Design Considerations for In-System Reprogrammable™ (ISR™) Programming of Cypress CPLDs

Introduction

The In-System Reprogrammable™ (ISR™) feature of Cypress Complex Programmable Logic Devices (CPLDs) enables reconfigurability of devices while soldered onto a system board. A standard IEEE 1149.1 (JTAG) interface is available to facilitate device configuration as well as boundary scan operations for straightforward prototyping, production program and test, field update, and general-purpose applications (note the FLASH370i™ family does not support boundary scan). The purpose of this application note is to detail applicable board design considerations and offer specific design guidance in order to simplify development of systems employing ISR capabilities.

The simplest method for integrating ISR functionality into a system is to interface directly to a Cypress ISR PC cable. Seamless PC control is provided by simply tying the ISR signals to an ISR header. While sections of this application note discuss specifics of the Cypress ISR PC cables, the design considerations pertaining to CPLD family or board level issues are implementation-generic and are directly applicable with any method of In-System Reprogrammable configuration, be it control via ISR PC cable, automated test equipment (ATE), or embedded microprocessor. The guidelines discussed herein are general and apply to any ISR interface methodology. For example, proper ISR signal layout and termination practices should be followed to ensure good signal integrity and reprogramming reliability. This involves awareness of device drive capabilities and transmission line effects pertaining to critical ISR signal distribution.

The In-System Reprogrammable feature is available in Cypress's Delta39K™, Quantum38K™, Ultra37000(V)™, and FLASH370i CPLD families, as well as Cypress's Programmable Serial Interface™ (PSI™) programmable PHY family. Differences between product families yield certain device-specific design considerations. For example, Delta39K, Quantum38K and PSI devices all offer 3.3V, 2.5V, and 1.8V I/O standard configurations for the ISR interface, whereas Ultra37000 offers 5V and 3.3V capability. Additionally, certain compact package options with the Ultra37000(V) and FLASH370i CPLDs allow dual-function ISR pins to switch between programming and regular I/O modes. These device-specific differences and their implications on ISR chain configuration are discussed herein.

This application note discusses transmission line effects that can arise from the ISR chain, which can cause signal integrity problems, and provides suggestions for trace layout to minimize these effects. Transmission line effects are inherent in the ISR programming set-up due to impedance mismatching between the ISR programming source (such as an ISR cable), traces and trace layout on the PCB, and ISR devices themselves. Properly terminated discrete buffers on the PCB, while not required in most cases, produce the highest quality waveforms and provide the best solution to controlling transmission line effects.

This note also discusses all issues related to programming and reprogramming the devices in-system (i.e., while they are soldered onto a printed circuit board). These issues include: an explanation of the available ISR programming cables and ISR connections, using the ISR programming pins for both functional logic and for programming, connecting ISR devices in a chain for programming as well as proper functional operation.

For the purpose of this note, "ISR device" refers to any ISR-capable device family. Also, "Ultra37000(V)" refers to both 5V Ultra37000 and 3.3V Ultra37000V. This design considerations note complements other Cypress ISR application notes, which offer introductory or device-specific information.

ISR Programming Pins

The programming pins for the Delta39K, Quantum38K, and PSI devices are standard JTAG signals: TDI, TDO, TMS, and TCK. Ultra37000(V) CPLDs include these pins, plus an additional control pin called JTAGen in some device packages. For the FLASH370i family, the same pins are named SDI, SDO, SMODE, SCLK, and ISREn respectively. The reason for using the standard JTAG naming convention for most ISR devices is that these families support Boundary Scan testing and are compliant with the IEEE 1149.1 (JTAG) standard. The programming interface for the FLASH370i devices also complies with most of the standard; however, these devices do not support boundary scan so the names were changed from the standard JTAG naming convention to prevent misleading the user. For the purposes of this note the JTAG pin nomenclature is used.

All ISR devices can be cascaded into a single chain for programming purposes so that one programming source (such as an ISR PC cable) can program all of the ISR devices on a board. The pins on an ISR device used for programming are: JTAGen, TDI, TDO, TMS, and TCK. Their names and functions are defined below.

TDI (SDI) – Data Input

During programming, this pin provides the serial data input to the device.

TDO (SDO) – Data Output

During programming, this pin provides the serial data output from the device.

TCK (SCLK) – Clock

During programming, this pin is the clock input. TDI and TMS are sampled on the rising edge of TCK, while TDO changes following the falling edge of TCK.

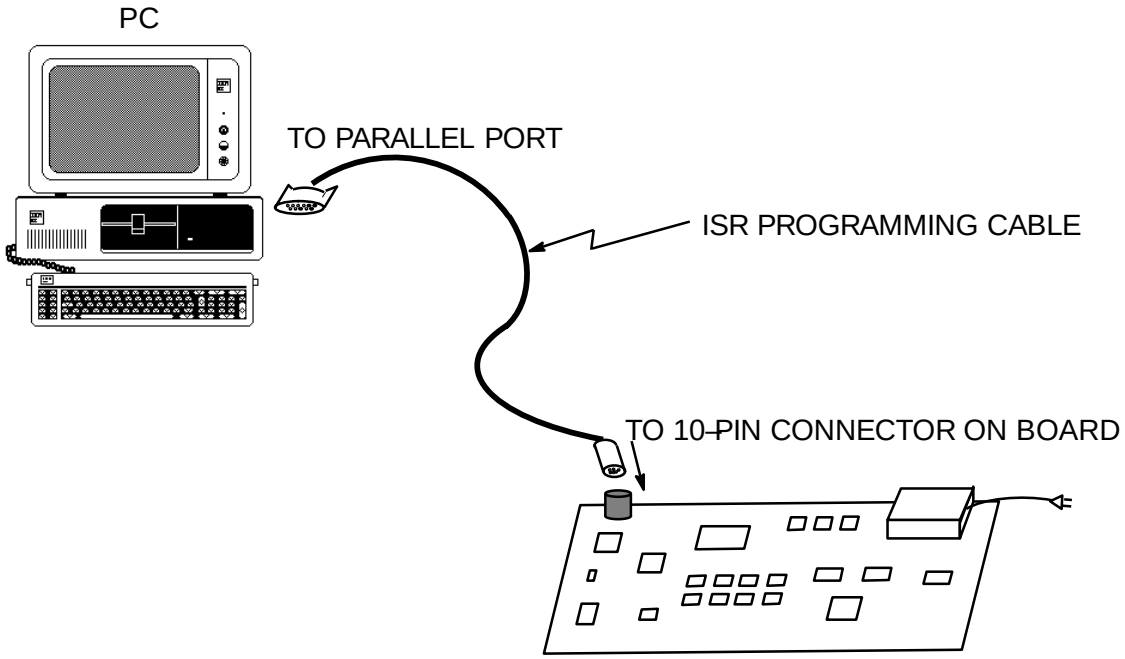


Figure 1. ISR Programming Set-up using a PC

TMS (SMODE) – Mode Control

During programming, this is the mode control input that directs the Test Access Port (TAP) controller state machine contained within the ISR interface on the device.

