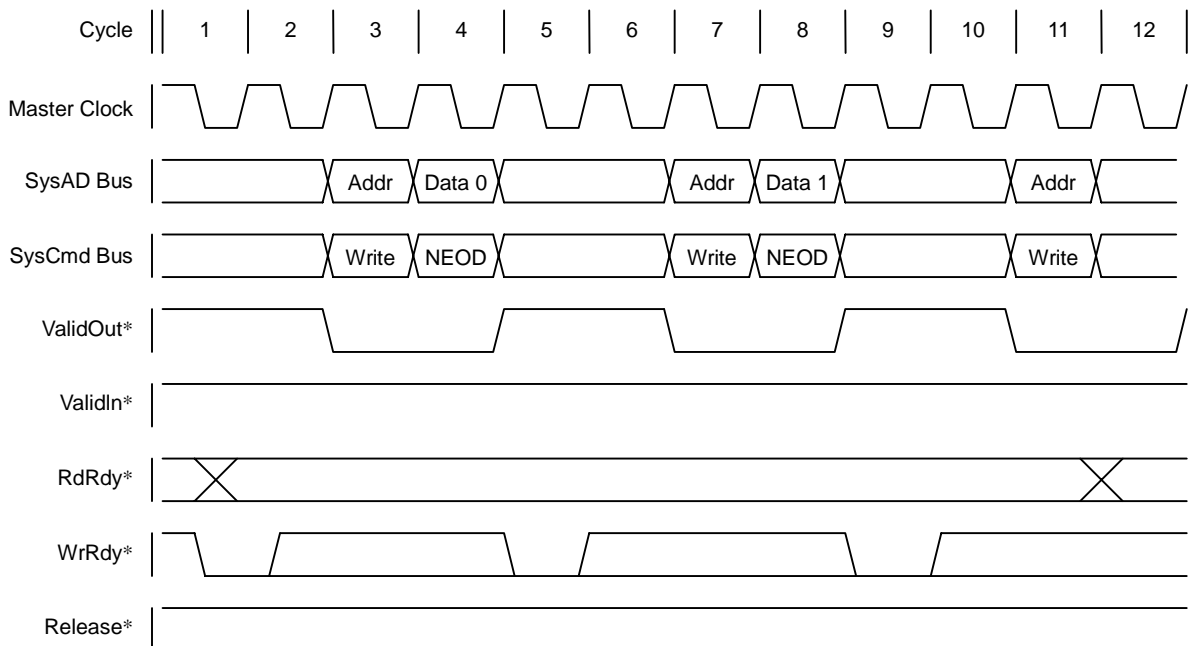
Figure 6.2.14 R4000 Compatible Write

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 6.2.15 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.
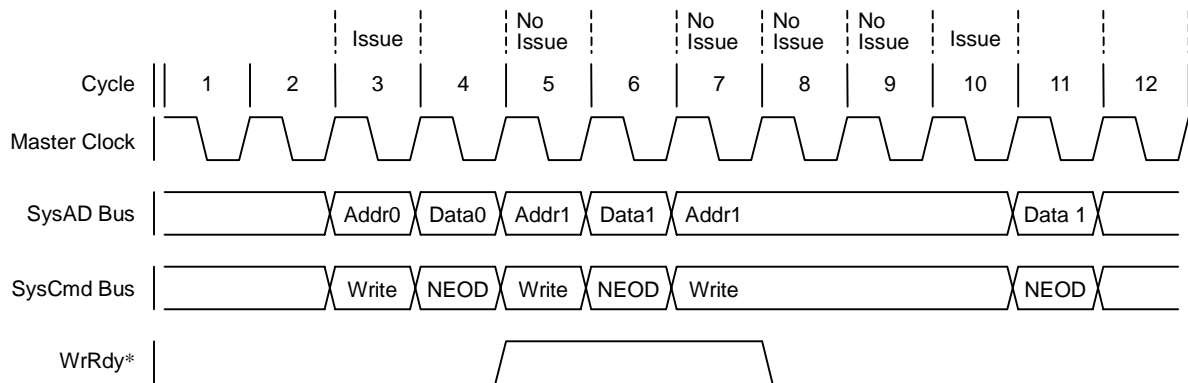
Figure 6.2.15 Reissue Write Protocol

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 6.2.16 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.
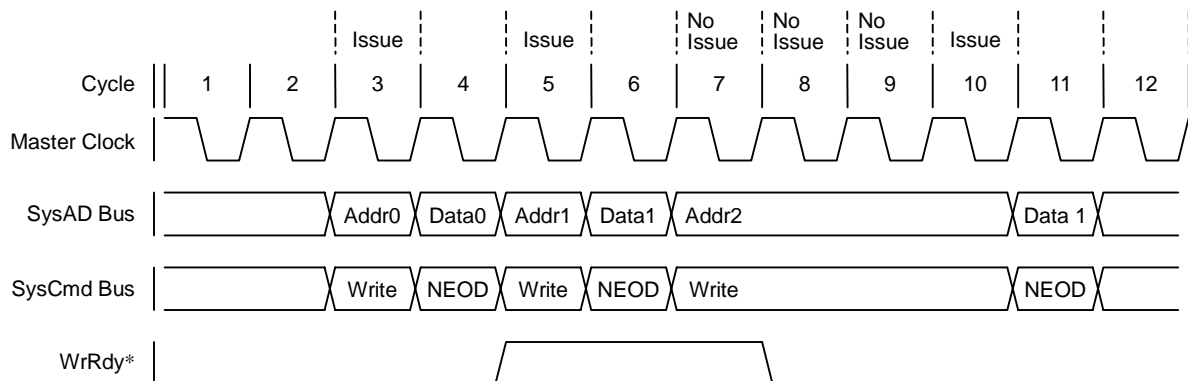
Figure 6.2.16 Pipeline Write Protocol

6.2.8.5   Processor block write request

The master state system interface is used to issue processor block write requests. Figure 6.2.17 illustrates a processor non-coherent block request made for 8-word data with the "D" data pattern.

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.
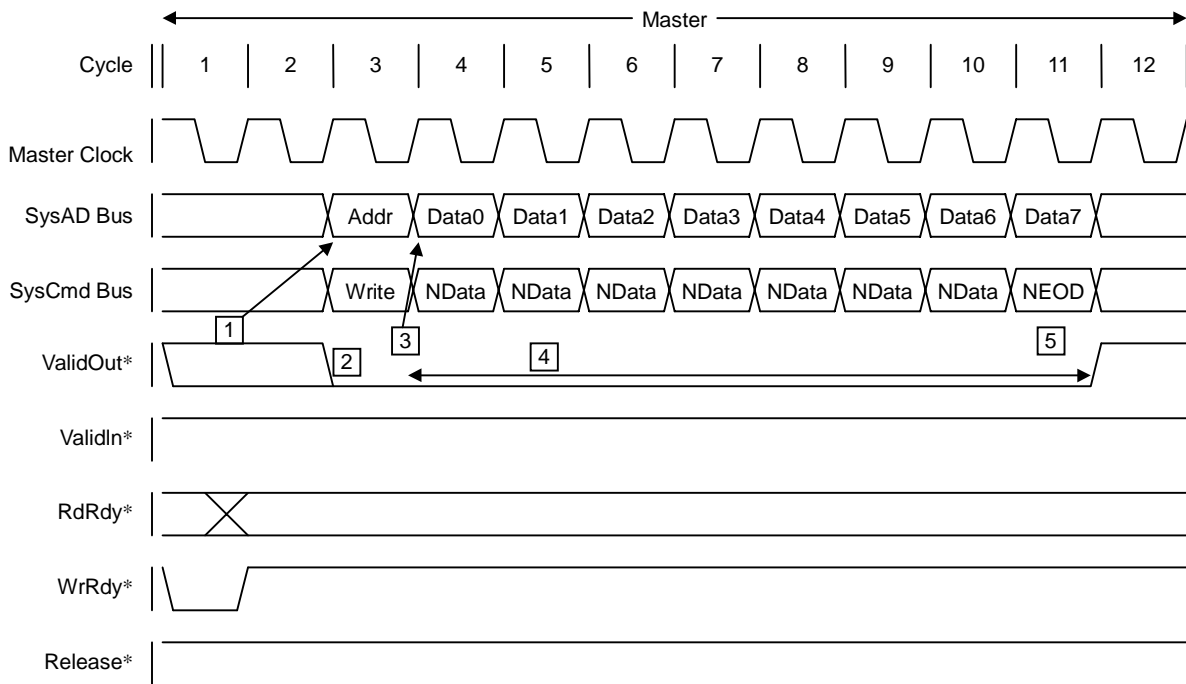
Figure 6.2.17 Processor non-coherent block request protocol

6.2.8.6   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.

6.2.8.7 External arbitration protocol

As previously mentioned, the ExtRqst* signal and Release* signal are used in system interface arbitration. Figure 6.2.18 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.
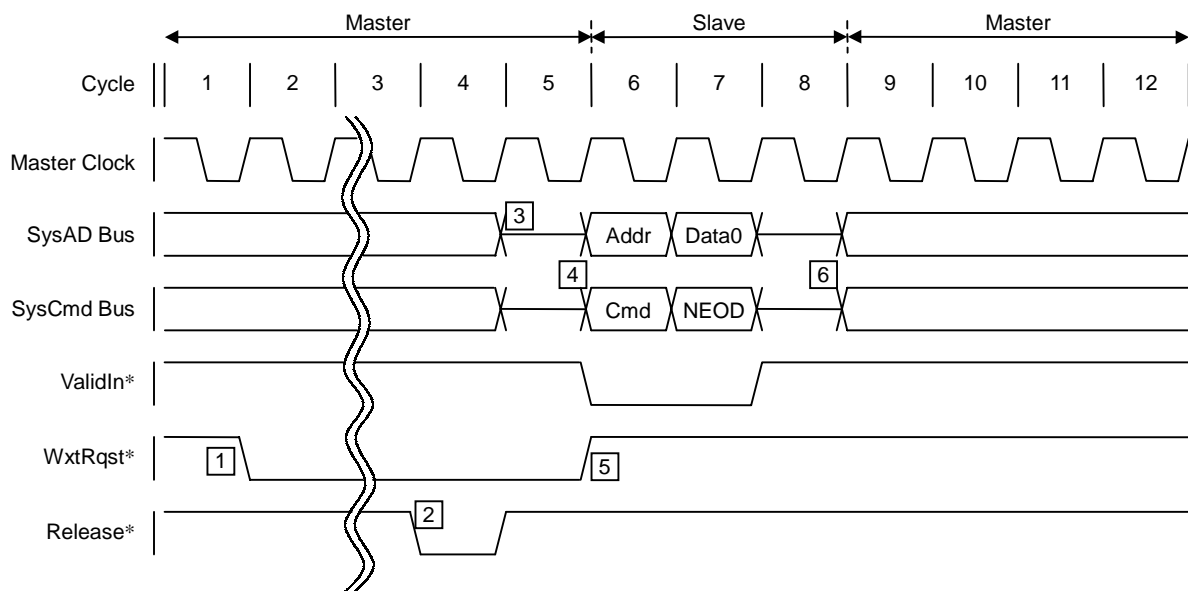
Figure 6.2.18 Arbitration Protocol Relating to External Requests

6.2.8.8   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 6.2.19 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.
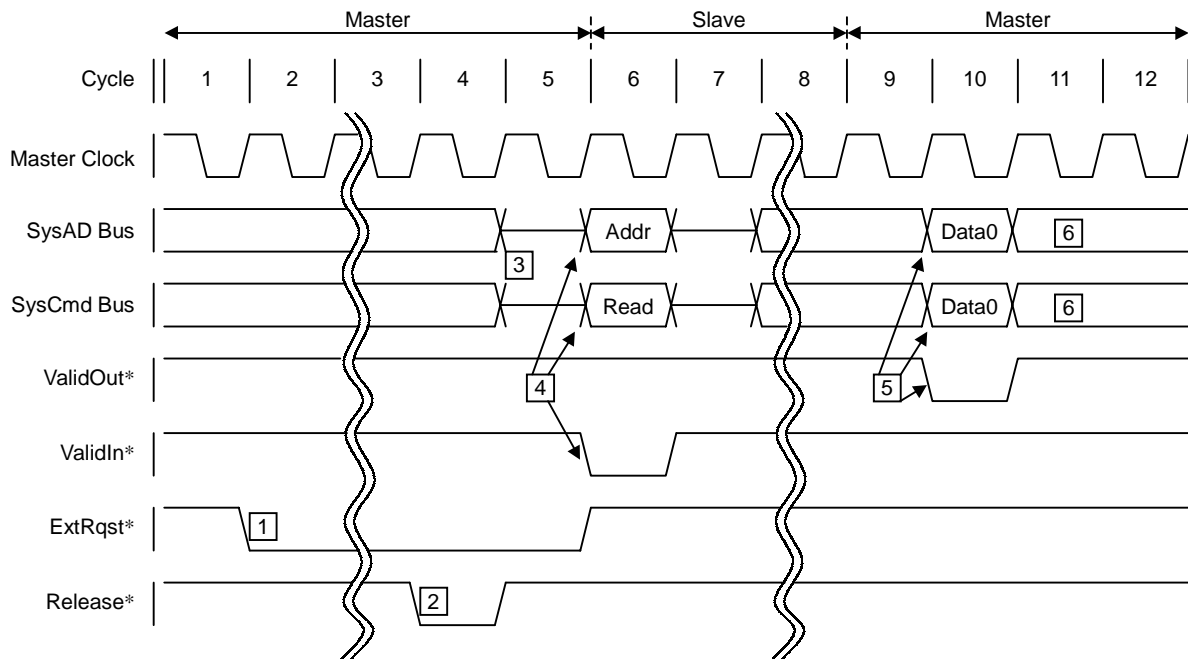
Figure 6.2.19 External Read Request when System Interface is in the Master State

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.

6.2.8.9   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 6.2.20 illustrates the timing of the external null request, which consists of the following steps.

1.   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.

2.   The SysAD bus is not available during the external null request address cycle (there are no valid data in the bus).

3.   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.
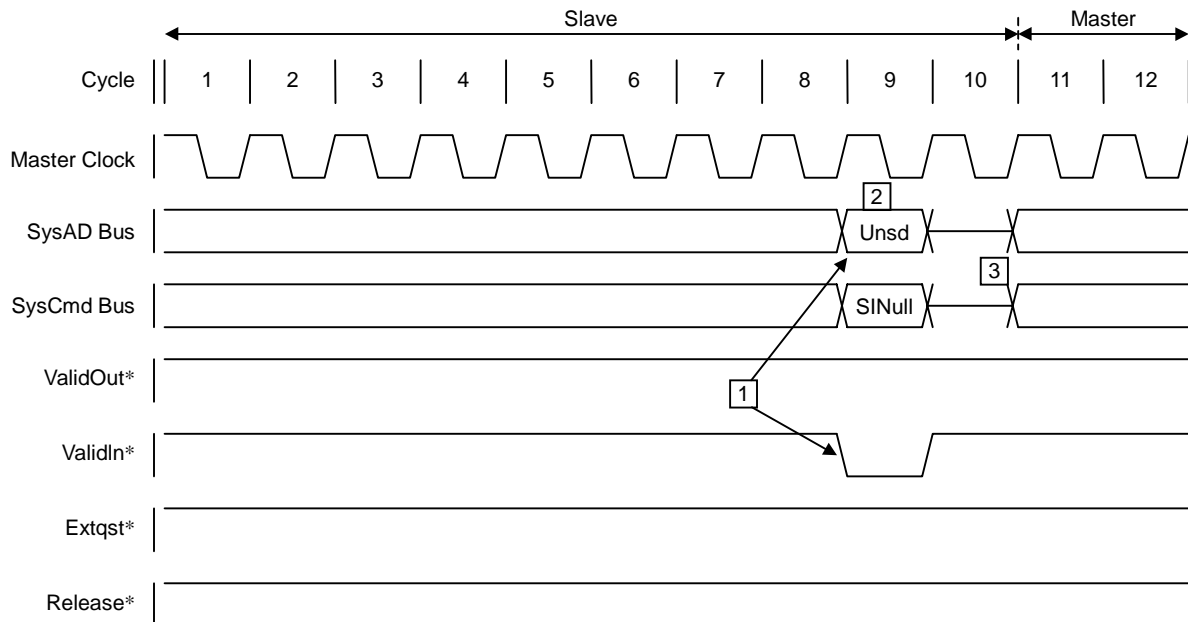
Figure 6.2.20 System Interface Release External Null Request

6.2.8.10 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 6.2.21 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.
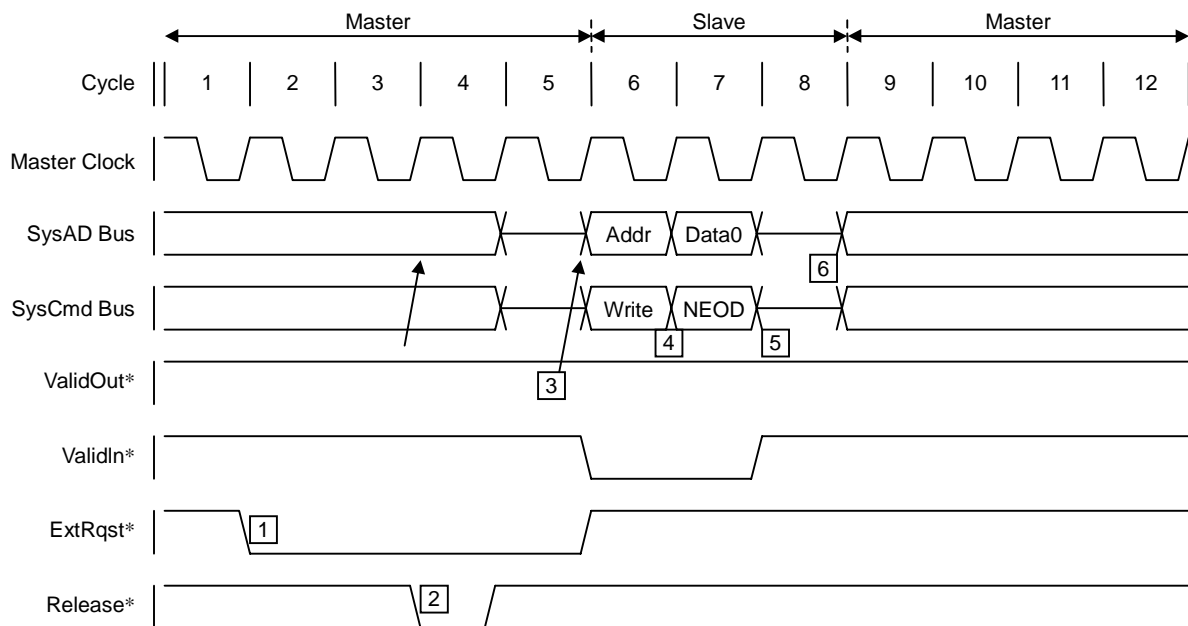
Figure 6.2.21 External Write Request when System Interface Starts in the Master State

6.2.8.11 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 external agent 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 6.2.22 illustrates a processor word read request and the subsequent word read response. Also, Figure 6.2.23 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.
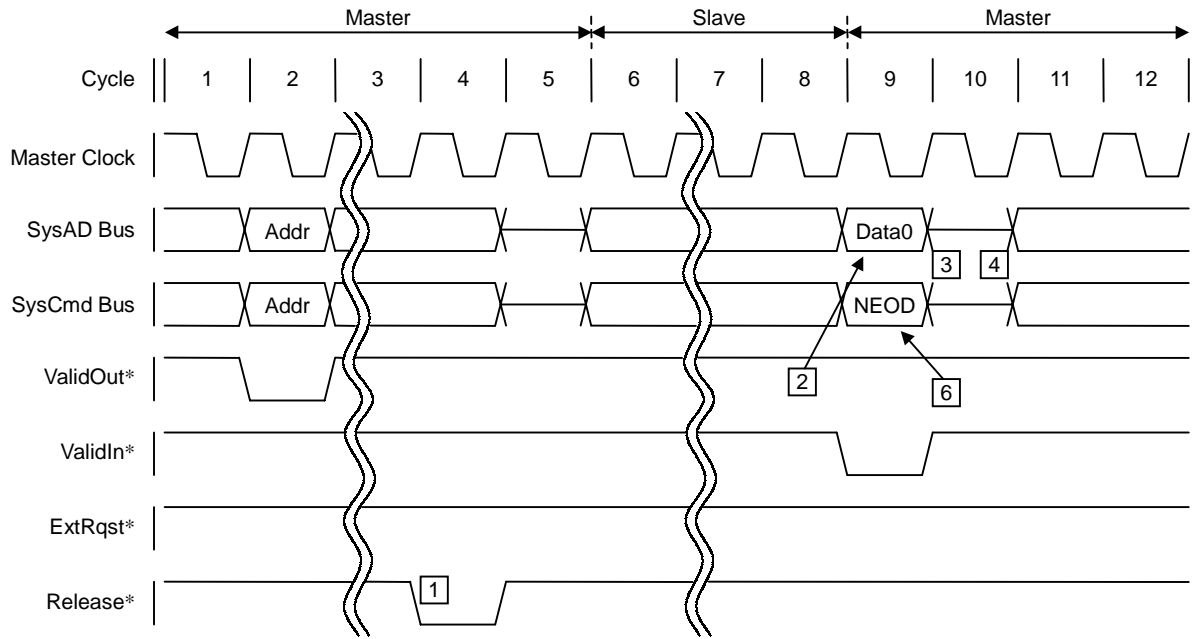
Figure 6.2.22 Processor Word Read Request and Subsequent Word Read Response
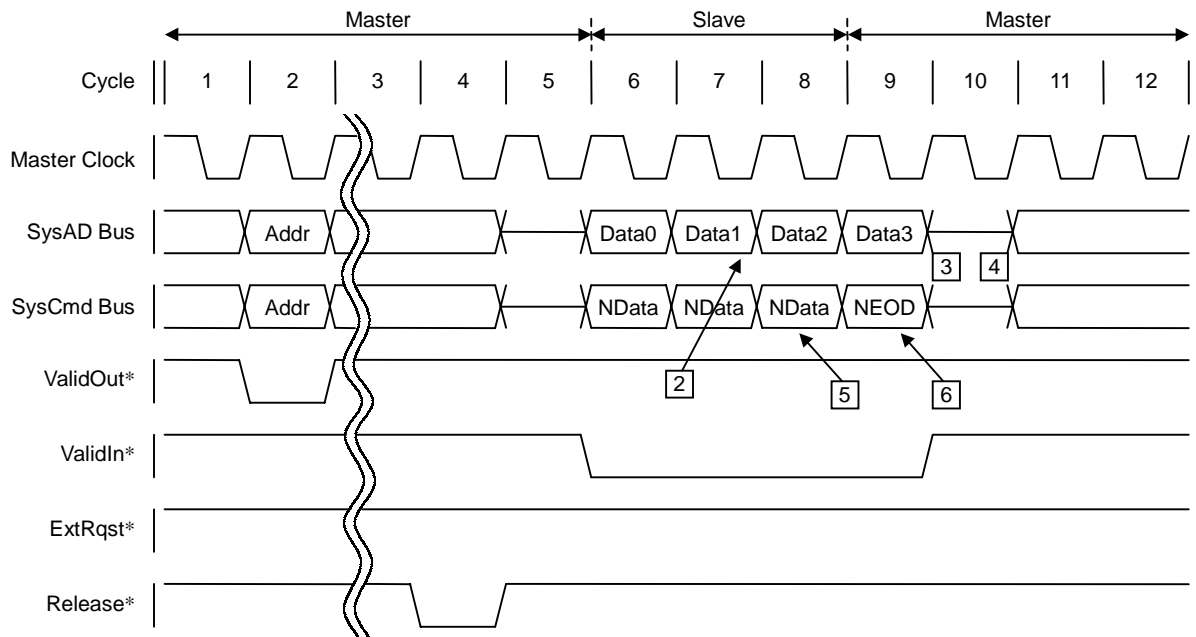
Figure 6.2.23 Block Read Response when System Interface is in the Slave State

6.2.9   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.