JTAGen (ISRen) – Enables the 4-pin JTAG Interface

This pin is present on packages of Ultra37000(V) or FLASH370i devices where the JTAG pins share their functionality with I/O pins. Dual-mode device details are found in Appendix A. For the Ultra37000(V), when the JTAGen pin is at a TTL HIGH level the JTAG pin functionality is selected. When it is at a TTL LOW level the I/O pin functionality is selected. For the FLASH370i devices the JTAG pin functionality is selected when it is at a supervoltage of 12.0V and the I/O pin functionality is selected when it is at a TTL LOW level. This functional difference in the JTAGen pin for these two ISR device families is important and is discussed later in this application note. All other ISR device families do not use JTAGen.

ISR* – Indicates Active ISR Operation Taking Place

This additional signal is not used by any CPLD device for programming. Rather, it is present on the ISR header to allow the programming source (e.g., ISR PC cable) to signal to the user's board when ISR operations are occurring. This can be useful for monitoring purposes or for user logic that controls any dual-function ISR pins of specific Ultra37000(V) or FLASH370i devices, as discussed in Appendix A. If on-board logic uses this signal, ISR* should be pulled-up to V_{CC} through a pull-up resistor on the circuit board to handle the case when no cable is attached.

The ISR Programming Cables

The simplest method to program ISR devices is to use a PC as shown in *Figure 1*. The ISR programming cable connects the parallel port of the PC to a cable header on the user's board where the ISR devices are soldered. The header on the user's board connects to traces that route to the JTAG pins on the ISR device itself. The ISR software runs on the PC and drives these pins on the board, through the cable and header, to program the devices with the appropriate bitstream.

Since each cable contains active components that buffer signals driven between the user's board and the PC, the cable must receive power from the user's board. It is recommended that the PC be powered-off prior to plugging in an ISR PC cable.

Table 1 shows the available Cypress ISR programming cables and their appropriate usage.

Table 1. Available ISR PC Programming Cables

Programming Cable	Part Number	Supported ISR Signal Voltages	Supported Devices
C3ISR	C3ISR.02 (rev. 0.02)	5V, 3.3V, 2.5V, 1.8V	Delta39K, Quantum38K, PSI, Ultra37000, Ultra37000V
UltraISR	37KISR.03 (rev. 0.03)	5V, 3.3V	Delta39K, Quantum38K, PSI, Ultra37000, Ultra37000V
ISRPCCABLE	ISRPCCABLE.03A (rev 0.03A)	5V (with 12V supervoltage on ISRen)	FLASH370i, Ultra37000 (Ultra37000V not supported)

The C3ISR cable offers the most flexible choice of programming voltage and device support. Any selection of 5V, 3.3V, 2.5V, or 1.8V can power the cable (V_{CC} of ISR connector pin), allowing TTL, LVTTL, LVCMOS, LVCMOS2, or LVCMOS18 ISR signal levels. The C3ISR cable is the recommended choice for ISR designs if the user does not need to program FLASH370i devices. A schematic of the C3ISR cable is shown in *Figure 2*.

The UltraISR PC cable contains a simple buffer capable of 5V and 3.3V signaling. The UltraISR cable schematic is shown in *Figure 3*. Both the C3ISR and UltraISRPCCABLE can be used to program all ISR families at 3.3V or 5V levels, with the exception of FLASH370i devices (since these require a 12V supervoltage generated by the cable on the ISRen pin).

The ISRPCCABLE, on the other hand, is the only cable with a built-in 5V-to-12V DC/DC converter to produce the 12V supervoltage to the ISRen pin required for FLASH370i programming, supplied by 5V from the user's board. The schematic for the ISRPCCABLE, used for programming the FLASH370i devices, is shown in *Figure 4*.

The ISRPCCABLE allows programming of the FLASH370i and Ultra37000 device families (but not Ultra37000V). The ISRPCCABLE (revision 0.03A or greater) can be used to program Ultra37000 devices because their JTAGen pin, if present on the selected package, is 12V tolerant. This means that the application of 12V on the Ultra37000's JTAGen pin is equivalent to placing a TTL HIGH value on the pin.