6.2.9.1   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 6.2.5 indicates the data transfer rate, data pattern, and setting.

Table 6.2.5 Data Transfer Rate, Data Pattern and Setting at Block write requests

Maximum Data Transfer Rate	Data Pattern	Setting Bit0
1 double word/3 Master Clock cycle	Dxx	0
1 double word/1 Master Clock cycle	D	1

Note: The setting Bit0 is bit0 of the G2Sconfig register (0xF FF10 0000).
      Bit0 is set to "0" when initialized.

6.2.9.2   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.

6.2.10   System Interface cycle time

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).

6.2.10.1 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 6.2.6 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.

Table 6.2.6 Release Latency for External Requests

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

6.2.11   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.

6.2.11.1 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."

6.2.11.2 Syntax of system interface commands

Following is an explanation of the SysCmd bus structure in the case of a system interface command. Figure 6.2.24 illustrates the structure common to all system interface commands.

8	7 5	4 0
0	Request type	Request specific

Figure 6.2.24 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 6.2.7 indicates the SysCmd(7:5) specification method.

Table 6.2.7 SysCmd(7:5) Specification Method for System Interface Commands

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

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

6.2.11.3 Read requests

Figure 6.2.25 illustrates the SysCmd format in the case of read requests.

8	7	5	4	0
0	000		Read request specific	
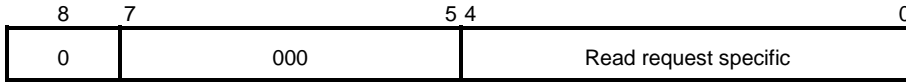
Figure 6.2.25 Bit Definition of SysCmd Bus for Read Requests

Table 6.2.8, 6.2.9, and 6.2.10 indicate the methods for specifying SysCmd(4:0) for read requests.

Table 6.2.8 Specification Method of SysCmd(4:3) for Read Requests

SysCmd(4:3)	Read Properties
0 ~ 1	Reserved
2	Non-coherent block read
3	Word or partial word read

Table 6.2.9 Specification Method of SysCmd(2:0) for Block Read Requests

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

Table 6.2.10 Data Size Expressed by SysCmd(2:0) for Double Word, Partial Double Word, Word or Partial Word 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: 6.10.4 of SysCmd(2:0) is only valid when in the 64-bit bus mode.

6.2.11.4 Write requests

Figure 6.2.26 illustrates the SysCmd format for write requests.

Table 6.2.11 indicates the methods of specifying write properties using SysCmd(4:3). Table 6.2.12 indicates the methods of specifying replacements properties using SysCmd(2:0) for block write requests. Table 6.2.13 indicates the methods of specifying the data size using SysCmd(2:0) for write requests.

8	7	5 4	0
0	010	Write request specific	
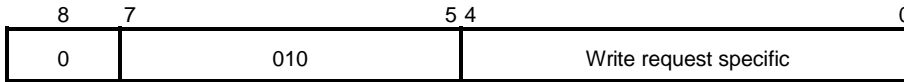
Figure 6.2.26 SysCmd Bus Bit Specification for Write Requests

Table 6.2.11 Methods of Specifying SysCmd(4:3) for Write Requests

SysCmd(4:3)	Write Properties
0 ~ 1	Reserved
2	Block write
3	Word or partial word write

Table 6.2.12 Specification Method of SysCmd(2:0) for Block Write Requests

SysCmd(2)	Reserved
SysCmd(1:0)	Write Block Size
0	Reserved
1	8 words
2 ~ 3	Reserved

Table 6.2.13 Methods for Specifying SysCmd(2:0) for Word or Partial Word 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	Reserved
5	Reserved
6	Reserved
7	Reserved

6.2.11.5 Null requests

Figure 6.2.27 illustrates the SysCmd format in the case of read requests.

8	7	5 4	0
0	011	Null request specific	
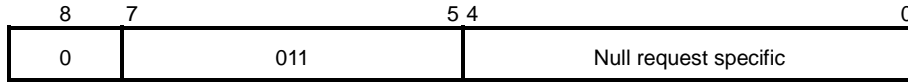
Figure 6.2.27 Bit Definition of the SysCmd Bus for Null Requests

Null request commands are always used for system interface release external null requests. Table 6.2.14 indicates methods of specifying SysCmd(4:3) for system interface release external null requests. SysCmd(2:0) is reserved for null requests.

Table 6.2.14 Method of Specifying SysCmd(4:3) for External Null Requests

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

6.2.11.6 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 6.2.28 is common to all system interface data identifiers.

8	7		5	4	3	0
0	Final Data	Response Data	Error Data	Data Check	Reserved	
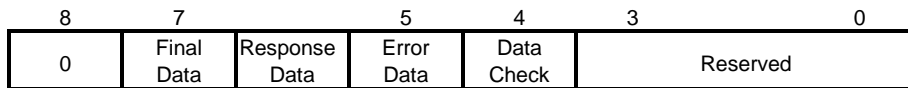
Figure 6.2.28 Bit Definition of the SysCmd Bus for Null Requests

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

6.2.11.7 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.

6.2.11.8 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 6.2.15 indicates methods of specifying SysCmd(7:3) for processor data identifiers. Table 6.2.16 indicates methods of specifying SysCmd(7:3) for external data identifiers.

Table 6.2.15 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(4:3)	Reserved

Table 6.2.16 Method 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)	Enable Data Check
0	Check data bits and check bits
1	Do not check data bits and check bits
SysCmd(3)	Reserved

6.2.12   System Interface Addresses

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

6.2.12.1 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.
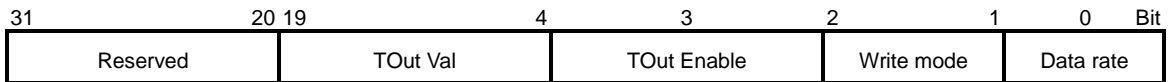
- Target addresses of the block request are aligned to the word boundaries. In other words, the lower 2 bits of the address become "0".

- 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.

6.2.13   Mode Register of System Interface (G2Sconfig)

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

Table 6.2.17 G2SConfig

Address	Field	Description
0xF_FF10_0000	G2Sconfig	Mode Register of System Interface

31	20 19	4 3	2	1 0	Bit
Reserved	TOut Val	TOut Enable	Write mode	Data rate	

Bit	Field	Description	ColdReset	Read/Write
31..20	—	Reserved	Undefined	Read
19..4	TOut Val	Set Data of Read Time Out Counter 0x0: Generate Read Time Out Error Other 0x0: Not generate Read Time Out Error	0xFFFF	Read/Write
3	TOut Enable	Enable bit of Read Time Out Counter In case of MODE43∗ = H, Read Time Out Counter does not work. We recommend it to write 0 or the value of Read data.	0	Read/Write
2..1	Write mode	Processor write protocols set 00: R4000 compatible write 01: Reserved 10: Pipeline write 11: Reissue write	00	Read/Write
0	Data rate	At the Block write data rate set 0: AWWWWWWWW 1: AWxxWxxWxxWxx WxxWxxWxxWxx	0	Read/Write

Note1: When initialized   Single write: ADxx (R4000 compatible)
                          Block write: ADDDD

Note2: In case of MODE43∗ = H, Read Time Out Counter does not work.
       It can generate Read Time Out Error when wrote in 0x0 at TOut Val. It does not let a value of TOut Enable affect it. Please write in a value except 0x0 at TOut Val.

Figure 6.2.29 G2Sconfig Register Formats

6.2.14   Data Error Detection

The TX4955A internal system interface uses the following two methods to detect data errors:

- Indication of good data by data identifier SysCmd(5)

- Determination by a check bit

6.2.14.1   Indication of good data by data identifier SysCmd(5)

This bit indicates whether data is good or not for all data. Therefore, in a block refill transfer, for example, no matter in what number of data SysCmd(5) may have been set, a bus error exception is always generated.

6.2.14.2   Determination by a check bit

When an error is detected by check bit determination, a bus error exception occurs. Because this determination also is made for all data, no matter in what number of data in a block refill transfer an error may have occurred, a bus error exception is always generated.

6.2.14.3   Timing at which a bus error exception occurs

Indication of good data (SysCmd(5)): Two cycles after read data

Check bit error detection: Three cycles after read response data

6.2.14.4   Precautions

- There is no means of identifying whether a bus error exception has been generated by indication of good data or by check bit determination. Nor does the TX49 core have a cache parity bit, but when a bus error exception occurs during a block refill transfer, the cache line is referenced INVALID.

- Regardless of whether a bus error exception has been generated by indication of good data or by check bit determination during a block refill transfer, a designated block size of data needs to be transferred.

6.3   System Interface of TX4300 type protocol mode

In TX4955A, it is built in system interface function corresponding to TX4300 type protocol. A selection of above-mentioned R5000 type protocol mode or TX4300 type protocol mode increases by external pin (MODE43* : 117 pin).

MODE43* = 0: TX4300 type protocol

MODE43* = 1: R5000 type protocol

Note: In TX4300 type protocol mode of TX4955A, there is not PReq signal. Therefore there is the need that PReq terminal of an external agent is always fixed in "L" Level. And there is not a function too to show that protocol errors were detected.

But search of protocol errors is possible when I have a counter built-in in TX4955A inside, and there is not lead reply between constant time because it is established the function that an exception generates bus error.

6.3.1   System Interface Description of TX4300 Type Protocol Mode

The TX4955A processor has a 32-bit address/data interface. The System interface consists of:

- 32-bit address and data bus, **SysAD**
- 5-bit command bus, **SysCmd**
- five handshake signals:
  - **EValid***, **PValid***
  - **EReq***
  - **PMaster***, **EOK***

Table 6.3.1 System Interface Signals

(TX4300 type protocol mode is MODE43* = 0)

Signal	I/O	Function
SysAD (31:0)	I/O	Address and data transfer bus, multiplexed between the processor and an external agent.
SysCmd (4:0)	I/O	Used for command and data identifier transmission between the processor and an external agent.
SysCmd (8:5) / GND	O	When TX4300 type protocol mode, those pin output "L".
SysADC (3:0) / GND	O	When TX4300 type protocol mode, those pin output "L".
SysCmdP / GND	O	When TX4300 type protocol mode, this pin output "L".
ValidIn* / EValid*	I	During the cycle it is asserted, EValid* indicates an external agent is driving a valid address or valid data on the SysAD bus, and a valid command or data identifier on the SysCmd bus.
ValidOut* / PValid*	O	During the cycle it is asserted, PValid*, indicates the processor is driving a Valid address or Valid data on the SysAD bus, and a valid command or data indentifier on the SysCmd bus.
ExtRqst* / Ereq*	I	Indicates an external agent is requesting System interface bus ownership.
Release* / PMaster	O	Indicates an external agent is capable of the system interface bus.
WrRdy* / EOK*	I	Indicates an external agent is capable of accepting a processor request.
RdRdy* / GND	O	When TX4300 type protocol mode this pin output "L".

Table 6.3.2 Pin Assign

(MODE43* = 0)

#	Signal	#	Signal	#	Signal	#	Signal
1	Vss	41	Vss	81	VccInt	121	SysAD28
2	BufSel1	42	TRST*	82	NMI*	122	SysAD29
3	JTDO	43	RdRdy* / GND	83	ExtRqst* / Ereq*	123	VccInt
4	JTDI	44	WrRdy* / EOK*	84	Reset*	124	Vss
5	JTCK	45	ValidIn* / Evalid*	85	ColdReset*	125	SysAD30
6	JTMS	46	ValidOut* / Pvalid*	86	VccIO	126	VccIO
7	VccIO	47	Release* / PMaster*	87	Endian	127	Vss
8	Vss	48	VccIO	88	VccIO	128	SysAD31
9	SysAD4	49	PLLReset*	89	Vss	129	SysADC2 / GND
10	SysAD5	50	VccInt	90	SysAD16	130	VccInt
11	VccInt	51	TintDis	91	VccInt	131	Vss
12	Vss	52	Vss	92	Vss	132	SysADC3 / GND
13	SysAD6	53	SysCmd0	93	SysAD17	133	VccIO
14	VccIO	54	SysCmd1	94	SysAD18	134	Vss
15	Vss	55	SysCmd2	95	VccIO	135	SysADC0 / GND
16	SysAD7	56	SysCmd3	96	Vss	136	VccInt
17	SysAD8	57	SysCmd4	97	SysAD19	137	Vss
18	VccInt	58	SysCmd5 / GND	98	VccInt	138	SysADC1 / GND
19	Vss	59	VccIO	99	Vss	139	SysAD0
20	SysAD9	60	Vss	100	SysAD20	140	VccIO
21	VccIO	61	SysCmd6 / GND	101	SysAD21	141	Vss
22	Vss	62	SysCmd7 / GND	102	VccIO	142	SysAD1
23	SysAD10	63	SysCmd8 / GND	103	Vss	143	SysAD2
24	SysAD11	64	SysCmdP / GND	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	BufSel0	75	VccIO	115	SysAD27	155	VssPLL
36	PCST3	76	Vss	116	VccIO	156	PLLCAP
37	PCST2	77	TPC3	117	MODE43*	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.

    MODE43* = 0: TX4300 type

             = 1: R5000 type

    GND means out to Low level when MODE43* = 0

6.3.2   System Events

System events include:

- Fetch miss in the instruction cache
- Load miss in the data cache
- Store miss in the data cache
- an uncached load or store
- actions resulting from the execution of cache instructions

When a system event occurs, the processor issues a request or a series of requests through the system interface to access some external resource to service that event. The system interface must be connected to an external agent that coordinates access to system resources.

Processor requests include both read and write requests:

- a read request supplies an address to an external agent
- a write request supplies an address and a word or block of data to be written to an external agent

Processor read requests that have been issued, but for which data has not yet been returned, are said to be *pending*. The processor will not issue another request while a read is already pending. A processor read request is said to be *complete* after the last transfer of response data has been received from an external agent. A processor write request is said to be complete after the last word of data has been transmitted.

External requests include both read responses and write requests:

- a read response supplies a block or single transfer of data from an external agent in response to a read request
- a write request supplies an address and a word of data to be written to a processor resource

When an external agent receives a read request, it accesses the specified resource and returns the requested data through a read response, which may be returned any time after the read request and at any data rate.

By default, the processor is the master of the system interface. An external agent becomes master of the system interface either through arbitration, or by default after a processor read request. The external agent returns mastership to the processor after the external request completes and/or after the processor read request has been serviced.

6.3.3   System Event Sequences and the SysAD Bus Protocol

The following sections detail the sequence and timing of processor and external requests.

Note: The following sections describe the SysAD bus *protocol*; the TX4955A processor always meets the conditions of this protocol. The TX4955A processor is capable of receiving sequences of transactions on the bus at full protocol speed and of receiving data on every cycle. At a minimum, the design of external agents must meet the requirements of this protocol, and would ideally take full advantage of the maximum speed of the TX4955A processor.

6.3.3.1   Fetch Miss

When the processor misses in the instruction cache on a fetch, it obtains a cache line of instructions from an external agent. The processor issues a read request for the cache line and waits for an external agent to provide the data in response to this read request.

6.3.3.2 Load Miss

When the processor misses in the data cache on a load, it obtains a cache line of data from an external agent. The processor issues a read request for the cache line and waits for an external agent to provide the data in response to this read request. If the cache data which the incoming line will replace contains valid dirty data, this data is written to memory. The read completes before the write of the dirty cast-out data.

6.3.3.3 Store Miss

When the processor misses in the data cache on a store, it issues a read request to bring a cache line of data into the cache, where it is then updated with the store data. If the cache data which the incoming line will replace contains valid dirty data, the data is written to memory. The read completes before the write of the dirty cast-out data.

To guarantee that cached data written by a store is consistent with main memory, the corresponding cache line must be explicitly flushed from the cache using a cache operation.

6.3.3.4 Uncached Load or Store

When the processor performs an uncached load, it issues a read request and waits for a single transfer of read response data from an external agent.

When the processor performs an uncached store, it issues a write request and provides a single transfer of data to the external agent.

The processor does not consolidate data on uncached writes. For example, writes of two contiguous halfwords takes two write cycles, they are never grouped into a single word write.

6.3.3.5 Cache Instructions

The TX4955A processor provides a number of cache instructions for use in maintaining the state and contents of the caches. Cache operations supported in the TX4955A processor are described in Chapter 5.

6.3.3.6 Byte Ordering (Endian)

The System interface byte order is set by the **Endian** of external pin. The byte order is big-endian when **Endian** is high, and little-endian when **Endian** is low. The *RE* (reverse-endian) bit in the *Status* register can be set by software to reverse the byte order available in User mode.

6.3.3.7 Physical Addresses

Physical addresses are driven on **SysAD (31:0)** during address cycles.

6.3.3.8 Interface Buses

Figure 6.3.1 shows the primary communication paths for the System interface: a 32-bit address and data bus, **SysAD (31:0)**, and a 5-bit command bus, **SysCmd (4:0)**. These **SysAD** and the **SysCmd** buses are bidirectional; that is, they are driven by the processor to issue a processor request, and by the external agent to issue an external request.

A request through the System interface consists of:

- an address
- a System interface command that specifies the precise nature of the request
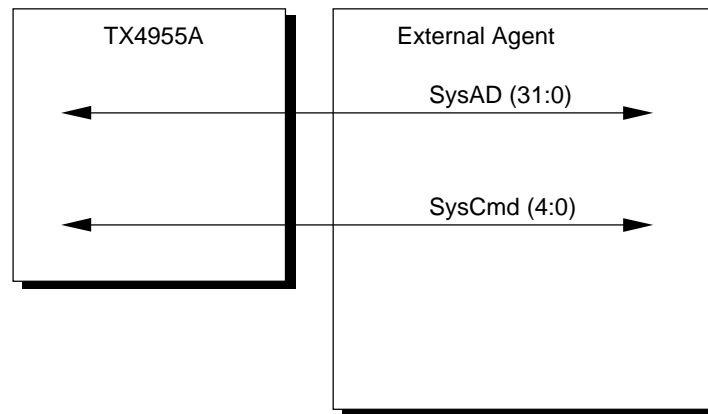- a series of data elements if the request is for a write or read response.

Figure 6.3.1 System Interface Buses

6.3.3.9   Address and Data Cycles

The **SysCmd** bus identifies the contents of the **SysAD** bus during any cycle in which it is valid. Cycles in which the **SysAD** bus contains a valid address are called *address cycles*. Cycles in which the **SysAD** bus contains valid data are called *data cycles*. The most significant bit of the **SysCmd** bus is always used to indicate whether the current cycle is an address cycle or a data cycle.

When the TX4955A processor is driving the **SysAD** and **SysCmd** buses, the System interface is in *master state*. When the external agent is driving the **SysAD** and **SysCmd** buses, the System interface is in *slave state*.

- When the processor is master, it asserts the **PValid*** signal when the **SysAD** and **SysCmd** buses are valid.

- When the processor is slave, an external agent asserts the **EValid*** signal when the **SysAD** and **SysCmd** buses are valid.

The **SysCmd** bus identifies the contents of the **SysAD** bus during valid cycles.

- During address cycles [**SysCmd (4)** = 0], the remainder of the **SysCmd** bus, **SysCmd (3:0)**, contains a *System interface command*, described later in this chapter.

- During data cycles [**SysCmd (4)** = 1], the remainder of the **SysCmd** bus, **SysCmd (3:0)**, contains a *data identifier*, described later in this chapter.

6.3.4   System Interface Protocols

Figure 6.3.2 shows the register-to-register operation of the System interface. That is, processor outputs come directly from output registers and begin to change with the rising edge of **SClock**.†

Processor inputs are fed directly to input registers that latch these input signals with the rising edge of **SClock**. This allows the System interface to run at the highest possible clock frequency.
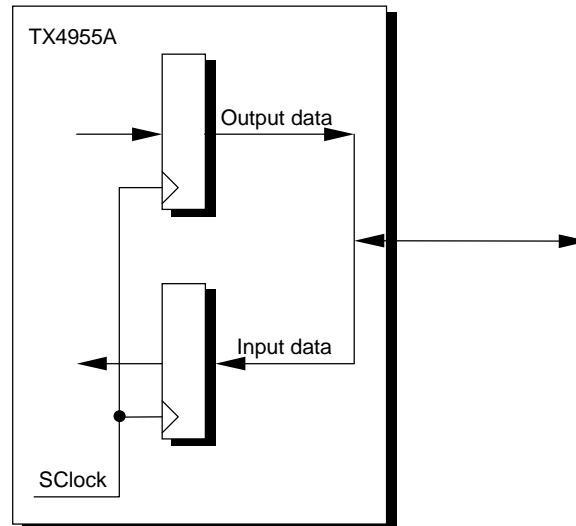

Figure 6.3.2 System Interface Register-to-Register Operation

6.3.4.1   Master and Slave States

When the TX4955A processor is driving the **SysAD** and **SysCmd** buses, the System interface is in *master state*. When the external agent is driving the **SysAD** and **SysCmd** buses, the System interface is in *slave state*.

In master state, the processor asserts the signal **PValid*** whenever the **SysAD** and **SysCmd** buses are valid.

In slave state, the external agent asserts the signal **EValid*** whenever the **SysAD** and **SysCmd** buses are valid.

6.3.4.2   Moving from Master to Slave State

The processor is the default master of the system interface. An external agent becomes master of the system interface through arbitration, or by default after a processor read request. The external agent returns mastership to the processor after an external request completes.

The System interface remains in master state unless one of the following occurs:

- The external agent requests and is granted the System interface (external arbitration).

- The processor issues a read request (uncompelled change to slave state).

The following sections describe these two actions.

---

† **SClock** is an internal clock used by the processor to sample data at the System interface and to clock data into the processor System interface output registers.

6.3.4.3   External Arbitration

The System interface must be in slave state for the external agent to issue an external request through the System interface. The transition from master state to slave state is arbitrated by the processor using the System interface handshake signals **EReq*** and **PMaster***. This transition is described by the following procedure:

1.   An external agent signals that it wishes to issue an external request by asserting **EReq***.

2.   When the processor is ready to accept an external request, it releases the System interface from master to slave state by negating **PMaster***.