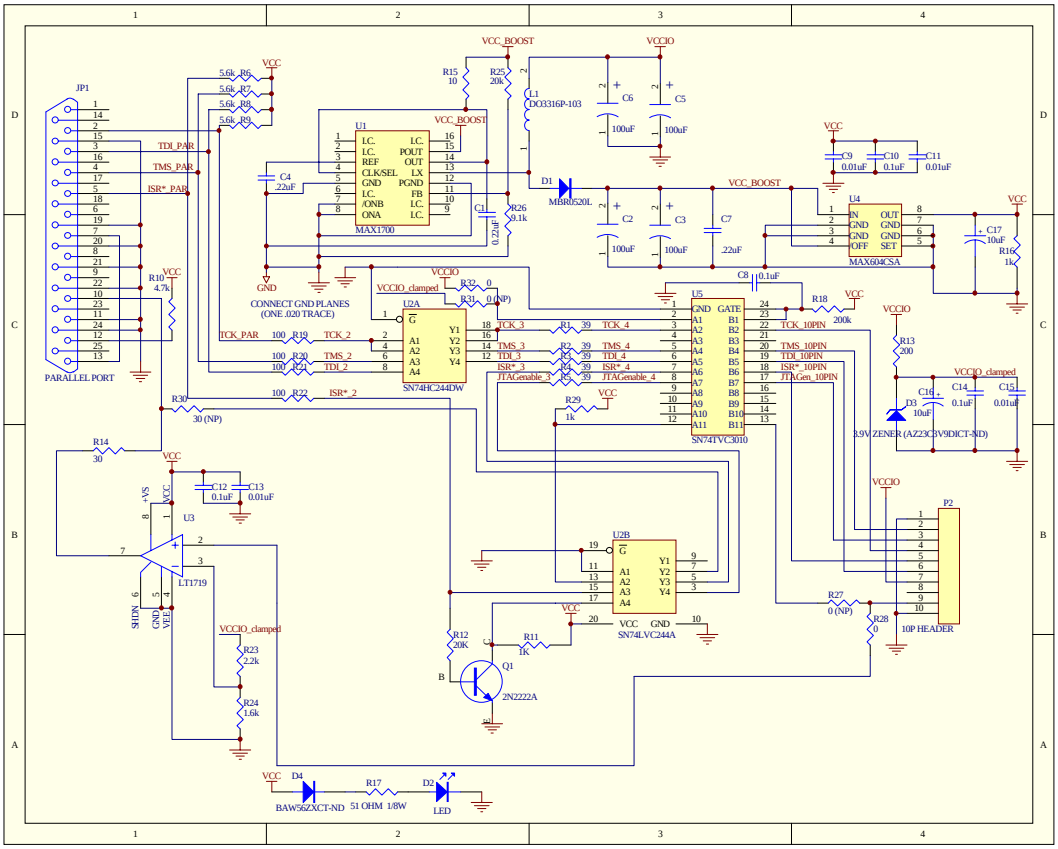


Figure 2. C3ISR Cable Schematic

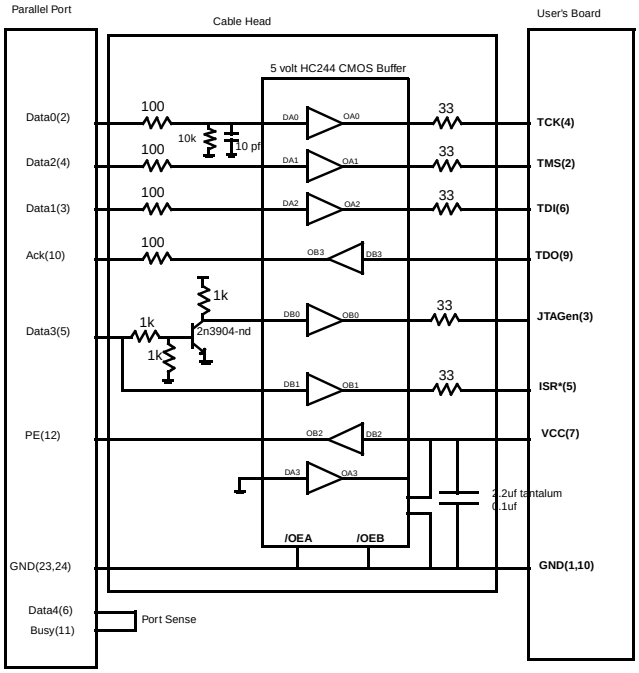


Figure 3. UltraISR Cable Schematic rev 0.03

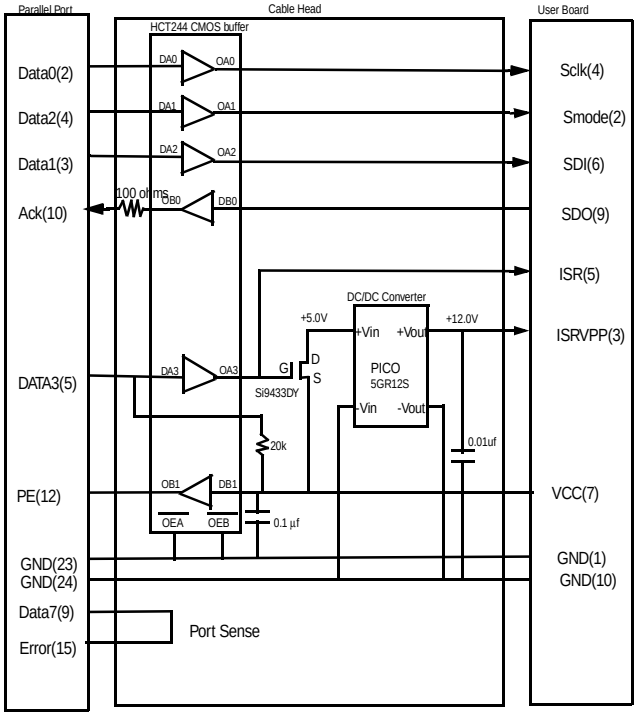


Figure 4. ISRPCCABLE Cable Schematic rev 0.03

Dimension of the ISR Cables

The C3ISR programming cable consists of a six-foot IEEE-1284 shielded cable attached to a blue transparent box containing the drivers and voltage conversion circuitry. A short one foot ribbon cable extending from this box attaches to the ISR connector on the user's board.

The UltraISR PC cable is a short one-foot ribbon cable with a small PC board mounted inside the parallel port connector casing.

The ISRPCCABLE is a six-foot cable with one foot as a detachable, flexible 10-wire ribbon cable. The ribbon cable section is connected to the 12V converter box at the end of a five-foot cable.

The user may need a parallel port extension cable since there may not be enough cable length from the parallel port on the PC to the circuit board. A high-performance IEEE-1284 cable extension is recommended for distances greater than six feet. All ISR programming cables plug into the female parallel port of the PC or the female end of an extension cable. However, signal integrity problems may arise with long parallel cable extensions.

Connecting the ISR Cables to the Circuit Board

To connect the cable to the system, a 10-pin, 2 x 5, boxed header male connector is mounted on the board. The ISR programming cable plugs into the boxed header. This boxed header connector has a small opening in the box on one side (the key) that allows the ISR programming cable to be plugged in one way only. The pins are on 0.100" centers. The length of each pin is 0.230", and the pin cross-section is 0.025" x 0.025". This boxed header connector is available as a straight-pin connector and as a right-angle connector. Additionally, an open header can be used. Part numbers for two compatible connectors are:

- DIGI-KEY part # S2012-05-ND
  (straight-pin connector)
- DIGI-KEY part # S2112-05-ND
  (right-angle connector)

All ISR programming cables provided by Cypress have a female end which plugs into these ISR connectors installed on the user's board. The position of the signal pins on the connector is shown in *Figure 5*. The orientation of this figure is such that the pin 1 location is the GND signal, which is located directly below the arrow on the female connector. The notch on the female connector is located near the ISR* signal, as shown. If the right-angle connector is used, make sure the raised key of the cable connector faces up so that it can be plugged into the boxed header without PCB interference.