3.   The System interface returns to master state as soon as the issue of the external request is complete.

6.3.4.4   Uncompelled Change to Slave State

An *uncompelled* change to slave state is the transition of the System interface from master state to slave state, initiated by the processor itself when a processor read request is pending. **PMaster*** is negated automatically after a read request. An uncompelled change to slave state occurs either during or some number of cycles after the issue cycle of a read request.

The uncompelled release latency depends on the state of the cache. After an uncompelled change to slave state, the processor returns to master state at the end of the next external request. This can be a read response, or some other type of external request.

An external agent must note that the processor has performed an uncompelled change to slave state and begin driving the **SysAD** bus along with the **SysCmd** bus. As long as the System interface is in slave state, the external agent can begin an external request without arbitrating for the System interface; that is, without asserting **EReq***.

After the external request, the System interface returns to master state.

6.3.4.5   Signal Timing

The System interface protocol describes the cycle-by-cycle signal transitions that occur on the pins of the system interface to realize requests between the processor and an external agent. Figure 6.3.3 shows the timing relationships between System interface signal edges.
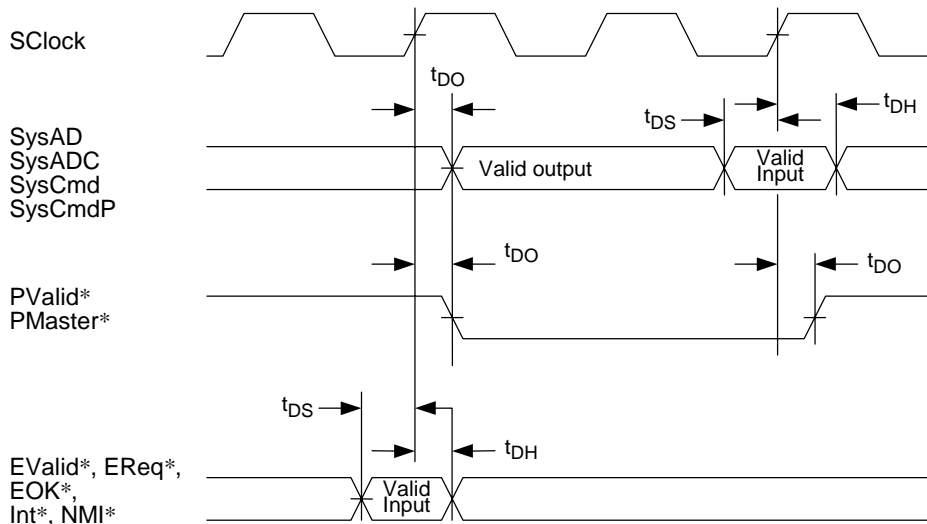
Note: These waveforms only describe edge-to-edge timing relationships.

Figure 6.3.3 System Interface Edge Timing Relationships

The Timing Summary section below describes the minimum and maximum timing values of each signal. The sections that follow describe the timing requirements for various bus cycles.

6.3.5   Timing Summary

In the following timing diagrams, gray-scale signals indicate values that are either Unknown or Don't Cares, within the specification limits. They may be any value as long they do not violate any bus value or timing specification. The timing diagrams illustrate cycles using the following signals:

- **PMaster***
- **EValid***
- **PValid***
- **EOK***
- **EReq***

PMaster*   (O)   Indicates the processor is the master of the system interface bus.
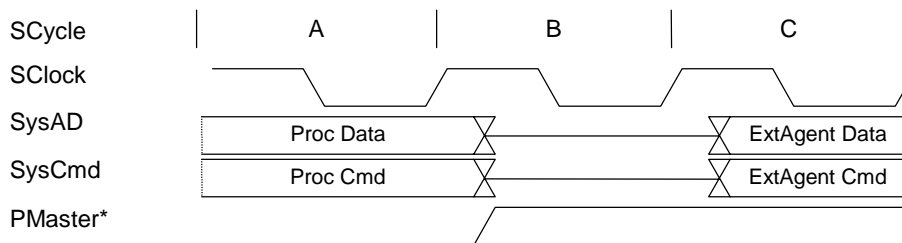
Figure 6.3.4 Sample Cycle with ***PMaster*** Asserted, Then Deasserted

A   Processor drives SysAD and SysCmd buses (processor is master).

B   **PMaster** is deasserted. SysAD and SysCmd buses are set to a tri-state(no bus master).

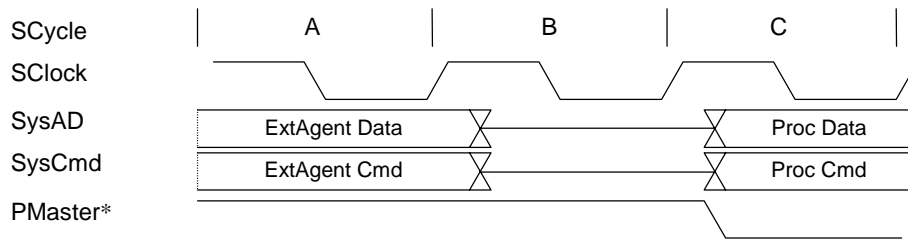C   External agent drives SysAD and SysCmd buses (external agent is master).

SCycle
SClock
SysAD
SysCmd
PMaster*

Figure 6.3.5 Sample Cycle with **PMaster*** Asserted

A   External agent drives SysAD and SysCmd buses (external agent is master).

B   SysAD and SysCmd buses are set to a tri-state (no bus master).

C   **PMaster*** is asserted. Processor drives SysAD and SysCmd buses (processor is master).

EValid* **(I), PValid*** **(O)**   During a cycle in which either signal is asserted, the signal indicates a new valid address or valid data is on the SysAD bus, and a new valid command or data identifier is on the SysCmd bus. **EValid*** indicates an external agent is driving new SysAD and SysCmd values. **PValid*** indicates the processor is driving new SysAD and SysCmd values.
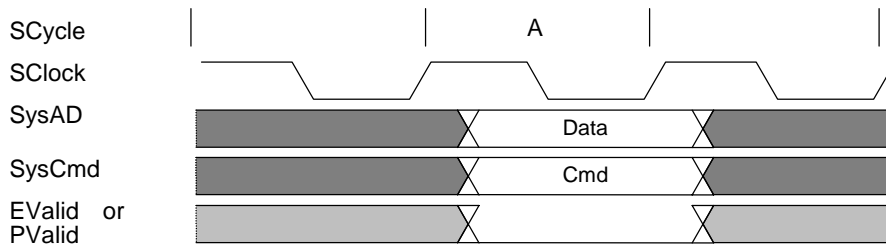
SCycle
SClock
SysAD
SysCmd
EValid or
PValid

Figure 6.3.6 Sample Cycle with **PValid*** and **EValid***

A: New SysAD and SysCmd values.

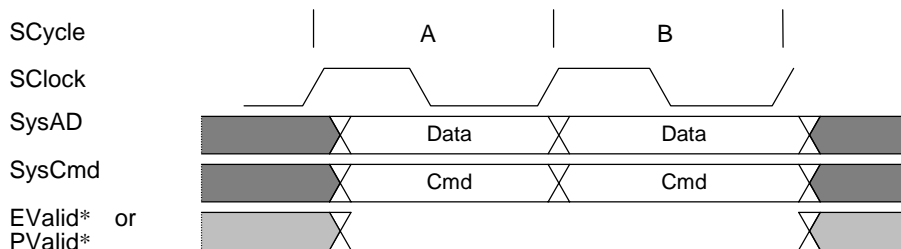   Each cycle either of these signals remains asserted indicates there is a new SysAD and SysCmd value.

SCycle
SClock
SysAD
SysCmd
EValid*   or
PValid*

Figure 6.3.7 Sample Cycles with Multiple **PValid*** and **EValid***

A   New SysAD and SysCmd value.

B   Another new SysAd and SysCmd value.

EOK*   **(I)**   Indicates an external agent accepts a processor request. An external agent has excepted the processor read/write command if and only if the following has occurred:

A   **EOK*** is active.

B   The processor asserts **PValid*** and drives a read or write command. **EOK*** is asserted and the external agent accepts the processor command.

Once the external agent has accepted a processor write command, the agent must be able to accept the entire data size at the programmed data rate immediately following this command.

The external agent may provide read response data to the processor at any rate.

Deasserting EOK may kill a processor read/write request in progress. If this occurs, the external agent must ignore command and data from the processor in the following cycle.
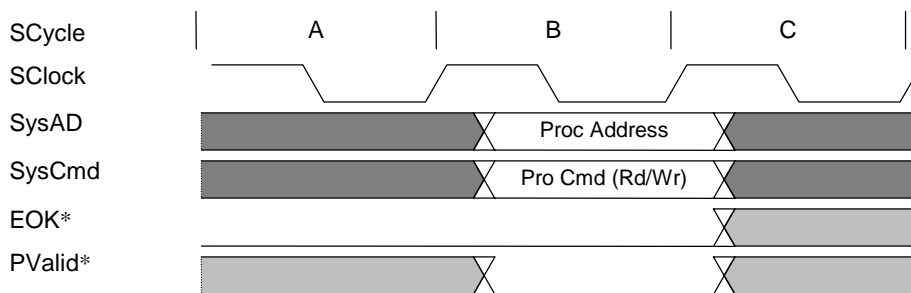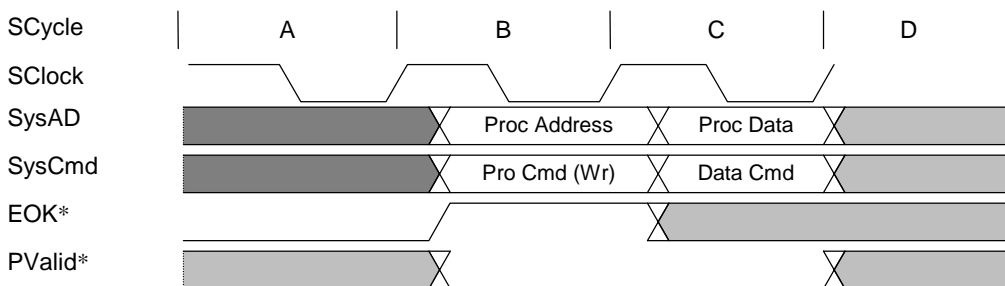
Figure 6.3.8 Sample Cycle with **EOK*** Asserted

Figure 6.3.9 Sample Cycle with **EOK*** Asserted, Then Deasserted

A   **EOK*** is active.

B   Processor asserts **PValid*** and drives a read or write command. **EOK*** is deasserted (external agent has killed the processor's command).

C   The external agent must ignore any SysAD and SysCmd data from the processor.

D   The external agent *does not* ignore any SysAD and SysCmd data from the processor.

EReq*   **(I)**   Indicates an external agent is requesting bus ownership of the System interface. To gain mastership of the bus, an external agent must arbitrate with the processor as follows:

A   External agent asserts EReq*

B   Wait for PMaster* to be deasserted (1 to N cycles).

C   External agent drives SysAD and SysCmd buses. The external agent is guaranteed to maintain mastership of the bus as long as **EReq** is asserted.

If at any time **EReq*** is deasserted, the external agent must go back to step A and re-arbitrate for the bus.
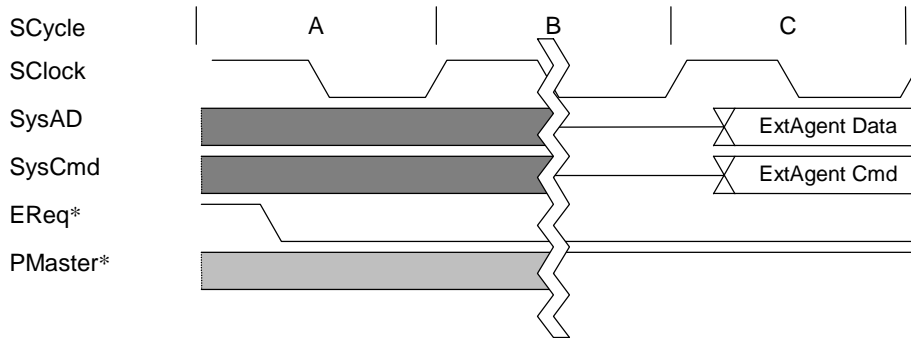

Figure 6.3.10 Sample Cycle with ***EReq**** Asserted

From the time that **EReq*** is asserted, the external agent is guaranteed to gain mastership of the bus after at most one processor request. However, if **EOK*** is being deasserted, the external agent will gain mastership of the bus without having to accept any processor requests.

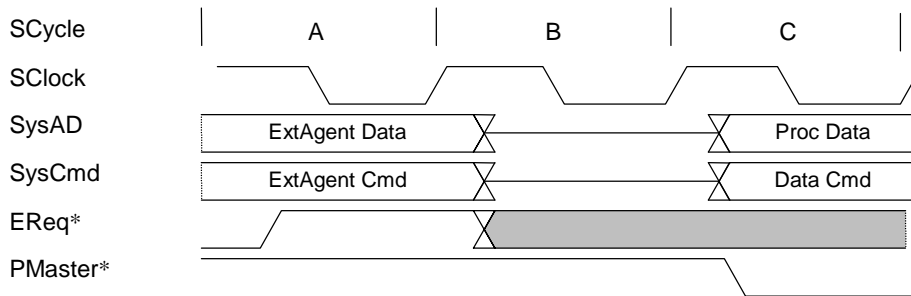The external agent relinquishes bus mastership by deasserting **EReq*** as shown below:

Figure 6.3.11 Sample Cycle with Deassertion of ***EReq****

A   External agent deasserts **EReq*** and external agent drives the bus.

B   Bus is set to a tristate.

C   Processor regains mastership of bus.

Except for a processor read request (see below), assertion of **EReq*** is the only way the external agent gets and maintains bus mastership.
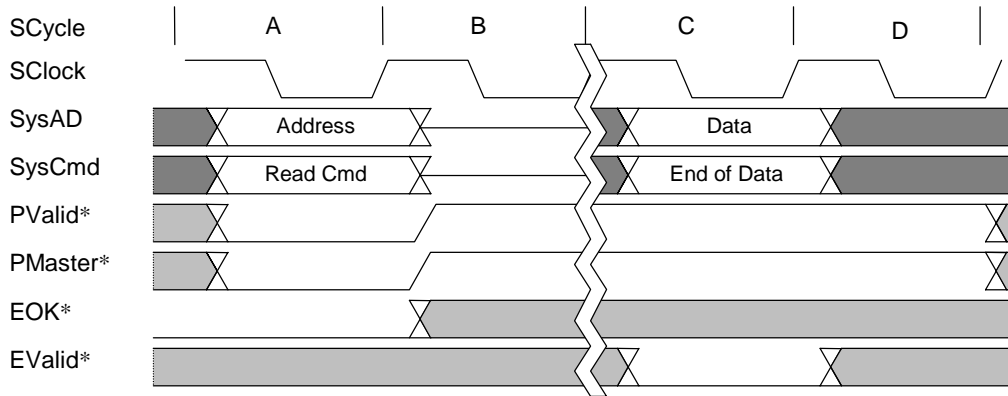
Figure 6.3.12 Sample Cycle with Assertion of **EReq**

A   Processor drives a valid read command and an external agent accepts it.

B   **PMaster*** is deasserted and the bus is set to a tristate.

C   External agent drives last of requested data. During all cycles between B and C the external agent is guaranteed mastership of the bus.

6.3.6   Arbitration

The processor is the default master of the bus. It relinquishes ownership of the bus either when an external agent requests and is granted the system interface, or until the processor issues a read request. The transition from processor master to processor slave state is arbitrated by the processor, using the System interface handshake signals **EReq*** and **PMaster***.

When a processor read request is pending, the processor transitions to slave state by deasserting **PMaster***, allowing an external agent to return the read response data. The processor remains in slave state until the external agent issues an *End Of Data* read response, whereupon the processor reassumes mastership, signalled by the assertion of **PMaster***. Note that an external agent is able to retain mastership of the bus after an End Of Data read response if the external agent arbitrates for mastership using **EReq***.

When the processor is master, an external agent acquires control of the system interface by asserting **EReq***, and waiting for the processor to deassert **PMaster***. The processor is ready to enter slave state when it deasserts **PMaster***. The external agent must go through a three-step arbitration process (see the **EReq*** cycle in the Timing Summary) before driving the bus. Once the external agent has become master through **EReq*** arbitration, it can remain master as long as it continues to assert **EReq***. The System interface returns to master state (with the processor driving the bus) two cycles after **EReq*** is deasserted. Figure 6.3.13 illustrates an arbitration for external requests.
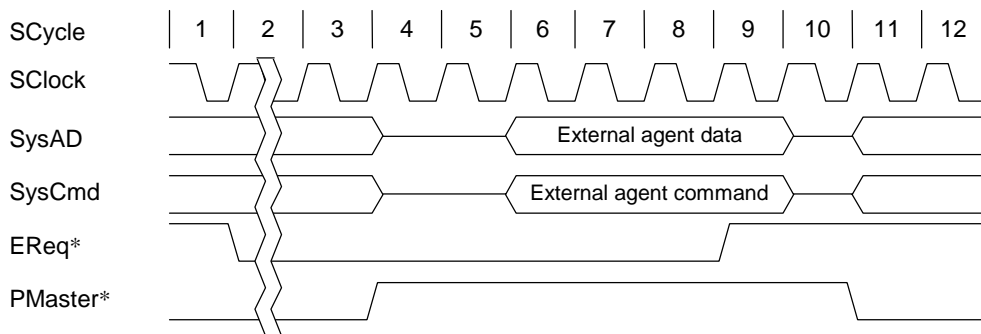

Figure 6.3.13 External Request Arbitration

When an external agent is master, it may always respond to a read with data. If the external agent has become master by **EReq***, it may issue transactions at will; that is, the processor must always accept any command or data on the bus at any time. There is no means for the processor to hold off the external agent once the external agent is master.

If the processor is in slave state and needs the bus,wait untile the external agent EReq* deasserted. Thereafter, when the processor sees **EReq*** deasserted, it resumes bus ownership, asserts the **PMaster*** line, and issues its own command. The processor becomes master and drives the bus two cycles after **EReq*** is deasserted.

An illustration of a processor request for bus mastership and the release of the bus by the external agent is illustrated in Figure 6.3.14.
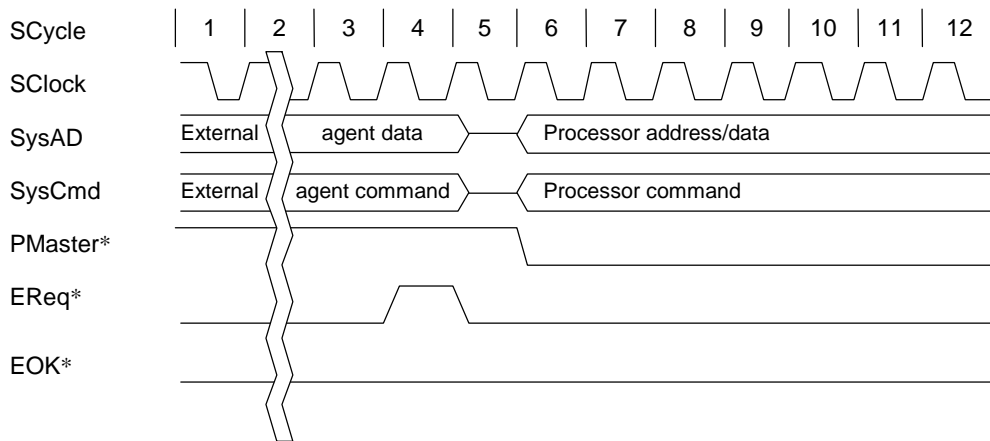
Figure 6.3.14 Processor Request For Bus Arbitration And External Agent Release

Upon assertion of **Reset*** or **ColdReset***, the processor becomes bus master and the external agent must become slave.

This protocol guarantees that either the processor or an external agent is always bus master. The master should never tristate the bus, except when giving up ownership of the bus under the rules of the protocol.

6.3.7   Issuing Commands

When the processor is master of the bus and wishes to issue a command, it cannot successfully issue the command until the external agent signals that it is ready to accept it. This readiness is indicated by assertion of the **EOK*** signal. Being master, the processor may place the command on the bus and continually reissue it while waiting for **EOK*** to be asserted; however, the command is not considered issued until **EOK*** has been asserted for two consecutive cycle (see the Timing Summary for **EOK*** earlier in this chapter).

If the **EOK*** signal is asserted in one cycle and then deasserted in the next, during which time a command is issued, that command is considered killed and must be retried. When a command is killed in this way, the processor begins to execute the read/write command. This action must be ignored by the external agent. If a write command is killed, the data cycle following this killed transaction must be ignored. If a read is killed, the processor releases the bus one cycle after and (assuming no **EReq***) regains mastership two cycles later. This allows the processor to retry the transaction.

6.3.8   Processor Write Request

A processor write request is issued by the following:

- driving a write command on the SysCmd bus
- driving a write address on the SysAD bus
- asserting **PValid*** for one cycle
- driving the appropriate number of data identifiers on the SysCmd bus
- driving data on the SysAD bus
- asserting **PValid***

For 1-to 4-byte writes, a single data cycle is used. 5-, 6- and 7-byte writes are broken up into two address/data transactions; one 4 bytes in size, the next handling the remaining 1, 2, or 3 bytes.

For all transactions larger than 7 bytes (e.g. 8, 16, 32), 4 bytes are sent on each data cycle until the appropriate number of bytes has been transferred. The final data cycle is tagged as end of data (EOD) on the command bus.

To be fully compliant with all implementations of this protocol, an external agent should be able to receive write data over any number of cycles with any number of idle cycles between any two data cycles. However, for the TX4955A processor implementation, data begins to arrive on the cycle immediately following the write issue cycle, and continues to arrive at a programed data rate thereafter. The processor drives data at the rate specified by the data rate configuration signals (see the section describing Data Rate Control, later in this chapter).

Writes may be cancelled and retried with the **EOK**∗ signal (see the section earlier, Issuing Commands).

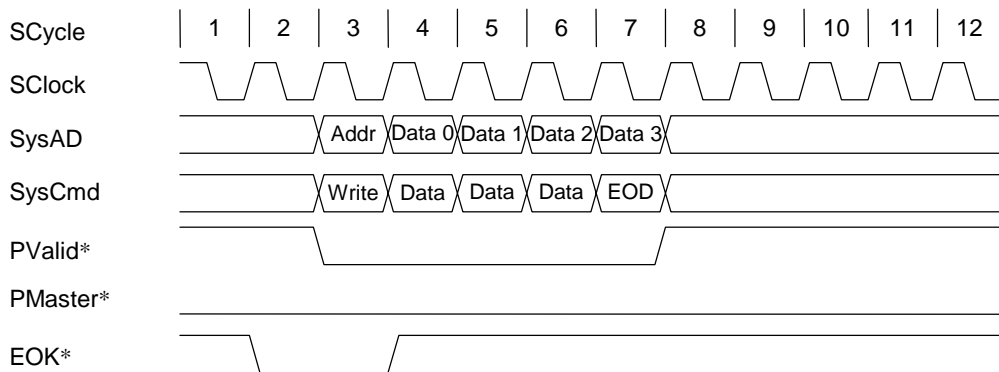Figure 6.3.15 illustrates the bus transactions for a 4-word data cache block store.

Figure 6.3.15 Processor Block Write Request With D Data Rate

Figure 6.3.16 illustrates a write request which is cancelled by the deassertion of **EOK**∗ during the address cycle of the second write, and which is retried when **EOK**∗ is asserted again.
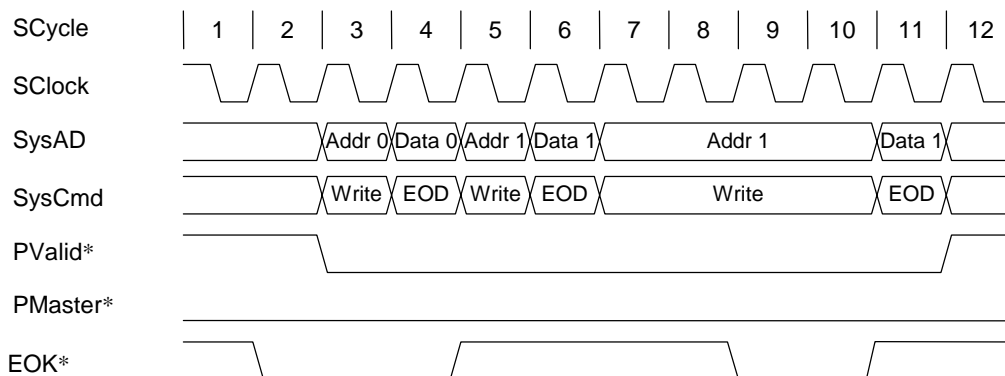

Figure 6.3.16 Processor Single Write Request Followed By A Cancelled And Retried Write Request

6.3.9   Processor Read Request

A processor read request is issued by the following:

- driving a read command on the SysCmd bus
- driving a read address on the SysAD bus
- asserting **PValid***

Only one processor read request may be pending at a time. The processor must wait for an external read response before starting a subsequent read.

The processor moves to slave state after the issue cycle of the read request, by deasserting the **PMaster*** signal. An external agent may then return the requested data through a read response. The external agent, which is now bus master, may issue any number of writes before sending the read response data.

An example of a processor read request and an uncompelled change to slave state occurring as the read request is issued is illustrated in Figure 6.3.17.
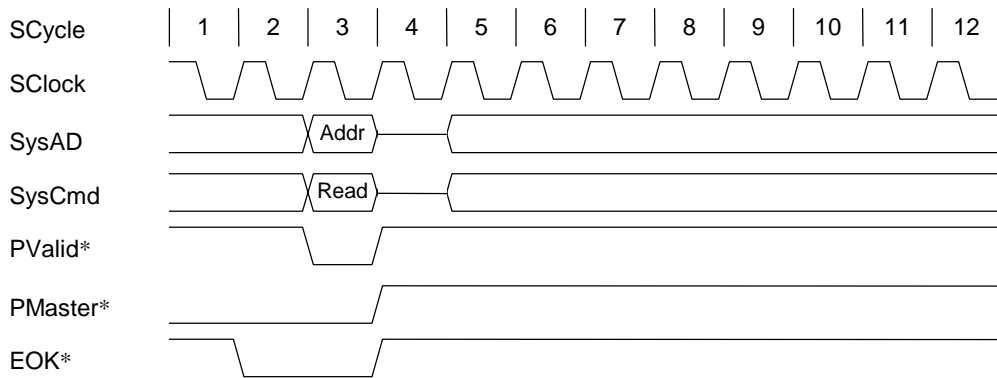

Figure 6.3.17 Processor Read Request

The TX4955A support the Read Time Out Function. This Function is when the retern data wait but the time out counter. See chapter 6.3.20 Mode Register of System Interface (G2SConfig).

6.3.10   External Write Request

External write requests are similar to a processor single write except that the signal EValid is asserted instead of the signal **PValid***. An external write request consists of the following:

- an external agent driving a write command on the SysCmd bus and a write address on the SysAD bus
- asserting **EValid*** for one cycle
- driving a data identifier on the SysCmd bus and data on the SysAD bus
- asserting **EValid*** for one cycle.

The data identifier associated with the data cycle must contain a last data cycle indication. Note that the external agent must gain and maintain bus mastership during these transactions (see **EReq*** in the Timing Summary, earlier in this chapter).

An external write request example with the processor initially in master state is illustrated in Figure 6.3.18.
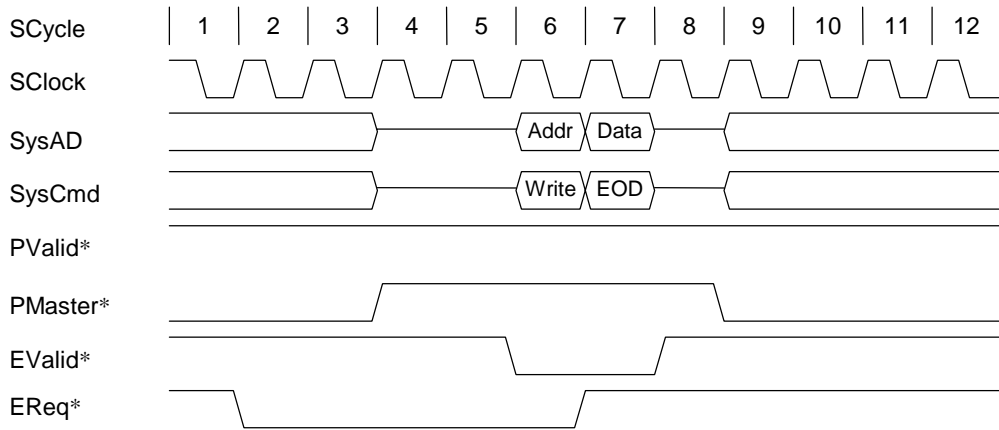
Figure 6.3.18 External Write Request

An example of a read response for a processor single word read request that is interrupted by an external agent write request is illustrated in Figure 6.3.22. External writes can not occur in the middle of a data response block; they can, however, occur before the first data response of the data block or after the last EOD response, but it can not occur between them.

Note: The only writable resources are processor interrupts. An external write to any address is treated as a write to the processor interrupts.

6.3.11 External Read Response

An external agent returns data to the processor in response to a processor read request by waiting for the processor to move to slave state. The external agent then returns the data through either a single data cycle or a series of data cycles sufficient to transmit the requested data. After the last data cycle is issued, the read response is complete and the processor becomes master (assuming **EReq*** was not asserted).

If, at the end of the read response cycles, **EReq*** has been asserted, the processor remains in slave state until the external agent relinquishes the bus. When the processor is in slave state and needs access to the SysAD bus, it waits until **EReq*** is deasserted.

The data identifier associated with a data cycle may indicate that the data transmitted during that cycle is erroneous; however, an external agent must return a block of data of the correct size regardless of this erroneous data cycle indication. If a read response includes one or more erroneous data cycles, the processor takes a bus error.

Read response data must only be delivered to the processor when a processor read request is pending. The state of the processor is undefined if a read response is presented to it when no processor read is pending.

An example of a processor single read request followed by a read response is illustrated in Figure 6.3.19.
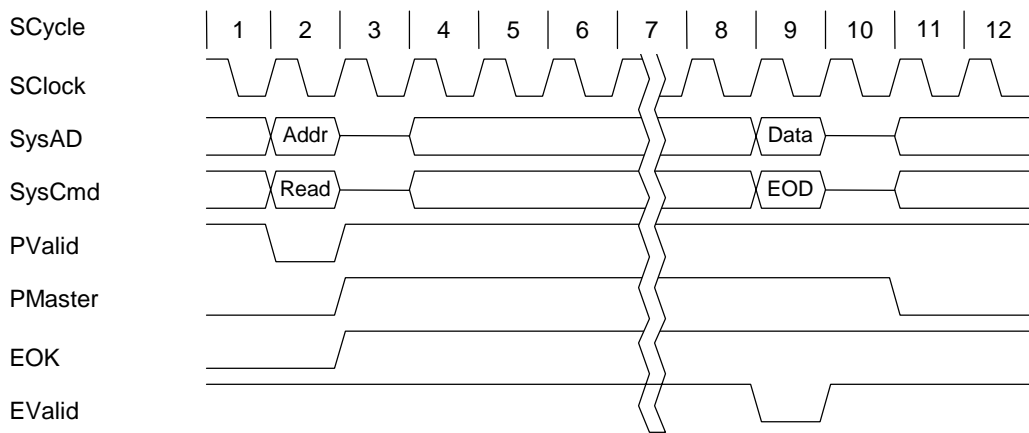

Figure 6.3.19   Single Read Request Followed By Read Response

A read response example for a processor block read with the system interface already in slave state is illustrated in Figure 6.3.20.
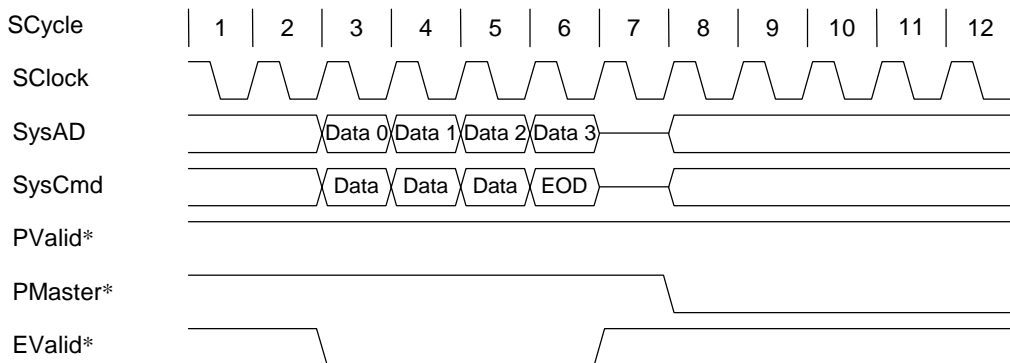

Figure 6.3.20 Block Read Response, System Interface Already In Slave State

A read response example for a processor single read request followed by an external agent write request is illustrated in Figure 6.3.21.
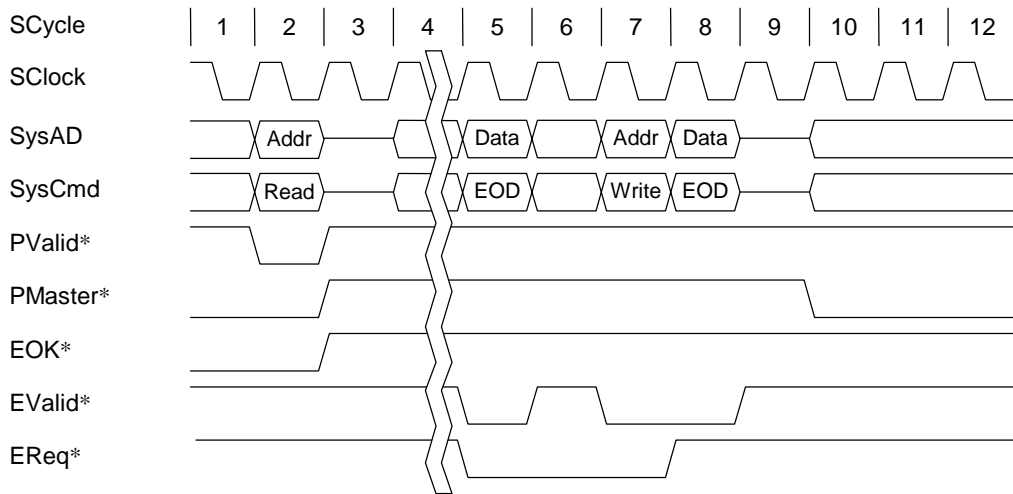

Figure 6.3.21 Single Read Request Followed By External Write Request

(External Agent Keeps Bus)

An example of a read response for a processor single word read request that is interrupted by an external agent write request is illustrated in Figure 6.3.22. Cycle 5 is the data for the external write request in cycle 4. Cycle 7 is the read response data.
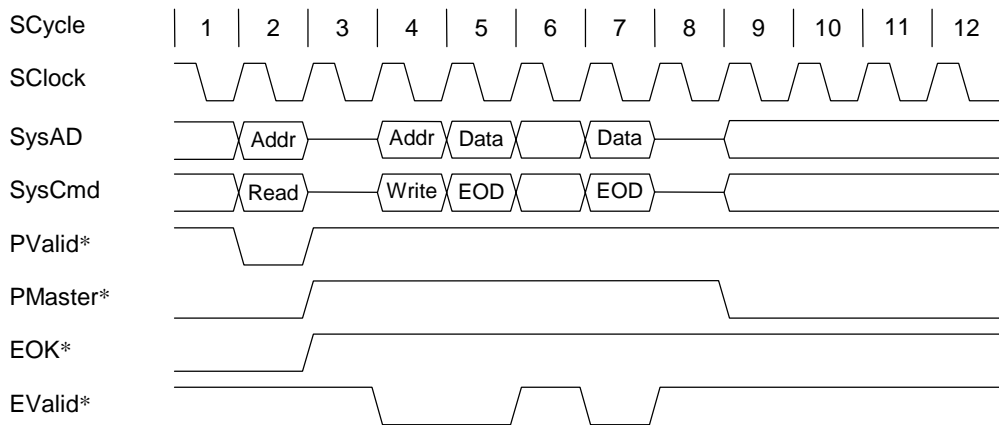

Figure 6.3.22 External Write Followed By External Read Response, System Interface In Slave State

6.3.12 Flow Control

EOK* may be used by an external agent to control the flow of processor read and write requests; while **EOK*** is deasserted the processor will repeat the current address cycle until an external agent signals it is ready, by asserting **EOK***. There is a one cycle delay from the assertion of **EOK*** to the state in which the Read/Write command becomes valid. **EOK*** must be asserted for two consecutive cycles for the command issue completion. Examples of **EOK*** use are given in Figure 6.3.23 and 6.3.24.
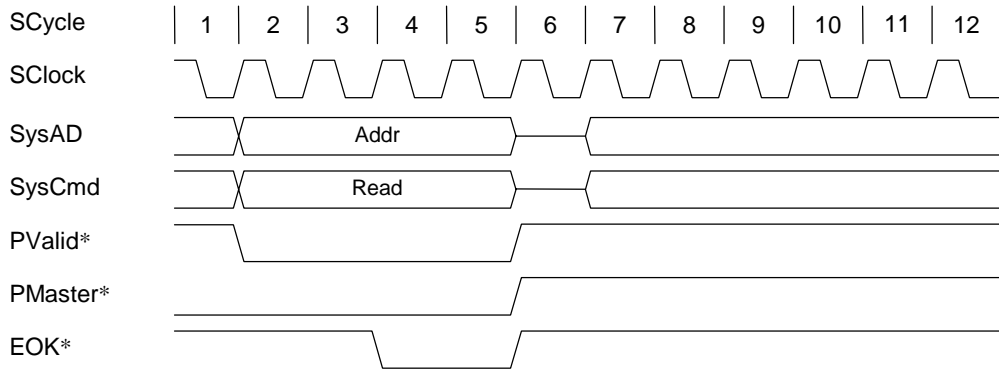
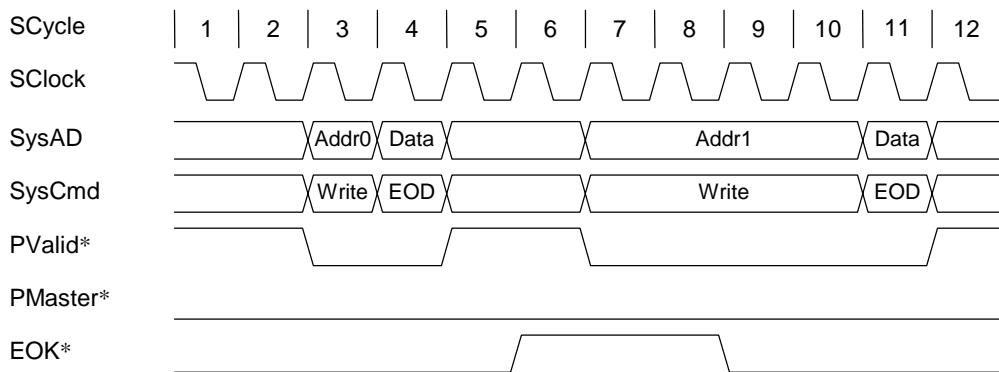Figure 6.3.23 Delayed Processor Read Request

Figure 6.3.24 Two Processor Write Requests, Second Write Delayed

6.3.13 Data Rate Control

The System interface supports a maximum data rate of one word per cycle, and an external agent may deliver data to the processor at this maximum data rate. The rate at which data is delivered to the processor can be controlled by the external agent by driving data and asserting **EValid*** only when it wants data to be available.

The processor interprets cycles as valid data cycles when **EValid*** is asserted and the SysCmd bus contains a data identifier. The processor continues to accept data until the end of data (EOD) indicator is received.

The rate at which the processor transmits data to an external agent is programmed in the *WBRATE* field in *G2S Config* register. Data patterns are specified using the letters **D** and **x** (**D** indicates a data cycle and **x** indicates an unused, or idle, cycle). A data pattern is specified as a sequence of data and unused cycles that will be repeated to provide the appropriate number of data cycles for a given transfer. For example, a data pattern of **DDxx** indicates a data rate of two words every four cycles.

TX4955A supports two data rates, **D** and **Dxx**. During a cycle indicated by an **x**, the processor continues to hold the same data as the previous cycle.

A processor block write request for two words with Dxx pattern is illustrated in Figure 6.3.25; this transaction results from a store doubleword instruction.
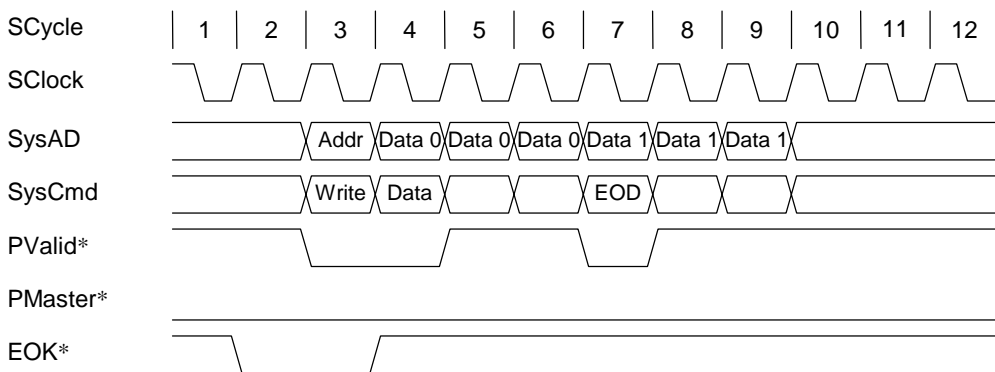
Figure 6.3.25 Processor Block Write Request With Dxx Data Rate

6.3.14   Consecutive SysAD Bus Transactions

The following figures (Figure 6.3.26 to 6.3.29) illustrate the minimum cycles required between consecutive bus transactions.

Figure 6.3.26   Processor Single Word Read Followed By Block Write Request

Figure 6.3.27   Consecutive Processor Single Word Write Requests With D Data Rate

Figure 6.3.28   Consecutive Processor Single Word Write Requests With Dxx Data Rate

Figure 6.3.29 Consecutive Processor Write Requests Followed By External Write Request

Block Read Maximum Rate

Maximum block reads can occur with the following data rate:

AxD...DxAxD...D (1 cycle between D and A)

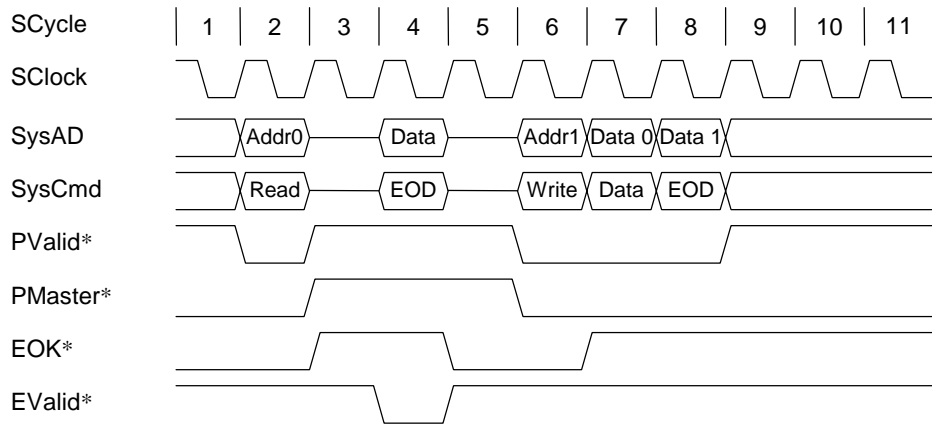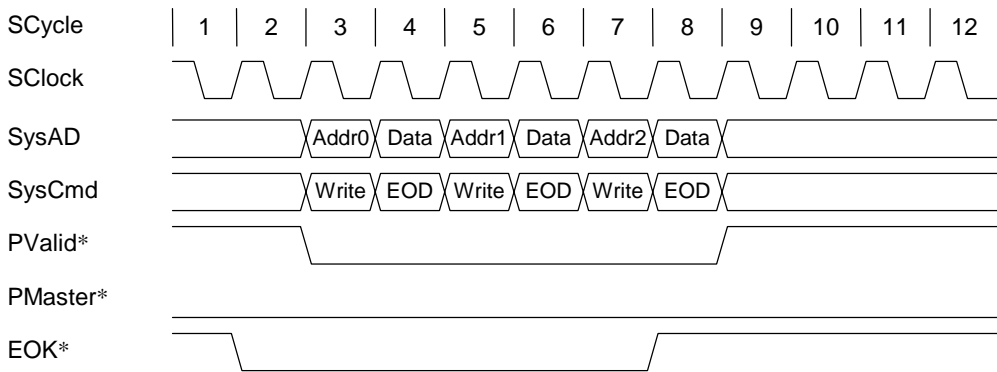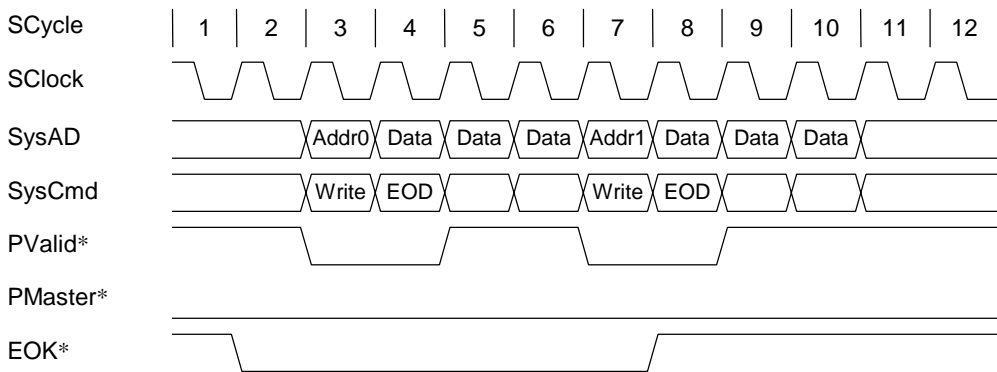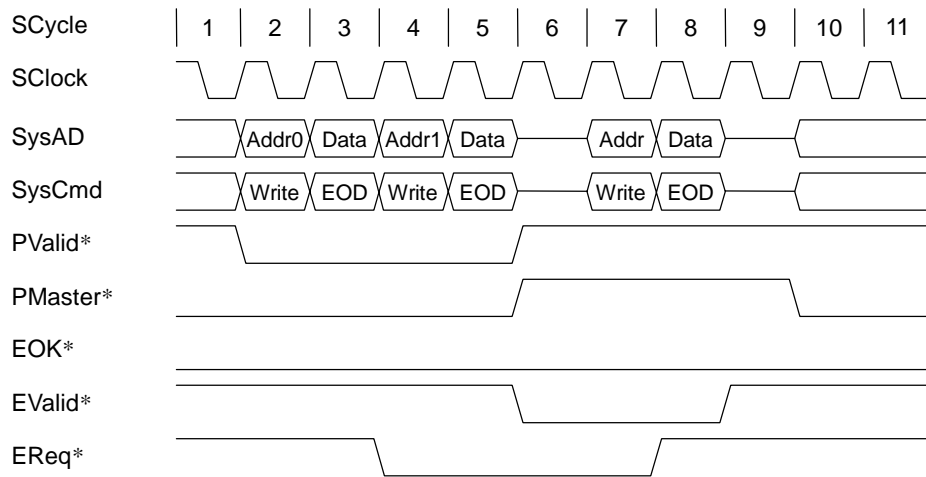where **A** is the address, **D** is data (four words, or DDDD, in the data cache miss, and 8 words, or DDDDDDDD, in an instruction cache miss), and **x** is an idle cycle.