PIN 1

TDO	V_{CC}	ISR*	JTAGen	GND
GND	NC	TDI	TCK	TMS

Figure 5. Layout of ISR Connector for PC Cable on Board Top View

To program a single ISR device using any ISR programming cable described here, route the TDI, TDO, TCK, TMS, and JTAGen pins from the cable connector to the TDI, TDO, TCK, TMS, and JTAGen (if present on the package) pins of the ISR device, respectively. If no ISR devices use JTAGen, this pin can float. Multiple devices can be programmed in a single ISR programming chain, explained later in this application note. For multiple devices in the chain, the TDI and TDO pins are connected in a serial chain such that the TDO of the first device in the chain connects to the TDI of the next device in the chain. The TDO of the last device in the chain then returns back to the 10-pin header TDO pin.

In addition to these programming pins, there is an additional signal available from the cable called ISR*. The purpose of this signal is to allow the user to monitor when ISR operations are in progress. If ISR* is a logic LOW, it indicates that JTAGen is asserted (at 12V supervoltage for FLASH370i devices) and ISR operations, such as programming, are in process on the board. If ISR* is a logic HIGH, it indicates JTAGen is 0V and no ISR operations are being performed. The ISR* pin is needed when using the JTAG/IO pins in both ISR (JTAG) and functional (IO) modes, as supported by certain packages of the Ultra37000 and FLASH370i families. It is used by on-board logic to determine when the JTAG signals should be enabled and other driving signals to the ISR device should be placed in the high-impedance state (three-stated). This is discussed in detail in Appendix A. When the ISR cable is connected, this signal is driven appropriately to a HIGH or LOW level. When the cable is disconnected, the ISR* signal must be pulled up on the circuit board, using a pull-up resistor to the V_{CC} pin of the ISR connector to indicate to the board that no ISR operation is in progress.

There are three other connection points on the cable and cable header: V_{CC}, GND, and NC. V_{CC} connects to the V_{CC} plane on the board containing the ISR devices. It supplies power to the active components within the ISR cable. This is necessary for any ISR programming cable to be able to buffer JTAG signals or translate voltage levels. On the UltraISR PC cable, V_{CC} simply supplies power to the buffer in the cable for the JTAG signals. On the C3ISR cable, V_{CC} from the user's board supplies power to the buffer and voltage translation logic of the cable. GND provides a common ground reference between the board and the ISR programming cable. NC is a "no connect" pin and is not used. For boards containing both 5V and 3.3V, it is recommended to connect 5V to the V_{CC} header pin instead of 3.3V. This doesn't apply for Delta39K, Quantum38K, or PSI devices, which cannot tolerate 5V.

Connecting the ISRPCCABLE to the Board

Since the ISRPCCABLE provides a high voltage (12V) to the device, it is recommended that the ISR cable be plugged into the PC and the customer's board before power is applied to the board. (This is not as important for the C3ISR or UltraISR PC cables since there are no 12V signals generated in the cables.) Once the cable is connected, the user can power up the board. This assures a normal slew rate on the ISREn pin for all possible conditions. It is also recommended that the user run the ISR software after the cable is plugged into the user's board and PC parallel port. This allows the software to correctly set the ISR pins to their appropriate initialized state on the parallel port of the PC. All of the ISR signals are buffered in the ISR cables and are permanently enabled.

With this configuration, it is recommended to place a 10-nF capacitor located at the 10-pin header connector on the circuit board from the ISREn pin to ground. If the ISR cable is hot-socket connected to the user's board, which is not recommended, then this capacitor insures that a proper, slow ramp up to 12V is applied to the devices to be programmed under all operating conditions, with no risk of damage to the ISREn pin. Since the ISR cable could easily be hot-socket connected by accident, it is advisable and simple to incorporate this decoupling capacitor. Again, this only applies if the ISRPCCABLE is used.

Simple ISR Device Cascading

You can cascade many ISR devices in a system. That is, you can daisy-chain the devices together and connect their programming pins in such a way that all devices can be programmed from a single connection to the ISR programming source.

To do this, simply tie all of the TCK and TMS pins (and JTAGen if present) of each device to the corresponding pins of the ISR connector. You then connect the TDI pin from the connector to the TDI pin of the first device in the chain, then connect the TDO output of that device to the TDI input of the next device in the chain, and so forth, until you finally connect the TDO output of the last device in the chain to the TDO pin of the cable connector or other programming source (see *Figure 6*).

Cascading With Other IEEE 1149.1-Compliant Devices

Other vendor IEEE 1149.1-compliant devices can be included in the chain. The ISR programming software will load the non-Cypress device's instruction register with the proper BYPASS instruction.

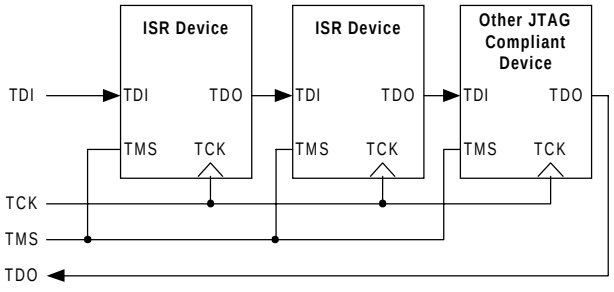


Figure 6. Simple Cascading of ISR Devices

Cascading Ultra37000 and FLASH370i ISR Devices

In addition to the optional external circuitry required to take advantage of dual JTAG/IO functions per dual-mode ISR device, the JTAGen pin has slightly different functionality between the FLASH370i and the Ultra37000 families. For details on Ultra37000 and FLASH370i cascading, see Appendix B.

Board Layout Considerations

Transmission line effects are a critical design consideration to ensure suitable signal integrity and reliable In-System Reprogrammability performance.