Back-to-Back Instruction Cache Misses

With a PClock to SClock ratio of 2:1, back-to-back instruction cache misses can be refilled with the following data rate:

AxDDDDDDDDxxxAxDDDDDDDD (3 cycles between D and A)

That is, the address is followed by an idle cycle, the instruction is executed, three idle cycles occur, followed by the next address. This pattern is valid for the case in which two sequential instructions miss in the instruction cache, each instruction residing on a different cache line.

Running completely in uncached space (every instruction is uncached and a cache miss) results in a similar data pattern:

AxDxxxAxD (3 cycles between D and A)

Back-to-Back Uncached Loads

With a PClock to SClock ratio of 2:1, back-to-back uncached doubleword loads have the following data rate:

AxDDxxxAxDD (3 cycles between D and A)

That is, the address is followed by an idle cycle, a doubleword of data, three idle cycles, and the next address.

Back-to-back uncached word loads have the following data rate:

AxDxxxAxD (3 cycles between D and A)

6.3.15 Starvation and Deadlock Avoidance.

Careful use of the **EReq*** signal allows a system to avoid starvation and deadlock situations.

Whenever an external agent needs the bus, it can request the bus by asserting **EReq***. The external agent is guaranteed to gain mastership of the bus after accepting at most one read/write request from the processor. If the external agent also deasserts **EOK***, it is guaranteed to gain mastership of the bus without accepting any read/write request from the processor.

The external agent can allow the processor to gain bus mastership, perform one read/write request and then relinquish mastership by the following sequence of actions:

1. deassert **EReq***

2. assert **EReq***

3. arbitrate for the bus while asserting **EOK***

The minimum deassertion of **EReq*** can be one cycle in length.

shows an external agent relinquishing the bus to allow a single read/write request from the processor. The external agent must be ready to accept this request by keeping **EOK*** asserted, otherwise the read/write request is held off or killed and the processor relinquishes bus mastership without extending a request. This could lead to starvation of the processor.
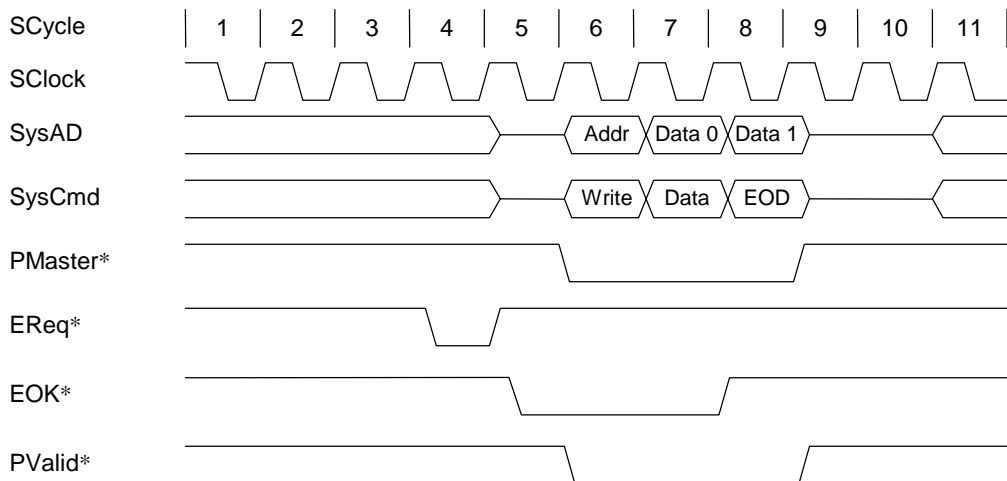
Figure 6.3.30 External Agent Gives Up Bus for One Processor Request

6.3.16 Discarding and Re-Executing Read Command

Figure 6.3.31 illustrates how a processor single read request is discarded and reexecuted. The following sequence describes the protocol.

- Because the **EOK*** signal is low in cycle 5, the processor tries to issue an address (cycle 6).

- If the **EOK*** signal is high at this point, the processor discards this read request and enters the slave status in the next cycle.

- Because the **EReq*** signal is inactive, the processor returns to the master status again and reissues a read request. Because the **EOK*** signal is low in both the cycles 7 and 8, the issuance cycle of the read request is determined.

- The external agent outputs data at the requsted address.
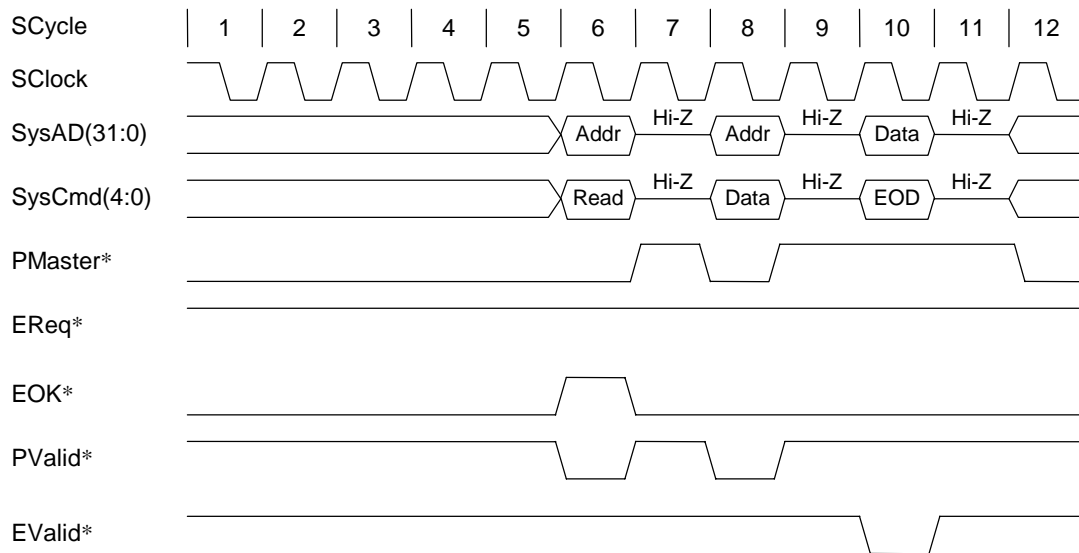
Figure 6.3.31 Discarding and Re-executing Processor Single Read Request

6.3.17   Multiple Drivers on the SysAD Bus

In most applications, the SysAD bus is a point-to-point connection between the processor and a bidirectional, registered transceiver located in an external agent. In this application, the SysAD bus has two possible drivers: the processor and the external agent.

However, an application may add additional drivers and receivers to the SysAD bus, allowing transmissions over the SysAD bus that bypass the processor. To accomplish this, the external agent(s) must coordinate its use of the SysAD bus by using arbitration handshake signals such as **EReq*** and **PMaster***.

To implement an independent transmission on the SysAD bus that does not involve the processor, the system executes the following sequence of actions:

1. The external agent(s) requests the SysAD bus by asserting **EReq***.
2. The processor releases the System interface to slave state.
3. The external agent(s) allows independent transmission over the SysAD bus, making certain the **EValid*** input to the processor is not asserted while the transmission occurs.
4. When the transmission is complete, the external agent(s) deasserts **EReq*** to return the system interface to master state.

To implement multiple drivers, separate **Valid** lines are required for non-processor chips to communicate.

6.3.18 Signal Codes

System interface commands and data identifiers are encoded in five bits on the SysCmd bus and transmitted between the processor and external agent during address and data cycles.

- When **SysCmd (4)** is a 0, the current cycle is an address cycle and **SysCmd (3:0)** contains a command.
- When **SysCmd (4)** is a 1, the current cycle is a data cycle and **SysCmd (3:0)** identifies data.

For commands and data identifiers associated with external requests, all bits and fields have a value or a suggested value.

For System interface commands and data identifiers associated with processor requests, reserved bits and reserved fields in the command or data identifier are undefined, except where noted.

For all System interface commands, the SysCmd bus specifies the system interface request type. The encoding of **SysCmd (4)** for system interface commands is Table 6.3.3.

Table 6.3.3 Encoding of System Interface Commands in *SysCmd (4)*

SysCmd(4)	Command
0	Address Cycle
1	Data Cycle

For address requests, the remainder of the SysCmd bus specifies the attributes of the address request, as follows:

- **SysCmd (3)** encodes the address request type.
- **SysCmd (2:0)** indicates the size of the address requests.

The encoding of SysCmd (3:2) for address requests is shown in Table 6.3.4.

Table 6.3.4 Encoding of **SysCmd(3)** and **SysCmd(2)** for Address Cycle

SysCmd (3)	Command	SysCmd (2)	Request Size
0	Read Request	0	Single data
1	Write Request	1	Block data

Note:TX4955A support only External Single data write request.

The encoding of **SysCmd (1:0)** for block or single address requests is shown in Table 6.3.5 and 6.3.6, respectively.

Table 6.3.5 Encoding of **SysCmd (1:0)** for Block Address Requests

SysCmd (1:0)	Block Size
0	Reserved.
1	Four words(only Data cache).
2	Eight worde.
3	Reserved.

Table 6.3.6   Encoding of *SysCmd (1:0)* for Single Address Requests

SysCmd (1:0)	Data size.
0	One byte valid (byte)
1	Two bytes valid (halfword).
2	Three bytes valid (tribyte).
3	Four bytes valid (single word)

The encoding of SysCmd (3:0) for processor data identifiers is described in Table 6.3.7. The encoding of SysCmd (3:0) for external data identifiers is illustrated in Table 6.3.8.

Table 6.3.7 Encoding of **SysCmd(3:0)** for Processor Data Identifiers

SysCmd (3)	Last Data Element Indication
0	Last data element
1	Not the last data element
SysCmd (2)	Reserved
SysCmd (1)	*Reserved for*: Good Data Indication
	Processor drives 0 (Data is error free)
SysCmd (0)	*Reserved for*: Data Checking Enable
	Processor drives 1 (Disable data checking)

Table 6.3.8 Encoding of **SysCmd(3:0)** for External Data Identifiers

SysCmd (3)	Last Data Element Indication
0	Last data element
1	Not the last data element
SysCmd (2)	Response Data Indication
0	Data is response data
1	Data is not response data
SysCmd (1)	*Reserved for*: Good Data Indication
0	Data is error free
1	Data is erroneous
SysCmd (0)	*Reserved for*: Data Checking Enable
	Processor ignores this field (Suggested drive of 1, disable data checking)

Note: External read requests for processor resources are not supported in the TX4955A processor.

6.3.19 Physical Addresses

Physical addresses are driven on all 32 bits (bits 31 through 0) of the SysAD bus during address cycles. Addresses associated with single read and write requests are aligned for the size of the data element; specifically, for single word requests, the low order two bits of the address are zero, for halfword requests, the low order bit of the address is zero. For byte and tribyte requests, the address provided is a byte address.

External agents returning read response data must support subblock ordering. Addresses associated with block read requests are aligned to the word of the desired data. The order in which data is returned in response to a processor block read request is:

- the word containing the addressed data word is returned first
- the remaining word(s) in the block are returned next, sequentially

Block writes are always block aligned

6.3.20 Mode Register of System Interface (G2SConfig)

The Mode Register of System Interface (G2SCofig) is a read/write register.

This register extend from TX4300.

This register is only WORD-Access.

Table 6.3.9 G2SConfig

Address	Field	Desctiption
0xF_FF10_0000	G2SConfig	Mode Register of System Interface

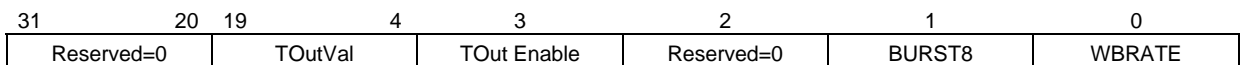| 31	20	19	4	3		2		1		0
Reserved=0		TOutVal		TOut Enable		Reserved=0		BURST8		WBRATE

Figure 6.3.32 G2SConfig Register Format

Table 6.3.10 G2SConfig Register Formats (MODE43* = 0)

Bit	Field	Description	ColdReset	read/write
31~20	—	Reserved	Undefined	read
19~4	TOutVal	Set Data of Read Time Out Counter	0xFFFF	read/write
3	TOutEnable	Enable bit of Read Time Out Counter 0: Disable 1: Enable	0	read/write
2	—	Reserved	Undefined	
1	BURST8	Data Formats at 8-word burst write 0: Double burst mode 4 words × 2 1: Single burst mode 8 words	0	read/write
0	WBRATE	Set bit of Data Out Formats 0: Every cycle Data Out 1: 4-word Data Out per 12 cycles	0	read/write

6.3.21 Read Time Out Counter (MODE43∗ = 0)

This counter is used to detect time-out when data is not returned during read.

The counter normally is set by loading the G2SConfig register's TOutVal as its

initial value. When one of the conditions below is met, the counter counts down

one every bus cycle and upon reaching the terminal count of 0, generates a time-out signal and asserts a bus error signal for one cycle before entering an idle state. If none of the following conditions is met, the value of TOutVal is reloaded into the counter.

- TOutEnable = 1
- Waiting for data

Note1: For TX4955A, Read Time Out Counter does not work when MODE43∗ = 1.
When the Read Time Out Counter functions is used, the value of the more than which added 24 the twice of the wait cycle of the main memory must be set to the TOutVal field.

Note2: Read Time Out Error occurs by writing 0x0 into TOutVal while Read Time Out Counter is disable.
So TOutVal must be set except for 0x0.

Chapter 7.   JTAG Interface

The TX4955A processor provides a boundary-scan interface that is compatible with Joint Test Action Group (JTAG) specifications, using the industry-standard JTAG protocol (IEEE Standard 1149.1/D6).

This chapter describes that interface, including descriptions of boundary scanning, the pins and signals used by the interface, and the Test Access Port (TAP).

7.1   What Boundary Scanning Is

With the evolution of ever-denser integrated circuits (ICs), surface-mounted devices, double-sided component mounting on printed-circuit boards (PCBs), and buried vias, in-circuit tests that depend upon making physical contact with internal board and chip connections have become more and more difficult to use. The greater complexity of ICs has also meant that tests to fully exercise these chips have become much larger and more difficult to write.

One solution to this difficulty has been the development of *boundary-scan* circuits. A boundary-scan circuit is a series of shift register cells placed between each pin and the internal circuitry of the IC to which the pin is connected, as shown in Figure 7.1.1. Normally, these boundary-scan cells are bypassed; when the IC enters test mode, however, the scan cells can be directed by the test program to pass data along the shift register path and perform various diagnostic tests. To accomplish this, the tests use the four signals described in the next section: **JTDI**, **JTDO**, **JTMS**, **JTCK, and TRST**∗.
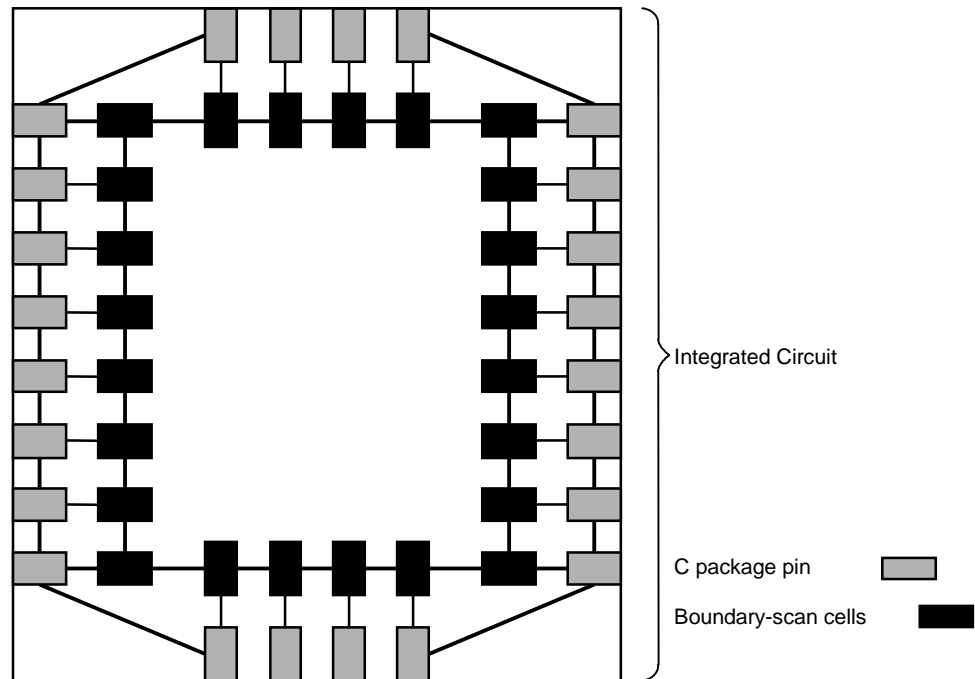

Figure 7.1.1 JTAG Boundary-scan Cells

7.2   Signal Summary

The JTAG interface signals are listed below and shown in Figure 7.2.1.

- JTDI   JTAG serial data in
- JTDO   JTAG serial data out
- JTMS   JTAG test mode select
- JTCK   JTAG serial clock input
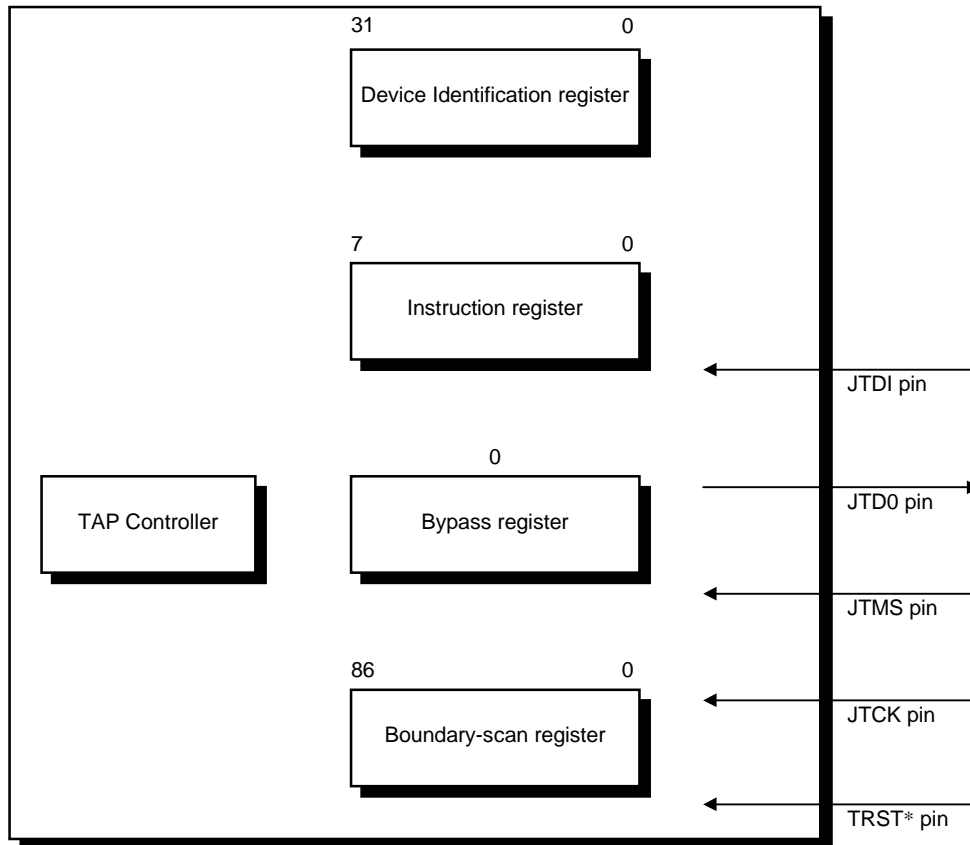- TRST   JTAG test reset input

Figure 7.2.1 JTAG Interface Signals and Registers

The JTAG boundary-scan mechanism (referred to in this chapter as *JTAG mechanism*) allows testing of the connections between the processor, the printed circuit board to which it is attached, and the other components on the circuit board.

The JTAG mechanism does not provide any capability for testing the processor itself.

7.3 JTAG Controller and Registers

The processor contains the following JTAG controller and registers:

- *Instruction* register
- *Boundary-scan* register
- *Bypass* register
- *ID Code* register
- Test Access Port (TAP) controller

The processor executes the standard JTAG EXTEST operation associated with External Test functionality testing.

The basic operation of JTAG is for the TAP controller state machine to monitor the JTMS input signal. When it occurs, the TAP controller determines the test functionality to be implemented. This includes either loading the JTAG instruction register (IR), or beginning a serial data scan through a data register (DR), listed in Table 8-1. As the data is scanned in, the state of the JTMS pin signals each new data word, and indicates the end of the data stream. The data register to be selected is determined by the contents of the *Instruction* register.

7.3.1 Instruction Register

The JTAG *Instruction* register includes eight shift register-based cells; this register is used to select the test to be performed and/or the test data register to be accessed. As listed in Table 7.3.1, this encoding selects either the *Boundary-scan* register or the *Bypass* register or Device Identification register.

Table 7.3.1 JTAG Instruction Register Bit Encoding

Instruction Code (MSB → LSB)	Instruction	Selected Data Register
00000000	EXTEST	Boundary Scan Register
00000001	SAMPLE/PRELOAD	Boundary Scan Register
00000010	Reserved	Reserved
00000011	IDCODE	Device Identification register
00000100 ~ 01111111	Reserved	Reserved
10000000 ~ 11111110	Debug Support Unit	Please refer DSU section
11111111	BYPASS	Bypass register

Figure 7.3.1 shows the format of the *Instruction* register

7	6	5	4	3	2	1	0
MSB							LSB

Figure 7.3.1 Instruction Register

The instruction code is shifted out to the Instruction register from the LSB.

Figure 7.3.2 Instruction Register Shift Direction

7.3.2　Bypass Register

The *Bypass* register is 1 bit wide. When the TAP controller is in the Shift-DR (Bypass) state, the data on the JTDI pin is shifted into the *Bypass* register, and the *Bypass* register output shifts to the JTDO output pin.

In essence, the *Bypass* register is a short-circuit which allows bypassing of board-level devices, in the serial boundary-scan chain, which are not required for a specific test. The logical location of the *Bypass* register in the boundary-scan chain is shown in Figure 7.3.3. Use of the *Bypass* register speeds up access to boundary-scan registers in those ICs that remain active in the board-level test datapath.
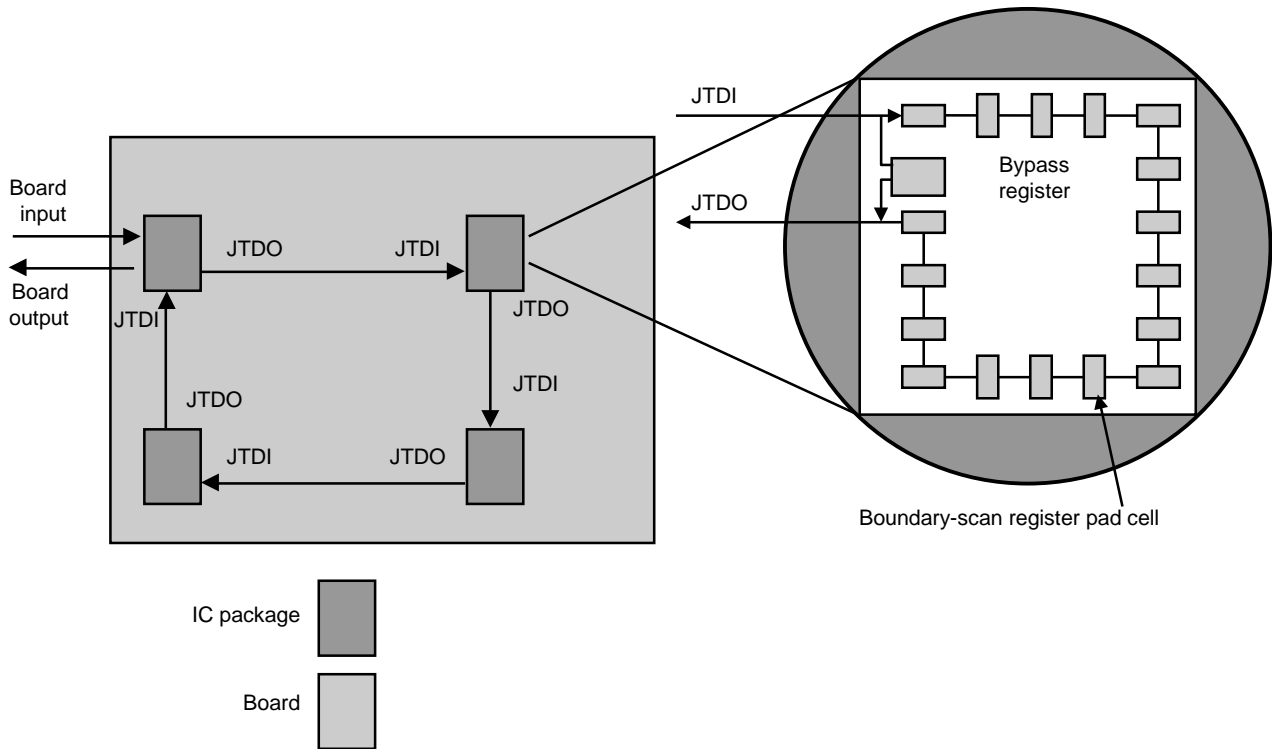
Figure 7.3.3 Bypass Register Operation

7.3.3   Boundary-Scan Register

The *Boundary Scan* register includes all of the inputs and outputs of the TX4955A processor, except some clock and phase lock loop signals. The pins of the TX4955A chip can be configured to drive any arbitrary pattern by scanning into the *Boundary Scan* register from the Shift-DR state. Incoming data to the processor is examined by shifting while in the Capture-DR state with the *Boundary Scan* register enabled.

The *Boundary-scan* register is a single, 87-bit-wide, shift register-based path containing cells connected to all input and output pads on the TX4955A processor. Figure 7.3.4 shows the *Boundary-scan* register.

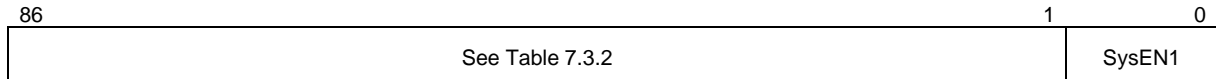| 86	1	0
See Table 7.3.2		SysEN1

Figure 7.3.4 Format of the Boundary-scan Register

The most-significant bit, SysEN (bit3 to bit0), are the JTAG output enable bit for all outputs of the processor. Output is enabled when those bit are set to 1 (default state).

The remaining 83 bits correspond to 83 signal pads of the processor.

At the end of this chapter, Table 7.3.2 lists the scan order of these 89 scan bits, starting from **JTDI** and ending with **JTDO**.

The JTDI input is loaded to the LSB of the Boundary Scan register. The MSB of the Boundary Scan register is retrieved from the JTDO output.

7.3.4   Device Identification Register

The Device Identification register is a 32-bit shift register. It is used to read serially from the IC the identification code indicating the IC manufacturer, part number, and version.

The following shows the Device Identification register structure.

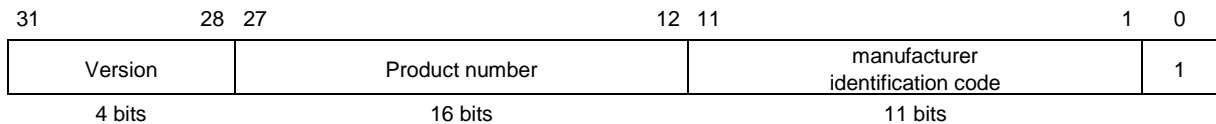| 31 28	27 12	11 1	0
Version	Product number	manufacturer identification code	1
4 bits	16 bits	11 bits	

Figure 7.3.5 Device Identification Register

The TX4955A device identification code is 0x00017031.

The device identification code is shifted out starting from the LSB.
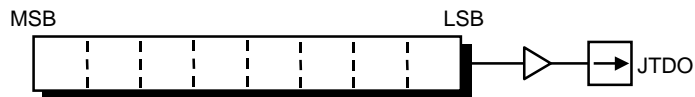

Figure 7.3.6 Devide Identification Register shift Direction

7.3.5   Test Access Port (TAP)

The Test Access Port (TAP) consists of the five signal pins: **TRST*** **JTDI**, **JTDO**, **JTMS**, and **JTCK**. Serial test data and instructions are communicated over these five signal pins, along with control of the test to be executed.

As Figure 7.3.7 shows, data is serially scanned into one of the four registers (*Instruction* register, *Bypass* register, Device Identification register, or the *Boundary-scan* register) from the **JTDI** pin, or it is scanned from one of these four registers onto the **JTDO** pin.

The **JTMS** input controls the state transitions of the main TAP controller state machine.

The **JTCK** input is a dedicated test clock that allows serial JTAG data to be shifted synchronously, independent of any chip-specific or system clocks.
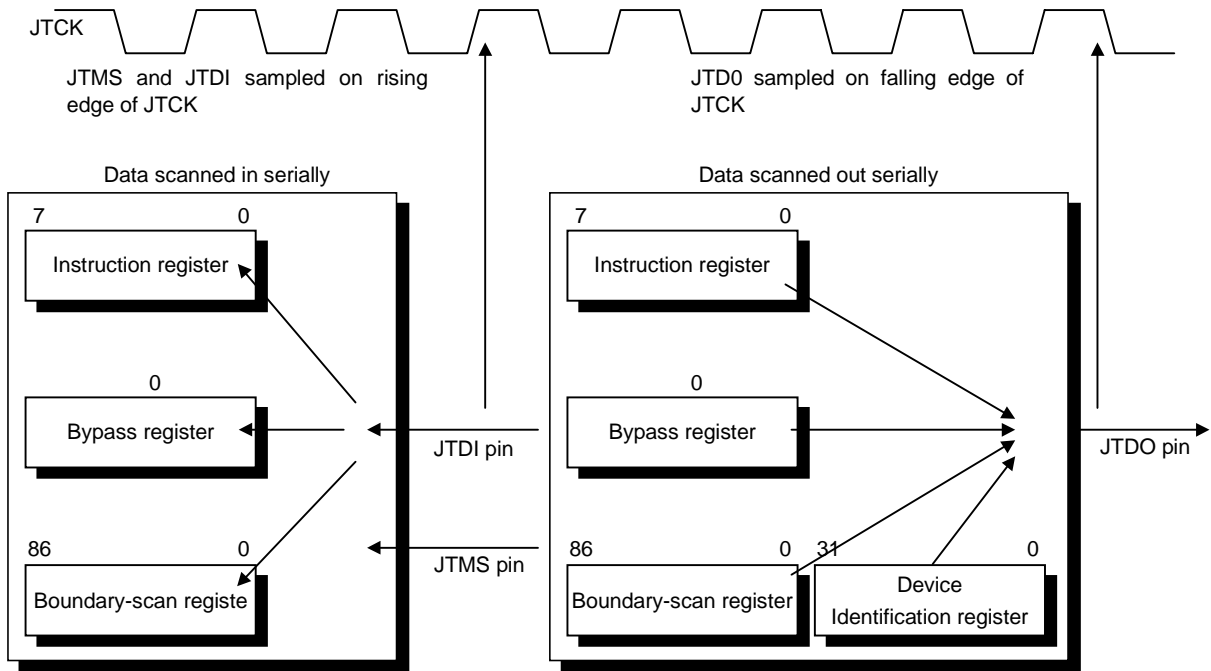

Figure 7.3.7 JTAG Test Access Port

Data on the **JTDI** and **JTMS** pins is sampled on the rising edge of the **JTCK** input clock signal. Data on the **JTDO** pin changes on the falling edge of the **JTCK** clock signal.

7.3.6   TAP Controller

The processor implements the 16-state TAP controller as defined in the IEEE JTAC specification.

7.3.7   Controller Reset

The TAP controller state machine can be put into Reset state the following:

- assertion of the **TRST*** signal (Low) resets the TAP controller.
- keeping the **JTMS** input signal asserted through five consecutive rising edges of **JTCK** input.

In either case, keeping **JTMS** asserted maintains the Reset state.

7.3.8   TAP Controller

The state transition diagram of the TAP controller is shown in Figure 7.3.8. Each arrow between states is labeled with a 1 or 0, indicating the logic value of JTMS that must be set up before the rising edge of JTCK to cause the transition.
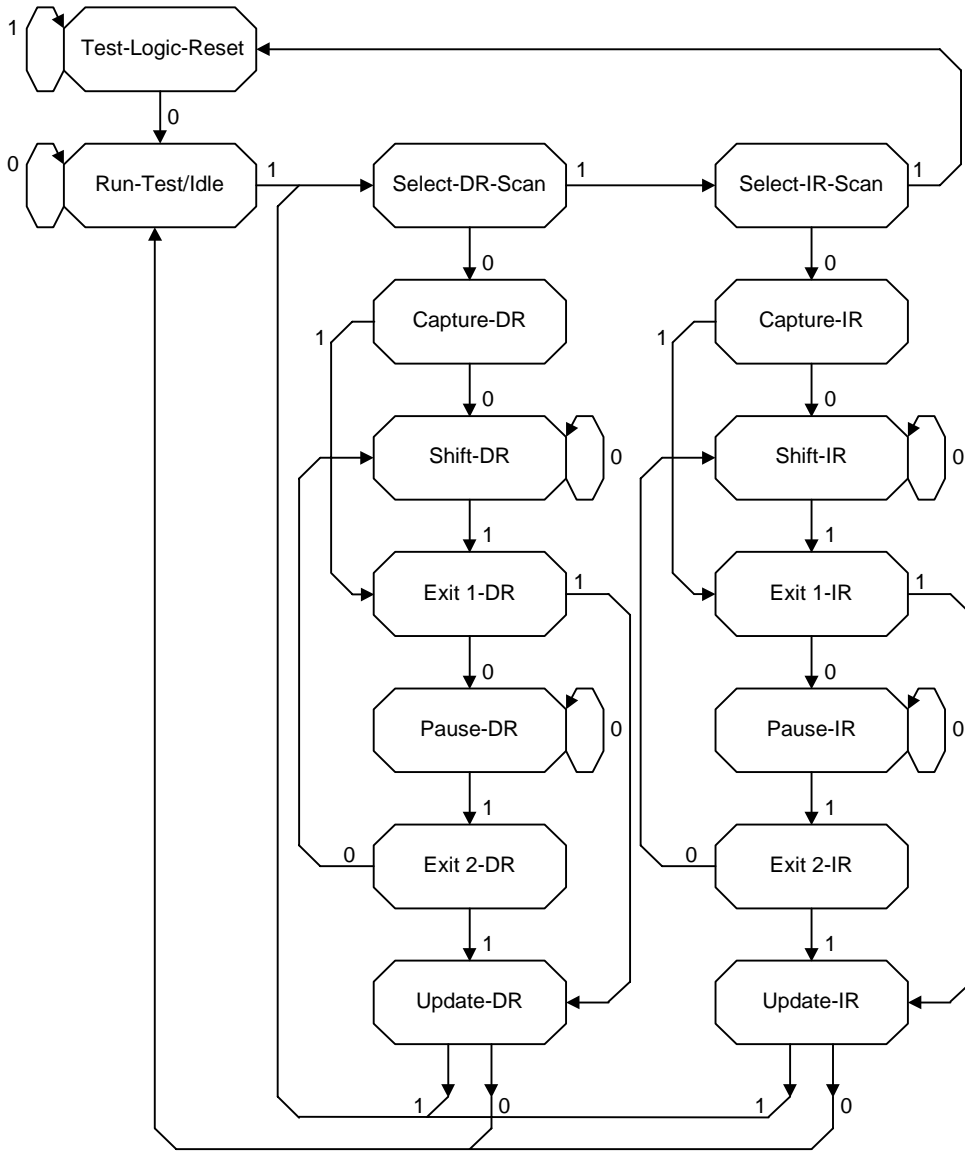

Figure 7.3.8 TAP Controller State Diagram

The following paragraphs describe each of the controller state. The left vertical column in Figure 7.3.8 is the data column, and the right vertical column is the instruction column. The data column and instruction column reference data register (DR) and instruction register (IR), respectively.

- Test-Logic-Reset
  When the TAP controller is in the Reset state, the value 0x3 is loaded into the
  parallel output latch, selecting the Device Identification register as default. The
  three most significant bits of the Boundary-scan register are cleared to 0, disabling
  the outputs.

  The controller remains in this state while JTMS is high. If JTMS is held low while
  the controller is in this state, then the controller moves to the Run-Test/Idle state.

- Run-Test/Idle
  In the Run-Test/Idle state, the IC is put in a test mode only when certain
  instructions such as a built-in self test (BIST) instruction are present. For
  instructions that do not cause any activities in this state, all test data registers
  selected by the current instruction retain their previous states.

  The controller remains in this state while JTMS is held low. When JTMS is high,
  the controller moves to the Select-DR-Scan state.

- Select-DR-Scan
  This is a temporary controller state. Here, the IC does not execute any specific
  functions.

  If JTMS is held low when the controller is in this state, then the controller moves
  to the Capture-DR state. If JTMS is held high, the controller moves to the
  Select-IR-Scan state in the instruction column.

- Select-IR-Scan
  This is a temporary controller state. Here, the IC does not execute any specific
  functions.

  If JTMS is held low when the controller is in this state, then the controller moves
  to the Capture-IR state. If JTMS is held high, the controller returns to the
  Test-Logic-Reset state.

- Capture-DR
  In this controller state, if the test data register selected by the current instruction
  on the rising edge of JTCK has parallel inputs, then data can be parallel-loaded
  into the shift portion of the data register. If the test data register does not have
  parallel inputs, or if data need not be loaded into the selected data register, then
  the data register retains its previous state.

  If JTMS is held low while the controller is in this state, the controller moves to the
  Shift-DR state. If JTMS is held high, the controller moves to the Exit1-DR state.

- Shift-DR
  In this controller state, the test data register connected between JTDI and JTDO
  shifts data one stage forward towards its serial output.

  When the controller is in this state, then it remains in the Shift-DR state if JTMS
  is held low, or moves to the Exit1-DR state if JTMS is held high.

- Exit 1-DR
  This is a temporary controller state.

  If JTMS is held low when the controller is in this state, then the controller moves
  to the Pause-DR state. If JTMS is held high, the controller moves to the
  Update-DR state.

- Pause-DR
  This state allows the shifting of the data register selected by the instruction
  register to be temporarily suspended. Both the instruction register and the data
  register retain their current states.

  When the controller is in this state, then it remains in the Pause-DR state if JTMS
  is held low, or moves to the Exit2-DR state if JTMS is held high.

- Exit 2-DR
  This is a temporary controller state.

  When the controller is in this state, then it returns to the Shift-DR state if JTMS is
  held low, or moves on to the Update-DR state if JTMS is held high.

- Update-DR
  In this state, data is latched, on the falling edge of JTCK, onto the parallel outputs
  of the data registers from the shift register path. The data held at the parallel
  output does not change while data is shifted in the associated shift register path.

  When the controller is in this state, it moves to either the Run-Test/Idle state if
  JTMS is held low, or the Select-DR-Scan state if JTMS is held high.

- Capture-IR
  In this state, data is parallel-loaded into the instruction register. The two least
  significant bits are assigned the values "01". The higher-order bits of the
  instruction register can receive any design specific values. The Capture-IR state is
  used for testing the instruction register. Faults in the instruction register, if any
  exists, may be detected by shifting out the data loaded in it.

  When the controller is in this state, it moves to either the Shift-IR state if JTMS is
  low, or the Exit1-IR state if JTMS is high.

- Shift-IR
  In this state, the instruction register is connected between JTDI and JTDO and
  shifts the captured data toward its serial output on the rising edge of JTCK.

  When the controller is in this state, it remains in the Shift-IR state if JTMS is low,
  or moves to the Exit1-IR state if JTMS is high.

- Exit 1-IR
  This is a temporary controller state.

  When the controller is in this state, then it moves to either the Pause-IR state if JTMS is held low, or the Update-IR state if JTMS is held high.

- Pause-IR
  This state allows the shifting of the instruction register to be temporarily suspended. Both the instruction register and the data register retain their current states.

  When the controller is in this state, it remains in the Pause-IR state if JTMS is held low, or moves to the Exit2-IR state if JTMS is held high.

- Exit 2-IR
  This is a temporary controller state.

  When the controller is in this state, it moves to either the Shift-IR state if JTMS is held low, or the Update-IR state if JTMS is held high.

- Update-IR
  This state allows the instruction previously shifted into the instruction register to be output in parallel on the rising edge of JTCK. Then it becomes the current instruction, setting a new operational mode.

  When the controller is in this state, it moves to either the Run-Test/Idle state if JTMS is low, or the Select-DR-Scan state if JTMS is high.

Tables 7.3.2 shows the boundary scan order of the processor signals.

Table 7.3.2 Tx4955A JTAG Boundary-Scan Ordering

[JTDI]	1: SysEN1	2: SysEN2	3: SysEN3	4: SysEN4	5: BufSel1	6: SysAD4
7: SysAD5	8: SysAD6	9: SysAD7	10: SysAD8	11: SysAD9	12: SysAD10	13: SysAD11
14: SysAD12	15: SysAD13	16: SysAD14	17: SysAD15	18: BufSel0	19: PCST3	20: PCST2
21: PCST1	22: PCST0	23: RdRdy*	24: WrRdy*	25: ValidIn*	26: ValidOut*	27: Release*
28: PLLRES*	29: TintDis	30: SysCmd0	31: SysCmd1	32: SysCmd2	33: SysCmd3	34: SysCmd4
35: SysCmd5	36: SysCmd6	37: SysCmd7	38: SysCmd8	39: SysCmdP	40: HALTDOZE	41: INT0*
42: INT1*	43: INT2*	44: INT3*	45: INT4*	46: INT5*	47: TPC3	48: TPC2
49: TPC1	50: DCLK	51: NMI*	52: ExtRqst*	53: Reset*	54: ColdReset*	55: Endian
56: SysAD16	57: SysAD17	58: SysAD18	59: SysAD19	60: SysAD20	61: SysAD21	62: SysAD22
63: SysAD23	64: SysAD24	65: SysAD25	66: SysAD26	67: SysAD27	68: MODE43*	69: DivMode1
70: DivMode0	71: SysAD28	72: SysAD29	73: SysAD30	74: SysAD31	75: SysADC2	76: SysADC3
77: SysADC0	78: SysADC1	79: SysAD0	80: SysAD1	81: SysAD2	82: SysAD3	83: PCST8
84: PCST7	85: PCST6	86: PCST5	87: PCST4	[JTDO]		

7.4   Instructions for JTAG

This section defines the instructions supplied and the operations that occur in response to those instructions.

7.4.1   The EXTEST Instruction

This instruction is used for external interconnect test, and targets the boundary scan register between JTDI and JTDO. The EXTEST instruction permits BSR cells at output pins to shift out test patterns in the Update-DR state and those at input pins to capture test results in the Capture-DR state.

Typically, before EXTEST is executed, the initialization pattern is first shifted into the boundary scan register using the SAMPLE/PRELOAD instruction. In the Update-DR state, the boundary scan register loaded with the initialization pattern causes known data to be driven immediately from the IC onto its external interconnects. This eliminates the possibility that bus conflicts damage the IC outputs. The flow of data through the boundary scan register while the EXTEST instruction is selected is shown in Figure 7.4.1, which follows:
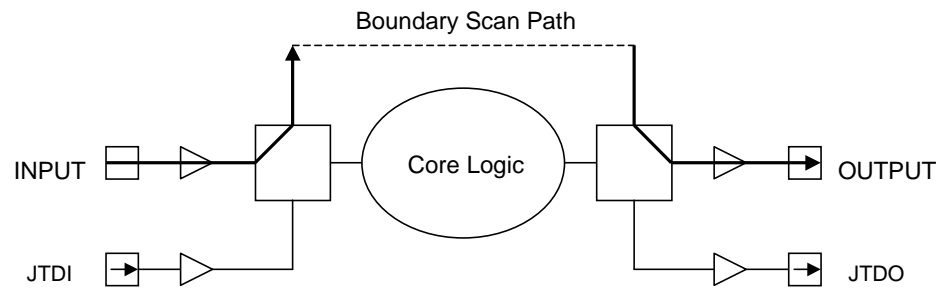

Figure 7.4.1 Test Data Flow While the EXTEST Instruction is Selected

The following steps describe the basic test algorithm of an external interconnect test.