Relevant design considerations discussed in this section include: key transmission line effects to minimize, recommended termination practices, signal layout, and on-board buffering guidance.

Condition for Terminating Transmission Line

Not all digital circuits require termination measures. For ISR programming applications, however, transmission line effects are most often significant. Transmission line effects take the form of unwanted voltage reflections and ringing under certain conditions. These effects are caused by impedance mismatch between source drivers, traces, receiver loads, and any discontinuity between them. For instance, low-impedance outputs driving high-impedance inputs will yield a certain reflection coefficient. The size of the voltage reflection is directly proportional to the impedance mismatch at a particular node. These inadvertent reflections can interfere at device receivers, thereby degrading system operation.

The classical way of stating the condition for a voltage reflection to occur is when the signal rise time is less than or equal to the round-trip (two-way) propagation delay of the line. In equation form, transmission line effects are prominent when:

$$L \geq \frac{t_r}{2 \bullet t_{pdL}} \qquad \text{Eq. 1}$$

where L is the length of PCB trace, t_r is the rise time of the signal at the source, and t_{pdL} is the one-way propagation delay of the line per unit length.

It is important to note that transmission line effects depend on signal rise and fall times rather then on signaling rate. For the Cypress ISR programming cables, for example, even though each clock cycle can span several hundred nanoseconds, the signal transition edge rates of the cable drivers are fast, in the order of 5 to 10 ns. Likewise, typical CPLD rise/fall times are on the order of a few nanoseconds. Transmission line effects are therefore inevitable; however, certain layout practices can greatly reduce their harmful effects.

These effects can occur on any of the four ISR signals but are of special concern for the clock signal TCK. This is because there is potentially plenty of time, many hundreds of nanoseconds, for the data signals to settle before the next rising clock edge. Further, the electrical characteristics of these other signals are critical only at the active edge of TCK. Particular care should be taken with the clock signal trace layout.

Thus, even for relatively short trace lengths, transmission line effects can be prominent at fast edge rates. A discussion of transmission line effects is necessary to understand how to best design the ISR chain. This discussion also applies to any method of ISR programming occurring at higher frequency, such as by an on-board microprocessor. Additional details can be found in the references located at the end of this application note.

Transmission Line Effects

There are two significant transmission line effects that can be encountered. They are illustrated in *Figure 7* and *Figure 8*. These effects are overshoots with ringing after the edge transition and notching of the signal during the edge transition itself.

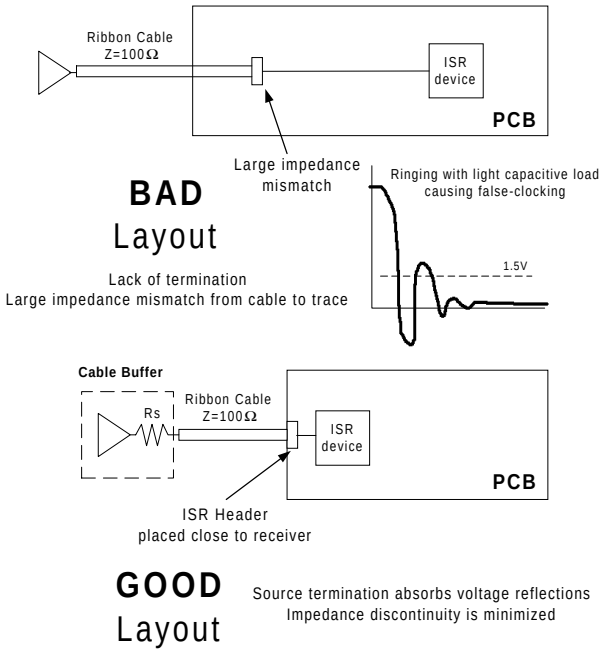


Figure 7. Transmission Line Effect—Ringing

The first effect, overshooting and ringing, is worst-case with very light capacitive load. Severe signal bounce can cross the receiver's input threshold, thereby causing false-clocking. While the ringing is not typically big enough to cause failure, it can be minimized by observing proper termination techniques.

The recommended interconnect scheme for a single load connected to an ISR header is shown in *Figure 7*. In the "bad layout" example, an external buffer drives a long transmission line with no termination used. The impedance mismatch from cable to PCB trace, combined with the lack of termination, creates voltage reflections and ringing. In the "good layout" example, as implemented by any Cypress ISR programming cable, source termination acts to absorb reflections. Placing the CPLD close to the ISR header will minimize the effect of the impedance mismatch between the ribbon cable and the device. Source termination is already employed in the cable, so no other termination components are necessary.

Since all ISR programming cables use source termination to absorb voltage reflections from the line, any additional type of end termination is not recommended. The effect of a single load pull-down resistor would degrade the V_{OH} output high level due to the voltage divider effect of the termination resistor and the buffer plus R_S resistance. A parallel termination load, which employs a resistor connection to V_{CC} and to ground, would degrade the V_{OL} of the output buffer due to the voltage divider action of the resistor to V_{CC} and the buffer plus R_S resistance to ground.

The other transmission line effect that can cause false-clocking is a notching, or glitching, of the clock transition while in the middle of the transition itself. The notch is evident on both edges of the clock. The notch occurs because of a voltage