1.   Initialize the TAP controller to the Test-Logic-Reset state.

2.   Load the instruction register with SAMPLE/PRELOAD. This causes the boundary scan register to be connected between JTDI and JTDO.

3.   Initialize the boundary scan register by shifting in determinate data.

4.   Then, load the initial test data into the boundary scan register.

5.   Load the instruction register with EXTEST.

6.   Capture the data applied to the input pin into the boundary scan register.

7.   Shift out the captured data while simultaneously shifting in the next test pattern.

8.   Read out the data in the boundary scan register onto the output pin.

Steps 6 to 8 are repeated for each test pattern.

7.4.2   The SAMPLE/PRELOAD Instruction

This instruction targets the boundary scan register between JTDI and JTDO. As the instruction's name implies, two functions are performed through use of the SAMPLE/ PRELOAD instruction.

• SAMPLE allows the input and output pads of an IC to be monitored. While it does so, it does not disconnect the system logic from the IC pins. The SAMPLE function occurs in the Capture-DR controller state. An example application of SAMPLE is to take a snapshot of the activity of the IC's I/O pins so as to verify the interaction between ICs during normal functional operation. The flow of data for the SAMPLE phase of the SAMPLE/PRELOAD instruction is shown in Figure 7.4.2.
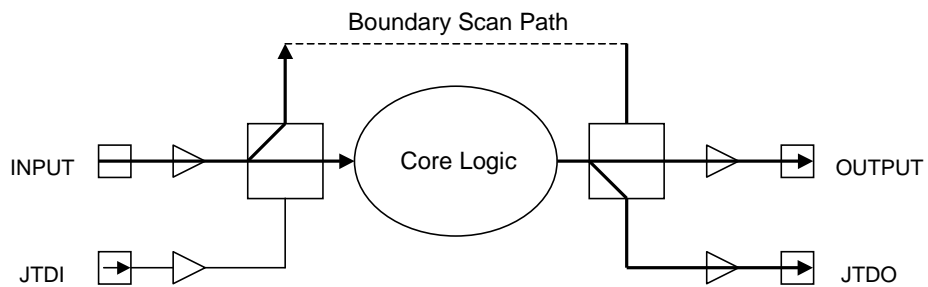
Figure 7.4.2 Test Data Flow While SAMPLE is Selected

• PRELOAD allows the boundary scan register to be initialized before another instruction is selected. For example, prior to selection of the EXTEST instruction, initialization data is shifted into the boundary scan register using PRELOAD as described in the previous subsection. PRELOAD permits shifting of the boundary scan register without interfering with the normal operation of the system logic. The flow of data for the PRELOAD phase of the SAMPLE/PRELOAD instruction is shown in Figure 7.4.3.
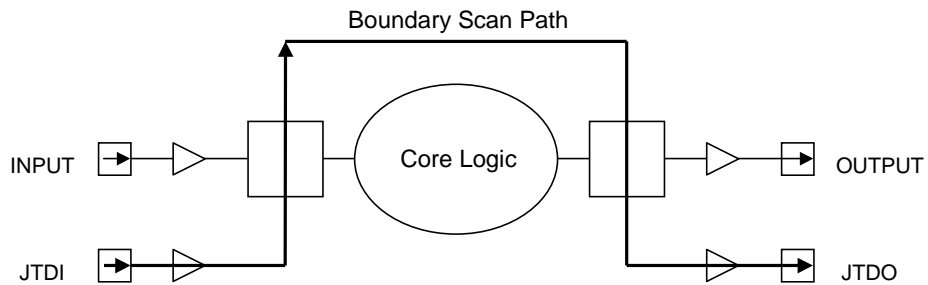
Figure 7.4.3 Test Data Flow While PRELOAD is Selected

7.4.3   The BYPASS Instruction

This instruction targets the bypass register between JTDI and JTDO. The bypass register provides a minimum length serial path through the IC (or between JTDI and JTDO) when the IC is not required for the current test. The BYPASS instruction does not cause interference to the normal operation of the on-chip system logic. The flow of data through the bypass register while the BYPASS instruction is selected is shown in Figure 7.4.4.
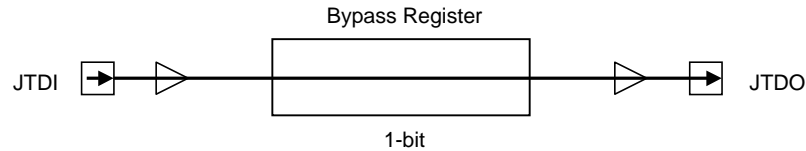
Figure 7.4.4 Test Data Flow While the Bypass Instruction is Selected

7.4.4   The IDCODE Instruction

This instruction targets the device identification register between JTDI and JTDO to identify manufacturer identity, part number, and version number for the part.

7.5　Note

This section discribes details of JTAG boundary-scan operation that are specific to the processor.

- The **MasterClock**, and **PLLCAP** signal pads do not supposrt JTAG.

- When performing a JTAG operation, be sure to run the **MasterClock** before and after a reset operation to properly release the processor reset.

- Reset for JTAG
  ① JTAG circuit is initialized by **TRST**∗ assertion. And then deassert **TRST**∗.
  ② At input to **JTMS** = 1 and asserted for more 5 JTCK cycles.

Chapter 8.   TX4955A Processor Interrupts

Four types of interrupt are available on the TX4955A. These are:

- one non-maskable interrupt, NMI
- six external interrupts
- two software interrupts
- one timer interrupt

These are described in this chapter.

8.1   Nonmaskable Interrupt

The non-maskable interrupt is signaled by asserting the **NMI*** pin (low), forcing the processor to branch to the Reset Exception vector. This pin is latched into an internal register by the rising edge of **SClock**, as shown in Figure 8.4.1. An NMI can also be set by an external write through the **SysAD** bus. On the data cycle, **SysAD (22)** acts as the write enable for **SysAD (6)**, which is the value to be written as the interrupt.

NMI only takes effect when the processor pipeline is running. Thus NMI can be used to recover the processor from a software hang (for example, in an infinite loop) but cannot be used to recover the processor from a hardware hang (for example, no read response from an external agent). NMI cannot cause drive contention on the **SysAD** bus and no reset of external agents is required.

This interrupt cannot be masked.

The **NMI*** pin is latched by the rising edge of **SClock**, however the NMI exception occurs in response to the falling edge of the **NMI*** signal, and is not level-sensitive.

Figure 8.4.1 shows the internal derivation of the **NMI** signal. The **NMI*** pin is latched into an internal register by the rising edge of **SClock**. Bit 6 of the Interrupt register is then ORed with the inverted value of **NMI*** to form the nonmaskable interrupt.

8.2   External Interrupts

External interrupts are set by asserting the external interrupt pins **Int (5:0)***. They also may be set by an external write through the **SysAD** bus. During the data cycle, **SysAD (21:16)** are the write enables for bits **SysAD (5:0)**, which are the values to be written as interrupts.

These interrupts can be masked with the *IM*, *IE*, and *EXL* fields of the *Status* register.

Note: Int5* is doubled as a Timer Interrupt. Therefore either is selected with the External pin (TintDis).

8.3   Software Interrupt

Software interrupts use bits 1 and 0 of the interrupt pending, *IP*, field in the *Cause* register. These may be written by software, but there is no hardware mechanism to set or clear these bits.

These interrupts are maskable.

8.4 Timer Interrupt

The timer interrupt signal is bit 15 of the *Cause* register, which is bit 7 of the interrupt pending, *IP*, field. The timer interrupt is set whenever the value of the *Count* register equals the value of the *Compare* register.

This interrupt is maskable through the *IM* field of the *Status* register.

Note: Timer Interrupt is doubled as a Int5∗. There fore either is selected with the External pin (TintDis).
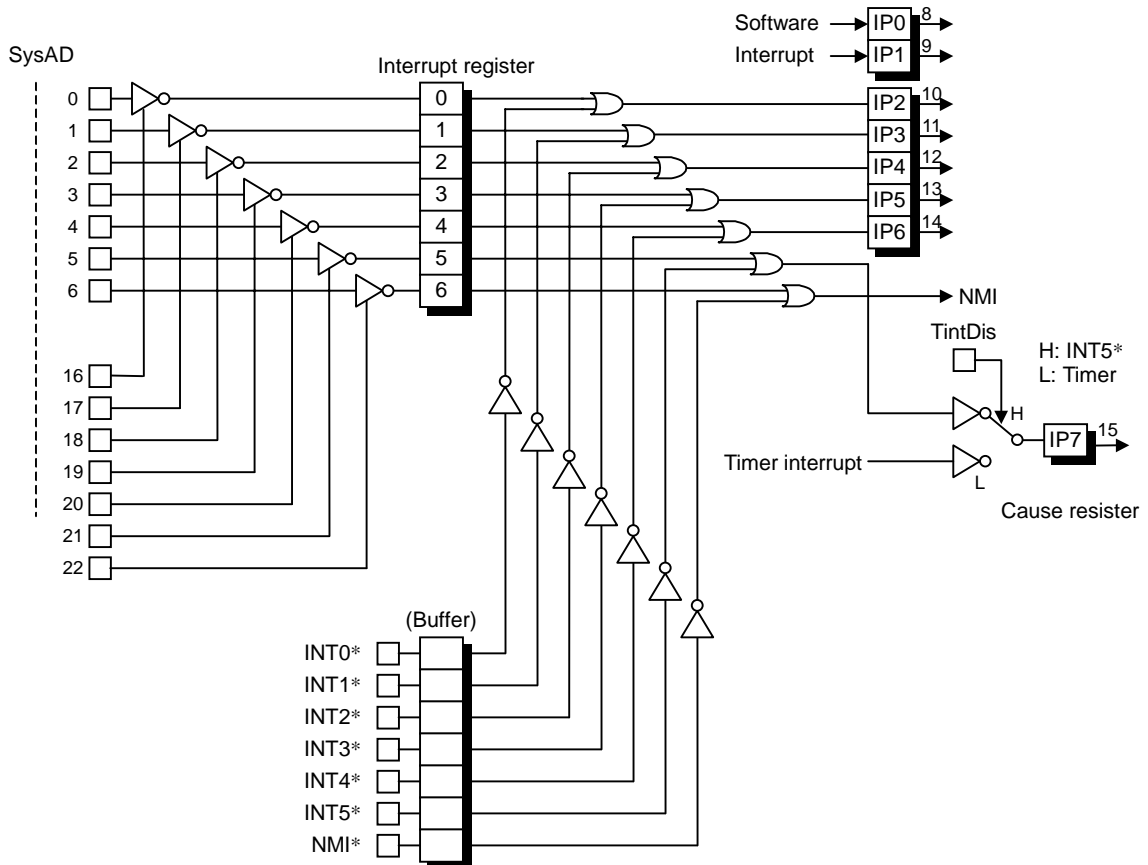
Note: Interrupt register : Please see Chapter 6 External Write Request

Figure 8.4.1 TX4955A Interrupt Control Circuit

8.5 Asserting Interrupts

External writes to the CPU are directed to various internal resources, based on an internal address map of the processor. An external write to any address writes to an architecturally transparent register called the *Interrupt* register; this register is available for external write cycles, but not for external reads.

During a data cycle, **SysAD (22:16)** are the write enables for the seven individual *Interrupt* register bits and **SysAD (5:0)** are the values to be written into these bits. This allows any subset of the *Interrupt* register to be set or cleared with a single write request. Figure 8.4.1 shows the mechanics of an external write to the *Interrupt* register, along with the nonmaskable interrupt described earlier.

Figure shows how the TX4955A hardware interrupts are readable through the *Cause* register.

- Bits 4:0 of the *Interrupt* register are bit-wise ORed with the current value of the interrupt pins **Int**∗**[4:0]** and the result is directly readable as bits 14:10 of the *Cause* register.

- Bit5 of the Interrupt register is ORed with the current value of the interrupt pin Int5∗. Bit15 of Cause register is selected this ORed signal or Timer Interrupt by TintDis.

IP (1:0) of the *Cause* register, which are described in Chapter 8 8.3, are software interrupts. There is no hardware mechanism for setting or clearing the software interrupts.

Figure 8.5.1 shows the masking of the TX4955A interrupt signals.

- *Cause* register bits 15:8 (IP7-IP0) are AND-ORed with *Status* register interrupt mask bits 15:8 (IM7-IM0) to mask individual interrupts.

- *Status* register bit 0 is a global Interrupt Enable (IE). It is ANDed with the output of the AND-OR logic to produce the TX4955A interrupt signal. The *EXL* bit in the *Status* register also enables these interrupts.
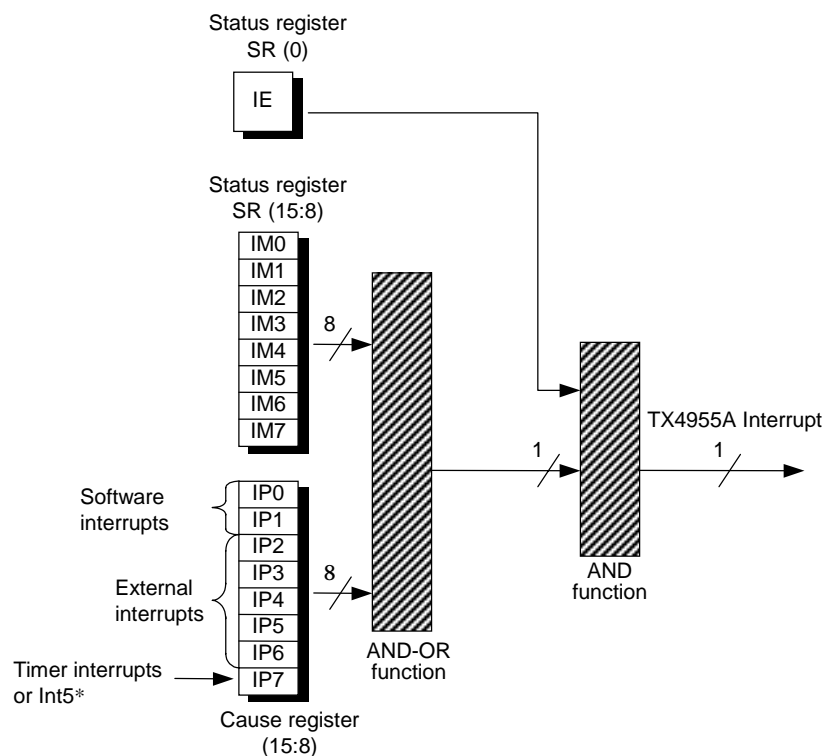

Figure 8.5.1 Masking of the TX4955A Interrupts

Chapter 9.   Debug Support Unit

9.1   Features

1. Utilizes JTAG interface compatible with IEEE Std. 1149.1.

2. Processor access to external processor probe to execute from the external trace memory during debug exception and boot time. This is to eliminate system memory for debugging purpose.

3. Supports DMA access through JTAG interface to internal processor bus to access internal registers, host system peripherals and system memory.

4. Debug functions
    * Instruction Address Break
    * Data Bus break
    * Processor Bus Break
    * Reset, NMI, Interrupt Mask

5. Instructions for Debug
    * SDBBP, DERET, CTC0, CFC0

6. CP0 Registers for Debug
    * Debug, DEPC, DESAVE

9.2   EJTAG interface

This interface consists of two modes of operation a Run Time Mode and Real Time Mode. The Run Time Mode provides functions such as processor Run, STOP, Single Step, and access to internal registers and system memory. The Real Time mode provides additional status pins used in conjunction with JTAG pins for Real Time Trace information.

Table 9.2.1 JTAG Interface

PIN NAME	I / O	FUNCTION
JTDI	I	**JTAG data input / Debug interrupt input** Run-time mode: Input serial data to JTAG data/instruction registers. Real-time mode: Interrupt input to change the debug unit state from real-time mode to run-time mode.
JTCK	I	**JTAG clock input** Clock input for JTAG. The JTDI and JTMS data are latched on rising edges of this clock.
JTDO/TPC(0)	O	**JTAG data output / Trace PC output** Data is serially shifted out from this pin. / Outputs a non-sequential program counter value synchronously with DCLK.
JTMS	I	**JTAG command** Controls mainly the status transition of the TAP controller state machine. When the serial input data is a JTAG command, apply a high signal (= 1) to this pin.
DCLK	O	**Debug clock (1/3 CPU clock)** Clock output for a real-time debug system. Timings of the serial monitor bus and PC trace interface signals all are defined by this debug clock DCLK. This Debug clock frequency is 1/3 that of Pclock.
PCST(8:0)	O	**PC trace status** Outputs PC trace status information and serial monitor bus mode.
TPC(3:1) Note 1	O	**Trace PC output** Outputs a non-sequential program counter value synchronously with DCLK.
TRST*	I	**Test reset input** Reset input for a real-time debug system. When TRST* is asserted (= 0), the debug support unit (DSU) is initialized.

Note1: Leave TPC (3-1) pins open when not using them as PC trace outputs for debugging.

9.3   JTAG Interface

Standard JTAG interface is used for on chip debugging during Run Time mode. The TX49 Debug Support Unit has following registers.

- Instruction Register
- Bypass Register
- Boundary-Scan Register
- Device Identification Register
- Implementation Register
- JTAG_Data_Register
- JTAG_Address_Register
- JTAG_Control_Register

9.4   Processor Access Overview

The core processor can access external processor probe for reading and writing to external monitor memory, registers and other external resources.

In addition the processor can execute from the external monitor memory located from 0xf_ff20 0000 to 0xf_ff2f ffff when the ProbEnb bit is set and the processor probe is turned ON. Any access to the monitor location from 0xf_ff20 0000 to 0xf_ff3f ffff are only allowed when the processor is in the debug mode (DM = 1).

9.5   Instruction

The instruction is a 8 bit field. Instructions for the TX49 Debug Support Unit are encoded between 0x80 and 0x9f and other codes are reserved for Toshiba Standard JTAG instructions (Includes EXTEST, SAMPLE/PRELOAD and IDCODE) and so on. Instructions are decoded as follows.

Figure 9.5.1 Instruction

Hex Value	Instruction	Description
0x00	EXTEST	Boundary Scan Register
0x01	SAMPLE/PRELOAD	Boundary Scan Register
0x02	Reserve	Reserve
0x03	IDCODE	Device Identification Register
0x83	EJTAG_ImpCode	Select Implementation Register
0x88	JTAG_ADDRESS_IR	Select JTAG_Address Register
0x89	JTAG_DATA_IR	Select JTAG_Data Register
0x8A	JTAG_CONTROL_IR	Select JTAG_Control Register
0x8B	JTAG_ALL_IR	Select JTAG_All Register
0x90	PCTRACE	PCIRACE Instruction
0xFF	BYPASS	Select BYPASS Register

Note: 0x80 ~ 0x9F the other code. Please does not used.

Any unused instruction between 0x80 and 0x9f defaulted to BYPASS instruction.

9.6   Debug Unit

9.6.1   Extended Instructions

- SDBBP
- DERET
- CTC0
- CFC0

9.6.2   Extended Debug Registers in CP0

- Debug Register
- Debug Exception PC (DEPC)
- Debug SAVE

9.7   Register Map

Table 9.7.1 Register Map

Address	Mnemonic	Description
0xf ff30 0000	DCR	Debug Control Register
0xf ff30 0008	IBS	Instruction Break Status
0xf ff30 0010	DBS	Data Break Status
0xf ff30 0018	PBS	Processor Break Status
0xf ff30 0100	IBA0	Instruction Break Address 0
0xf ff30 0108	IBC0	Instruction Break Control 0
0xf ff30 0110	IBM0	Instruction Break Address Mask 0
0xf ff30 0300	DBA0	Data Break Address 0
0xf ff30 0308	DBC0	Data Break Control 0
0xf ff30 0310	DBM0	Data Break Address Mask 0
0xf ff30 0318	DB0	Data Break Value 0
0xf ff30 0600	PBA0	Processor Bus Break Address 0
0xf ff30 0608	PBD0	Processor Bus Break Data 0
0xf ff30 0610	PBM0	Processor Bus Break Mask 0
0xf ff30 0618	PBC0	Processor Bus Break Control 0

9.8   Processor Bus Break Function

This function is to monitor the interface to core and provide debug interruption or trace trigger for a given physical address and data.

9.9　Debug Exception

Three kinds of debug exception are supported.

- Debug Single Step (DSS bit)
- Debug Breakpoint Exception (SDBBP Instruction)
- JTAG Break Exception (Jtagbrk bit in JTAG_Control_Register)

Note: During real time debugging, first two functions are disabled.

9.9.1　Debug Single Step (DSS)

When the debug register DSS bit is set, this exception has been raised each time one instruction is executed.

9.9.2　Debug Breakpoint exception (Dbp)

This exception is raised when SDBBP instruction is executed.

9.9.3　JTAG Break Exception

This exception is raised when JTAG unit set the Jtagbrk in JTAG_Control_Register.

9.9.4　Debug Exception Handling

Updates DEPC and Debug register.

Registers other than DEPC and Debug register retain their values.

9.9.5　Branching to debug handler

If the ProbEnb bit in JTAG_Control_Register[15] is set, the debug exception vector is located at

　　PC: 0xffff ffff ff20 0200

If the ProbEnb bit in JTAG_Conctrol_Register[15] is cleared, the debug exception vector is located at

　　PC: 0xffff ffff bfc0 0400

9.9.6　Exception handling when in Debug Mode (DM bit is set)

All interrupts including NMI are masked. When the NMI interrupt has occurred during Debug mode, it is stored internally and the NMI interrupt is taken after debug handler is finished (DM is clear)

9.10　Real Time PC TRACE Output

In real time mode non-sequential Program Counter and trace information are outputted on TPC[3:0] and PCST[8:0] at 1/3 of the processor clock speed.

Chapter 10.   Electrical Characteristics

ESD Precautions: For handling precautions, see Section 1.1, Electrostatic Discharge (ESD), in
the chapter on General Safety Precautions and Usage Considerations.

10.1   Electrical Characteristics of TMPR4955AF

10.1.1   Absolute Maximum Ratings

V_{SS} = 0 V (GND)

Parameter	Symbol	Ratings	Unit
Supply voltage (for I/O)	VccIOMax	$-0.5 \sim 3.9$	V
Supply voltage (for internal)	VccIntMax	$-0.5 \sim 3.0$	V
Input voltage (∗1)	V_{IN}	$-0.5 \sim V_{CC}$ IO + 0.3	V
Storage Temperature	T_{STG}	$-65 \sim + 150$	°C

Note: The absolute maximum ratings are rated values which must not be exceeded during
operation, even for an instant. Any one of the ratings must not be exceeded. If any absolute
maximum rating is exceeded, a device may break down or its performance may be
degraded, causing it to catch fire or explode resulting in injury to the user. Thus, when
designing products which include this device, ensure that no absolute maximum rating
value will ever be exceeded.