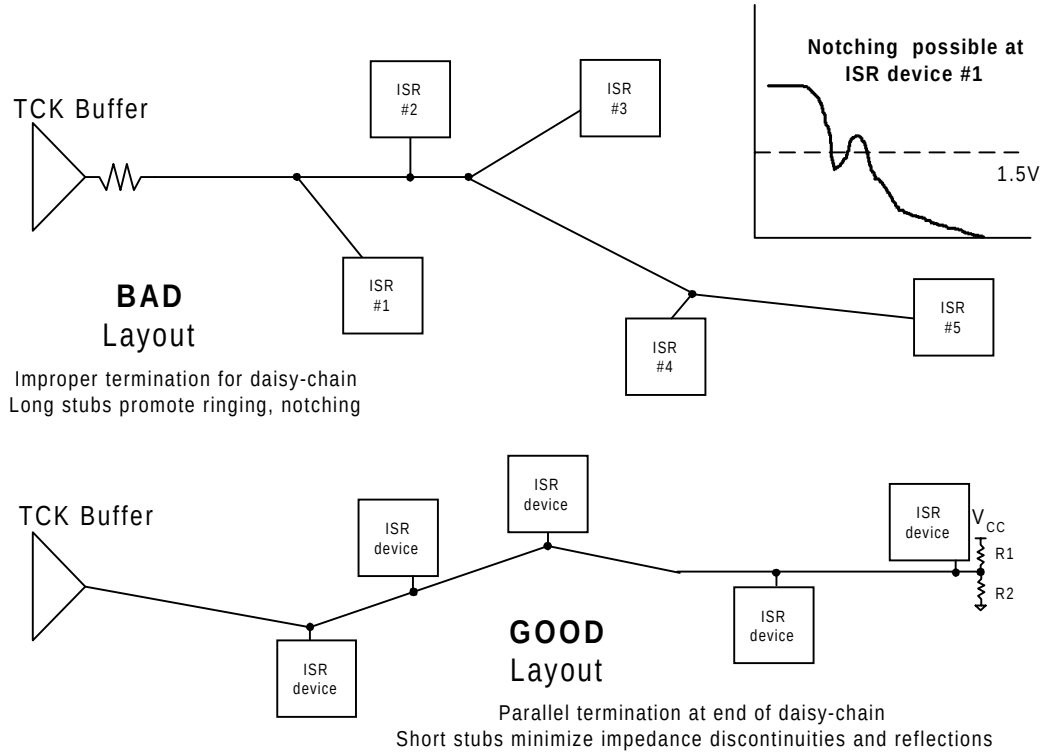


Figure 8. Transmission Line Effect—Notching

divider effect between the source impedance of the buffer and the characteristic impedance of the transmission line Z_O. As illustrated in the "bad layout" above, the long stubs and associated impedance mismatches cause reflections at each node. As the initial voltage wavefront propagates to the end of the transmission line, reflections from adjacent nodes can affect the voltage waveform at earlier receivers by creating notching. If the notch resides near the trip point of the ISR device, (1.5V for both Delta39K LVTTL levels and Ultra37000 devices), then it could result in a false-clock scenario resulting in ISR operation failure. This effect more typically occurs with multiple devices in the ISR chain because of differing clock line trace lengths and improper trace layout.

The daisy chain set-up shown in the "bad layout" example of *Figure 8* is not recommended. With source termination, as used in all ISR programming cables, all load devices should be lumped at the end of the line. As this is not practical for more than one ISR device, a buffering scheme must be adopted. A better layout practice is shown in the "good layout" example, yet this layout may still be susceptible to ringing or notching depending on the precise layout. Recall that since all ISR programming cables employ series termination, an on-board buffer is necessary. Other schemes for providing the best signal integrity are described in detail below.

TCK Clock Layout

The layout of the clock shown in *Figure 9* can make a big difference in reducing the transmission line effect of notching. If the clock trace is laid out exactly as shown in *Figure 8*, then the first device can experience a notch of duration 2T, where T is the transmission line delay from device #1 to the end of the chain at device #5. Making the clock trace the same length to device #1 as device #5, as shown in *Figure 9*, can remove this notching effect. In general it is good to avoid stubs on the clock line to minimize this effect.

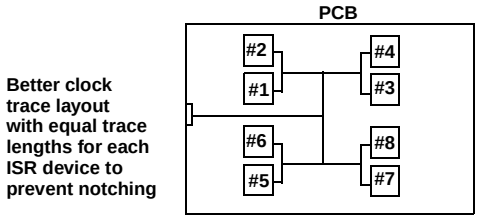


Figure 9. Clock Trace Layout

While this scheme can be effective, its implementation on a PCB may not be practical. This is due to the fact that the split traces must be impedance matched to the feed wire in order to prevent reflections and ringing between junctions.

An example implementation is shown in *Figure 10*. Here, the split trace widths are scaled smaller such that the parallel combination of the split traces matches with the characteristic impedance of the feed trace. The example shows a clock tree distributed to four ISR devices, each end terminated to a proper Thevenin termination voltage. The characteristic impedance of the feed wire, Z_O, matches with the four thin scaled $4*Z_O$ traces in parallel. While this configuration results in optimal signal integrity, it is little used on PCBs. This is due

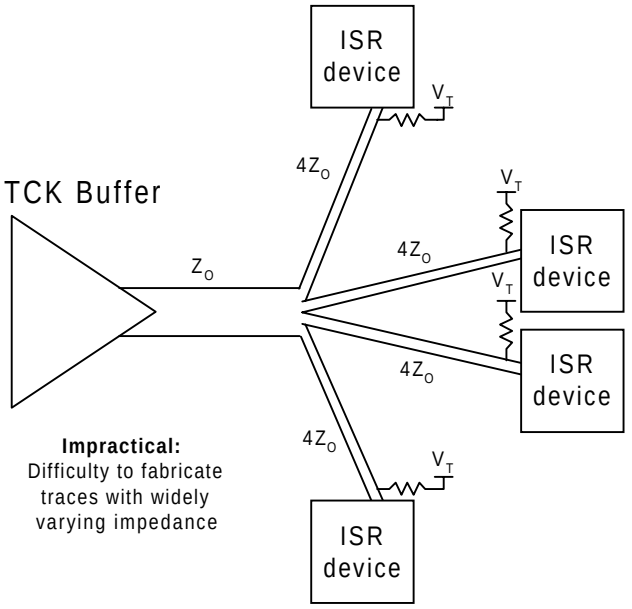


Figure 10. Clock Tree Trace Layout Implementation

to the manufacturing difficulty of fabricating traces with widely varying impedance on the same board.

In lieu of this technique, it is recommended use the buffering scheme described in the following section for best signal quality.

Adding Extra Buffering on the PCB

The best signal integrity for the clock line is achieved by incorporating buffering on the board as shown in *Figure 11*. This buffering, with proper termination, can guard more effectively against both of the transmission line effects previously described. The buffer should be placed close to the 10-pin header connector so that the termination resistor can match the cable characteristic impedance and prevent reflection at the buffer input. The on-board buffer can provide separate buffering for each ISR device clock line. The series resistor, placed close to the buffer output, can match the characteristic impedance of the trace on the board and prevent reflections at the buffer output. In *Figure 11* it is not crucial that the clock traces have the same length for each device clock since there is no danger of timing problems from one ISR device to the next. This is because the data output from one ISR device does not propagate to the next device until the falling edge of the clock as defined in the IEEE 1149.1 specification.

An HC buffer such as the HC244 for 5V V_{CC}, or an LVT244 for 3.3V V_{CC}, is recommended for on-board buffering. These are selected because they provide good drive capability and input noise margin. An FC buffer should be avoided since the faster edge rates will worsen transmission line effects. The source termination resistor is selected to match the source impedance to the characteristic impedance of the trace. In this way, the output resistance of the buffer plus the source termination resistor R_S equals Z_O. A carefully matched source prevents voltage reflections towards the load. It is important to note that there is often no best value for R_S due to uneven impedances when driving high and low for logic families such as TTL. This trade-off can still produce acceptable signal integrity.

Extra buffering with termination is recommended wherever the impedance of the trace changes greatly, such as at the connection between a PCB and an add-on card. If large numbers of ISR devices are required in the ISR programming chain, the clock buffer outputs can feed a clock tree layout, or can alternatively feed a daisy chain of cascaded ISR devices (replacing source termination with end termination). This combines the configuration shown in *Figure 11* with one of *Figure 9* or *Figure 8*.

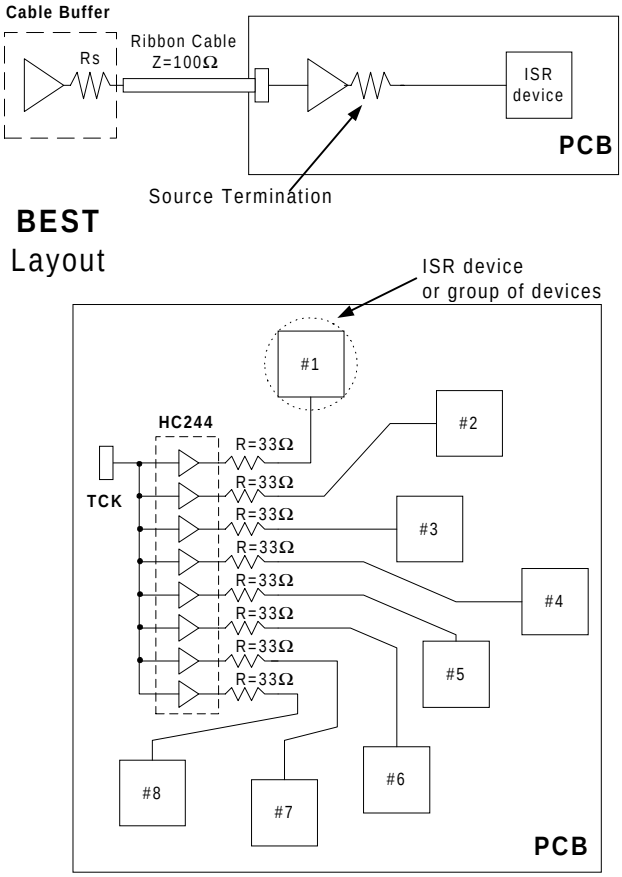


Figure 11. Buffering with Termination

Valid ISR Programming Voltages

Only the Delta39K, Quantum38K, and PSI families of programmable devices support any choice of programming voltage on the ISR signals (V_{CCJTAG}) from 3.3V, 2.5V, and 1.8V. All are provided by the C3ISR programming cable. The recommended voltage is 3.3V, device permitting.

Ultra37000 and FLASH370i devices permit 5V and 3.3V levels (with appropriate V_{CCIO} level). The FLASH370i device family is the only one that requires a 12V supervoltage on ISREn for programming. To maintain compatibility, Ultra37000 devices (except for Ultra37000V) are designed tolerant to 12V on the equivalent JTAGen pin.

Device-Specific ISR Design Considerations

The In-System Reprogammability feature is available on a wide range of Cypress programmable devices. While the board layout recommendations discussed herein apply to all ISR devices, there are some notable ISR design considerations that are distinct for each family.

Board-level design issues addressed in this section include: the state of ISR programming pins when floating, bus-hold pin differences between the FLASH370i and the Ultra37000, and supported I/O levels for JTAG signaling.

The Delta39K, Quantum38K, PSI, and Ultra37000 ISR device families incorporate internal weak pull-ups on the JTAG pins TDI and TMS as required by the IEEE 1149.1 specification. This is to ensure that, if a solder fault results in an open circuit on a JTAG pin, the internal TAP controller in the device will enter the predictable, safe Test-Logic-Reset state. The FLASH370i family has a bus-hold structure on these pins. While still JTAG compliant, this family alone does not support boundary scan capability.

It is important to consider that noise from a large number of I/Os switching simultaneously during boundary scan operations can affect functionality. Therefore, it is highly recommended that these boundary scan tests be done in a relatively low noise environment.

Delta39K/Quantum38K/PSI Families

The Delta39K, Quantum38K and PSI families permit flexible signal levels on the ISR pins, including 3.3V, 2.5V, and 1.8V, supporting LVTTL, LVCMOS, LVCMOS2, and LVCMOS18 I/O standards. Each voltage can be selected by applying the desired target voltage to the V_{CCJTAG} pin. The recommended voltage for JTAG port signals is 3.3V. These devices have single-function dedicated JTAG pins.

Since these families lack 5V tolerance, they cannot directly be driven by 5V levels without additional board components. Consequently, Delta39K, Quantum38K, and PSI devices cannot directly follow an Ultra37000 or a FLASH370i device driving 5V within a scan chain cascade, since TDI will violate input voltage specs. Board-level solutions for interfacing Delta39K, Quantum38K or PSI I/Os to 5V is described in the Cypress application note: “Interfacing Delta39K and Quantum38K CPLDs to 5V Devices.”

These families, based on volatile SRAM technology, require configuration at power-up prior to normal operation. I/Os are enabled only after configuration is complete. Up to this time, I/Os are three-stated and can be subjected to live signals. For the case of a self-boot solution, the initial data stored on the internal FLASH ensures I/Os three-state after device configuration completes. For volatile devices requiring external configuration, they will continually restart configuration cycles until either a valid configuration bitstream is downloaded from the boot EEPROM or JTAG instructions program the part. More information on Delta39K configuration can be found in the Cypress application note: “Configuring Delta39K/Quantum38K.”