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

10.1.2   Recommended Operating Conditions

V_{SS} = 0 V (GND)

Parameter	Symbol	Conditions	Min	Max	Unit
Supply Voltage (for I/O)	VccIO		3.1	3.5	V
Supply Voltage (for internal)	VccInt		1.4	1.6	V
Operating Case Temperature	T_C		0	+70	°C

Note: The recommended operating conditions for a device are operating conditions under which it
can be guaranteed that the device will operate as specified. If the device is used under
operating conditions other than the recommended operating conditions (supply voltage,
operating temperature range, specified AN and DC values etc.), malfunction may occur.
Thus, when designing products which include this device, ensure that the recommended
operating conditions for the device are always adhered to.

10.1.3 DC Characteristics

T_C = 0°C to 70°C, VccInt = 1.5 V ± 0.1 V, VccIO = 3.3 V ± 0.2 V

Parameter	Symbol	Conditions	Min	Typ.	Max	Units
Output High Voltage	V_{OH}	VccIO = 3.3 V, Vss = 0 V, I_{OH} = −4 mA	2.4			V
Output Low Voltage	V_{OL}	VccIO = 3.3 V, Vss = 0 V, 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 (Internal)	I_{CCInt}	VccIO = 3.5 V, VccInt = 1.6 V, MasterClock = 100 MHz, PClock = 200 MHz		350	550	mA
Operating Current (I/O)	I_{CCIO}	VccIO = 3.5 V, VccInt = 1.6 V, MasterClock = 100 MHz, PClock = 200 MHz		50	60	mA
Input Leakage	I_{LI}	Except (*3) port			±10	µA
Pull-up [*3]	Rinu		30	50	100	KΩ
Pull-down [*4]	Rind		30	50	100	KΩ
Output Leakage	I_{LO}				±20	µA
Input Capacitance	C_{IN}				10	pF
Output Capacitance	C_{OUT}				10	pF

(*1) V_{IL} Min = − 1.5 V for pulse width less than 10 ns.
(*2) Except for MasterClock input
(*3) Applies to Int(5:0)*, NMI*, RESET*, JTMS, JTCK, JTDI, TPC1 inputs
(*4) Applies to TRST*, RdRdy*, TPC3, TPC2 inputs

10.1.4 AC Characteristics

10.1.4.1 Clock Timing

T_C = 0°C to 70°C, VccInt = 1.5 V ± 0.1 V, VccIO = 3.3 V ± 0.2 V

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	100	MHz
Internal Operation Frequency	f_{PCK}		50	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 TMPR4955AF is only guaranteed with the Phase Lock Loop enabled.
(*2) All output timings assume a 25 pF capacitive load. Output timings should be derated where appropriate.

10.1.4.2 System Interface

T_C = 0°C to 70°C, VccInt = 1.5 V ± 0.1 V, VccIO = 3.3 V ± 0.2 V, BufSel = 100%

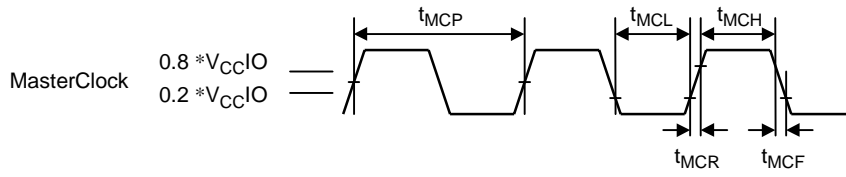| Parameter	Symbol	Min	Max	Units
Data Output [*1, 2, 3]	t_{DO}	1.0	6.5	ns
Data Setup [*3]	t_{DS}	3.5		ns
Data Hold [*3]	tDH	1.0		ns

(*1) Timings are measured from 1.5 V of the MasterClock to 1.5 V of signal.
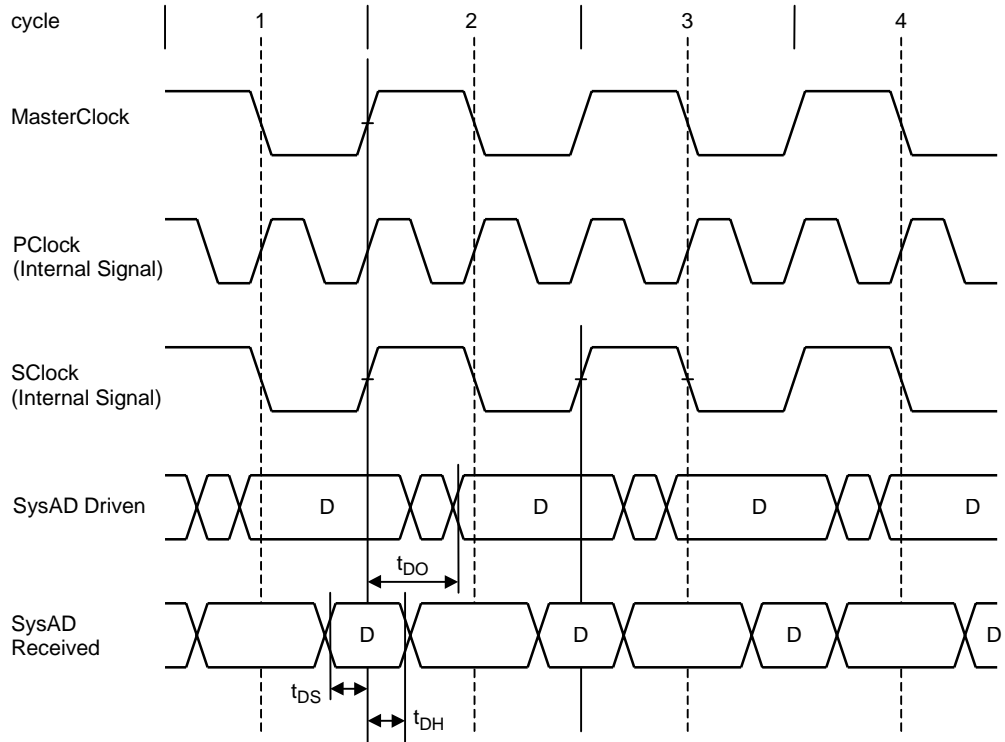(*2) Capacitive load for all output timings is 25 pF.
(*3) Data Output, Data Setup and Data Hold apply to all logic signals driven out of or driven into the TMPR4955AF on the system interface. Clocks are specified separately.
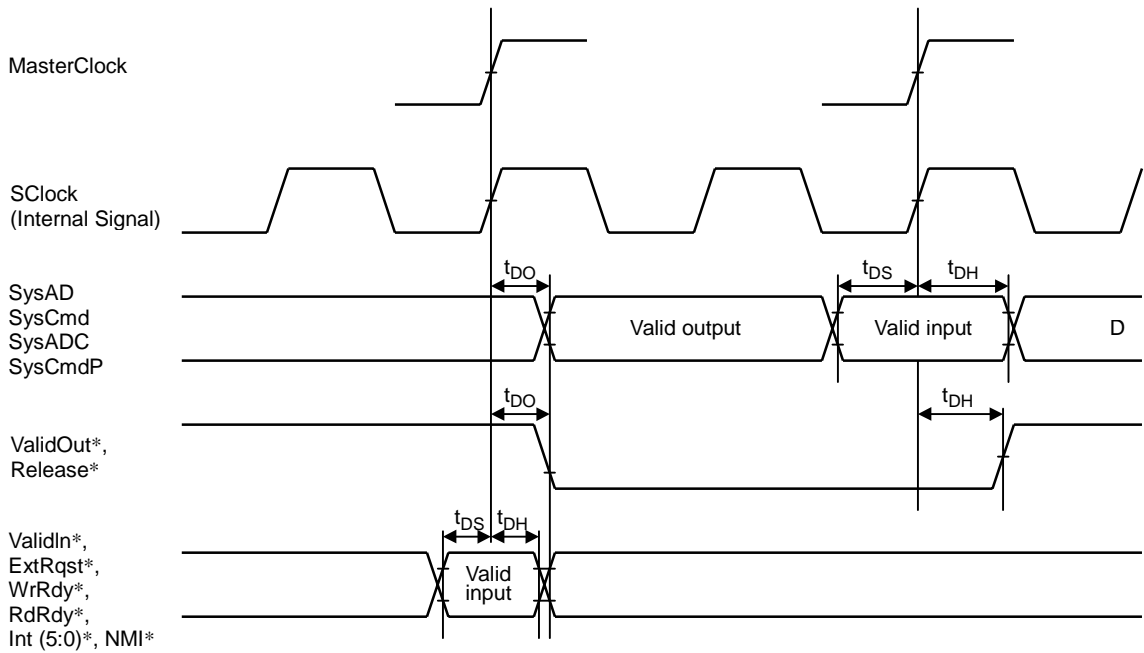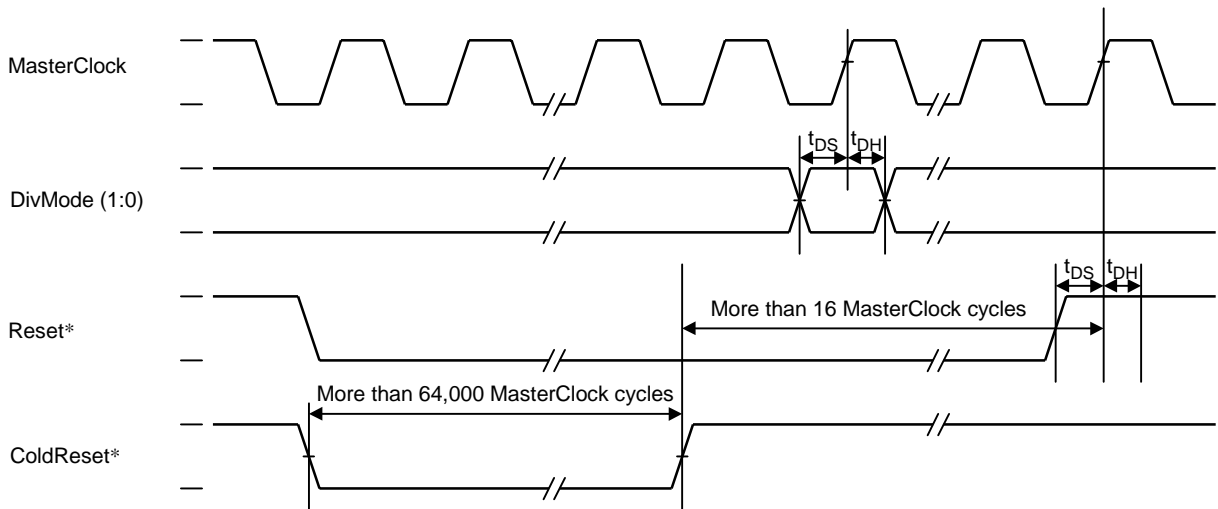
10.1.5 Timing Diagrams

10.1.5.1 Clock Timing

MasterClock $0.8 *V_{CC}IO$ $0.2 *V_{CC}IO$

t_{MCP} t_{MCL} t_{MCH} t_{MCR} t_{MCF}

10.1.5.2 PClock to SClock DIVISOR of 2

cycle 1 2 3 4

MasterClock

PClock (Internal Signal)

SClock (Internal Signal)

SysAD Driven D D D D

t_{DO}

SysAD Received D D D D

t_{DS} t_{DH}

10.1.5.3 System Interface Timing

MasterClock

SClock
(Internal Signal)

SysAD
SysCmd
SysADC
SysCmdP

t_{DO} t_{DS} t_{DH}

Valid output   Valid input   D

ValidOut*,
Release*

t_{DO} t_{DH}

ValidIn*,
ExtRqst*,
WrRdy*,
RdRdy*,
Int (5:0)*, NMI*

t_{DS} t_{DH}

Valid
input

10.1.5.4 Cold Reset Timing

MasterClock

DivMode (1:0)

t_{DS} t_{DH}

Reset*

More than 16 MasterClock cycles

t_{DS} t_{DH}

ColdReset*

More than 64,000 MasterClock cycles

10.1.5.5 Warm Reset Timing

MasterClock

Reset*

t_{DS}

t_{DS} t_{DH}

More than 16 MasterClock cycles

ColdReset*
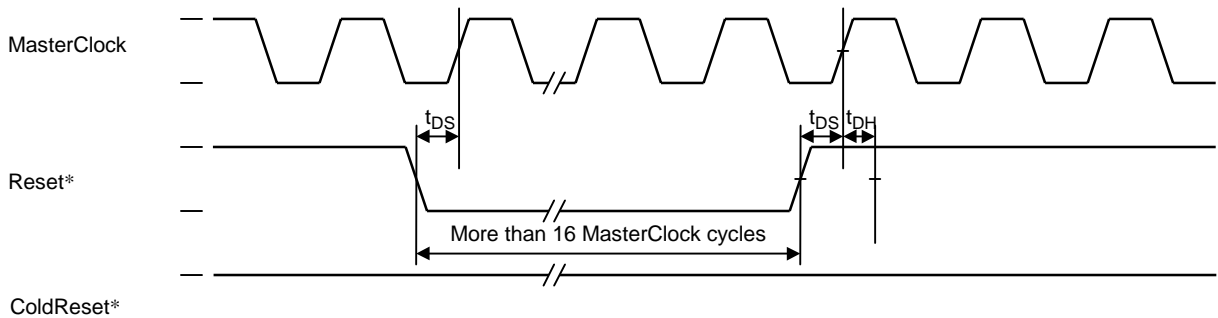
Chapter 11.   Package Dimension

QFP160-P-2828-0.65A   Unit: mm
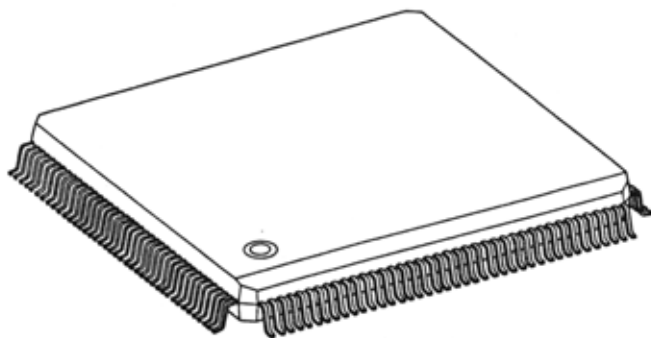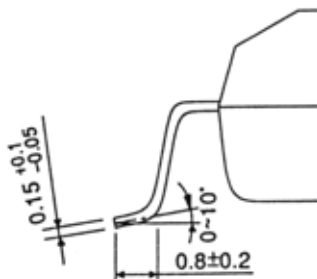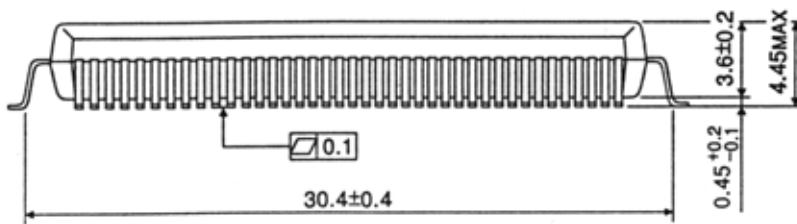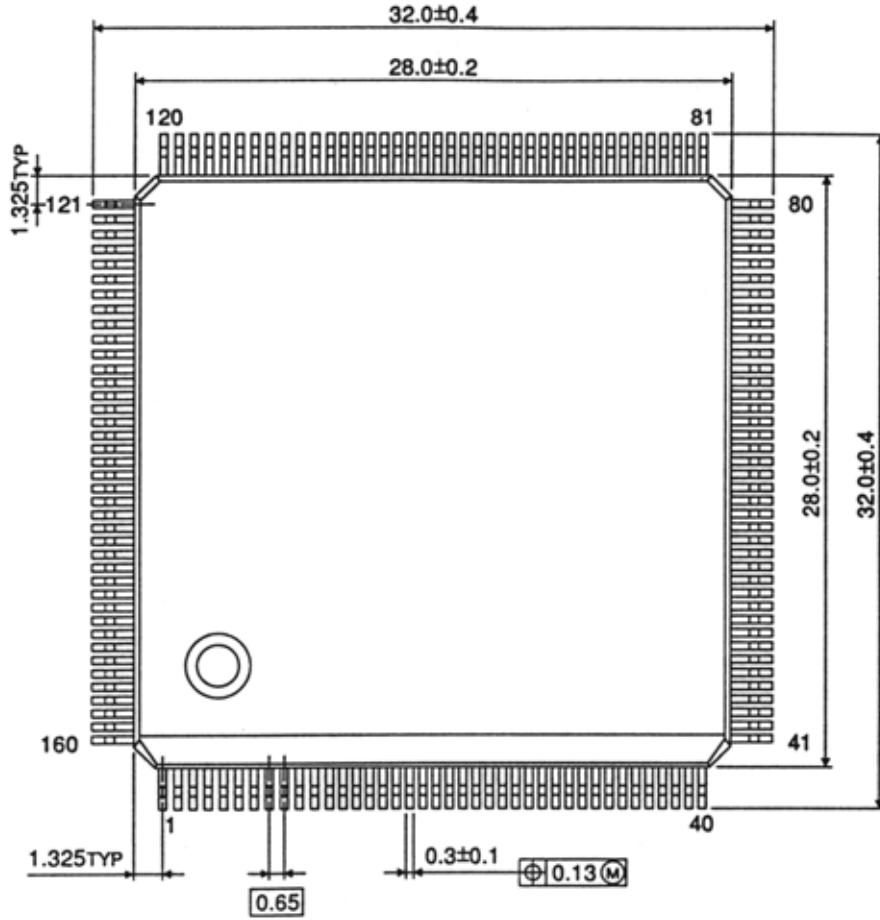
A. PLL Passive Components

The Phase Locked Loop circuit requires several passive components for proper operation, which are connected to VccPLL, and VssPLL, as illustrated in Figure A.1.
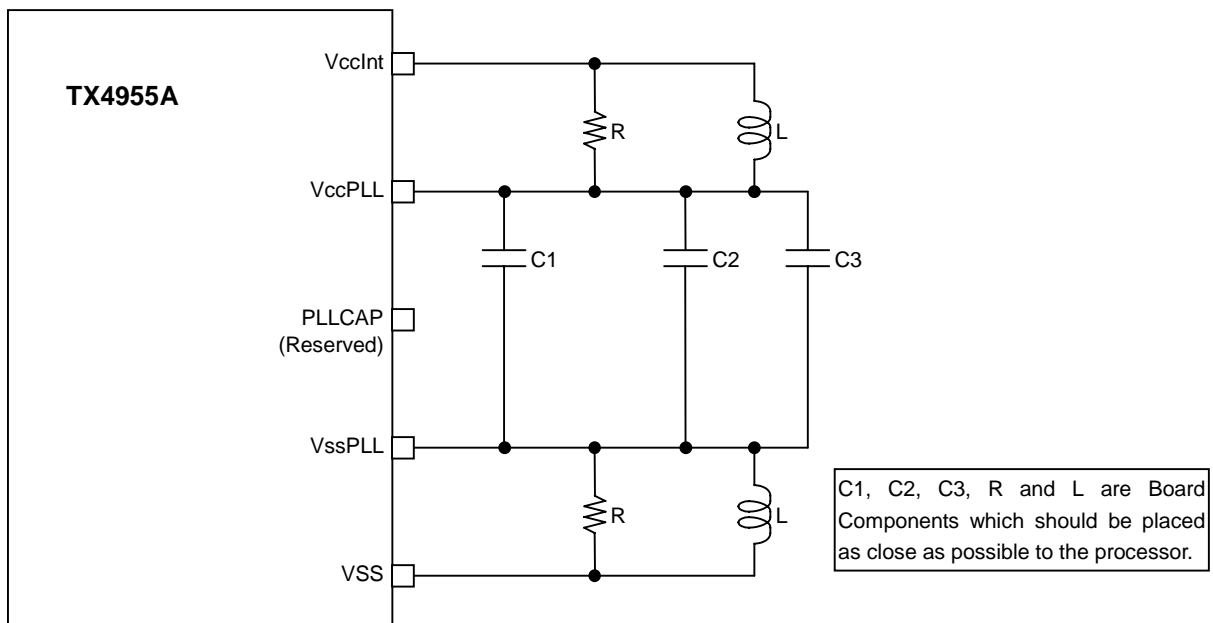
Figure A.1 PLL Recommended Circuit

Values:

- R = 5 Ω [*1]
- C1 = 1 nF [*1]
- C2 = 82 nF [*1]
- C3 = 10 μF [*1]
- VccInt = 1.5 V ± 0.1 V

The inductors (L) can be used as alternatives to the resistors (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 and ground (VccInt/Vss).

B. Movement parameter setting of a processor

A table explains movement parameter with a processor.

Item	Description
Single write protocol	These modes are selected by G2SConfig-Register. Write mode: 00 R4000 compatible — AWxx 01 reserved — reserved 10 Reissue write — AW 11 Pipeline write — AW
Writeback data rate	These modes are selected by G2SConfig-Register. Date rate: 0 AWWWWWWWW 1 AWxxWxxWxxWxxWxxWxxWxxWxx
Clock multiplier	These modes are selected by external pin (DivMode (1:0)) DivMode (1:0) / Master Clock / PClock / ratio 00 / 41.8 MHz/50 MHz / 167 MHz/200 MHz / 1:4 01 / 66.8 MHz/80 MHz / 167 MHz/200 MHz / 1:2.5 10 / 83.5 MHz/100 MHz / 167 MHz/200 MHz / 1:2 11 / 55.7 MHz/66 MHz / 167 MHz/200 MHz / 1:3
Endian set	Endian is selected by external pin (Endian). Indicates the initial setting of the endian during areset. 0: Little Endian 1: Big Endian
Timer Interrupt	Timer-Interrupt is selected by external pin (TintDis). It is Selection of Timer-Interrupt or Int5. 0: Timer-Interrupt enable Int5 is disable 1: Timer-Interrupt disable Int5 is enable
SysAD bus protocol type	SysAD bus protocol type is selected by external pin (MODE43*). It is selection of TX4300 type or R5000 type. 0: TX4300 type 1: R5000 type

Note1: A: Address (32 bit), W: Word (32 bit)

Note2: Initial set of data rate
   Single write: AWxx (R4000 compatible)
   Block write: AWWWWWWWW

C. Differences Between the TMPR4955 and the TMPR4955A

Product Name	TMPR4955F	TMPR4955AF
Power Supply: Core (incl. PLL)	2.5 V	1.5 V
I/O	3.3 V	3.3 V
Pin Assignment (No.2)	VccIO	BufSel1
(No.35)	VccIO	BufSel0
I/O Buffer Drive (Ratio)	1.0	Selectable from 0.5, 1.0 and 1.5

Note: BufSel (1:0) = 11 10 01 00
Output Drive Ratio = 100% 150% Reserved 50%

Signals influenced by BufSel(1:0)

Signal Name	I/O
SysAD(31:0)	I/O
SysCmd(8:0)	I/O
SysADC	I/O
SysCmdP	I/O
ValidOut*	O
Release*	O
HALTDOZE	O