Ultra37000 Family

Ultra37000 devices support 5V or 3.3V I/Os (including the ISR pins) since the I/O power supply, V_{CCIO}, is split from the core power supply. Ultra37000V devices support only 3.3V, yet have I/Os which are tolerant to 5V inputs.

Ultra37000 CPLDs differ from Delta39K in that certain device packages offer dual-function pins that operate as ISR signals in programming mode or as regular input/output in I/O mode. These are typically smaller packages with a restricted pin count. For these dual-function devices, the pin called JTAGen controls the multiplex between modes. When programming (JTAGen is HIGH), JTAGen enables the ISR interface for dual-mode devices. When the JTAGen signal disables the ISR interface (JTAGen is LOW), the ISR device will start driving some of its output pins, based upon the functionality of the design.

The user can configure a dual-function pin as an input, output, or bidirectional input/output. This requires additional external logic to mediate between JTAG signals and I/O signals. This external logic requirement is discussed in detail in Appendix A.

While specific FLASH370i devices also support dual-mode pins, there are small differences between ISRen for FLASH370i and JTAGen for Ultra37000 to take into consideration. Simple cascading of single- and dual-function devices of both families is discussed in *Appendix B.*

State of ISR Pins When the ISR Pins Float for Ultra37000 Devices

The ISR pins can be floating if, for example, a programming cable is not attached to the on-board ISR header. The only ISR programming difference between the Ultra37000 and FLASH370i family is the connection of the bus-hold latches on the ISR interface pins. Specifically, for Ultra37000 CPLDs the bus-hold latches are disconnected from the JTAG pins TCK, TMS, TDI, and TDO when the JTAGen pin is HIGH, thereby enabling the JTAG port. The bus-hold latches are connected when the JTAGen pin is LOW, thereby disabling the JTAG port. For the single-function devices there is no JTAGen pin and the ISR interface is permanently enabled; therefore, the bus-hold latches for JTAG pins are permanently disabled. These differences allow the Ultra37000 family to support JTAG Boundary Scan testing. (Bus-hold latches could have caused significant DC loading on the JTAG drivers of TCK and TMS depending on the source impedance of the drivers and the number of devices connected in the ISR chain. This could occur because these signals, TCK and TMS, are connected in parallel to all the devices in the ISR chain.)

An additional difference is that internal pull-up resistors are enabled on the TDI and TMS JTAG pins when the ISR interface is enabled. This change allows conformance to the IEEE 1149.1 specification and is necessary to place the Test Access Port (TAP) controller of a JTAG device in a known benign state such as Test-Logic-Reset if solder open faults occur in the ISR chain. The bus-hold latch is still permanently enabled on the JTAGen pin and powers up in the HIGH state. To determine whether external resistors are needed we once again must consider the single- and dual-function mode cases.

For single-function mode devices the only pin that needs an external resistor pull-down is the TCK pin since there are already internal pull-ups for TDI and TMS and the TDO pin is a dedicated output pin.

For dual-function mode devices operating in single-function mode or dual-function mode no external pull-ups are necessary since the bus-hold latches are reconnected to the ISR pins once the ISR interface is disabled.

The value of this pull-down resistor for TCK is not crucial since the external JTAG pin driver simply has to be strong enough to overpower the resistor. Typical values are 10 kΩ or 4.7kΩ.

FLASH370i Family

Like the Ultra37000 family, FLASH370i CPLDs have a separate I/O power supply, V_{CCIO}, which can support 5V or 3.3V levels. Only the ISRPCCABLE can program an ISR chain containing FLASH370i devices. This is because this cable alone produces the required 12V supervoltage on the ISRen pin required for device programming. This cable, however, does not support 3.3V ISR signaling so it cannot be used with any 3.3V ISR device like the Delta39K.

Smaller FLASH370i packages contain pins which offer dual-mode functionality between ISR programming mode and normal I/O mode, similar to the Ultra37000 family. The necessary external logic to support dual functionality is detailed in *Appendix A.* Specific design considerations with cascading single- and dual-function Ultra37000 and FLASH370i devices are found in *Appendix B.*

Additional FLASH370i device and board-level design considerations are found in *Appendix C.*

State of General Cypress ISR Device I/Os at Power-Up

When ISR devices are shipped from Cypress, the devices have already been programmed, erased, and programmed again as part of the testing process. They will not, therefore, be blank when they first come out of the tube. They will, however, be programmed such that all of the I/Os are three-stated after power-up. Furthermore, all I/Os (except TDO) are three-stated during device programming. This allows soldering of ISR devices directly onto a user's board without having to erase them first. It allows the user to power-up a board and program the ISR devices on it without worrying whether their initial, non-blank state will cause any problems such as output contention with other devices on the board.

The ISR programming procedure is to take ISR devices directly from the tube, solder them onto the board, connect the programming source (such as the ISR programming cable attached to a PC), turn on the power to the board, and then program the devices for the first time. Since many of the I/Os on the ISR device(s) are undoubtedly inputs, other devices on the board could be driving those pins immediately upon powering up the system. By having all of the ISR I/Os initially programmed to be three-stated, and by guaranteeing the same three-state during ISR programming, you are assured that the ISR device will not also drive those pins. This prevents bus contention, and prevents the ISR devices or other devices on the board from being damaged. ISR devices can be programmed for the first time in-system without fear that other board components driving the CPLD will negatively effect operation.

As previously discussed, all ISR device I/Os will initially three-state after first power-up regardless of whether the CPLD contains volatile or non-volatile configuration bits.

Summary

The In-System Reprogrammable feature of Cypress ISR devices provides the critical capability to manipulate programmable device configuration without the need to desolder components from a user's board. With a good understanding of ISR design considerations involved and adherence to the recommended layout and termination practices, a printed circuit board designer can implement a reliable method for device reconfiguration or in-system test using boundary scan.

The Cypress ISR PC programming cables provide the simplest method to build In-System Reprogrammability of programmable devices into systems. In-System Reprogrammability (ISR) of a programmable device has several benefits. It allows engineering development and debugging to occur without having to socket the ISR devices and without having to remove them and reprogram them in a device programmer. This saves time regardless of the package type used. ISR is especially valuable when fine-pitch packages like TQFPs and FBGAs are used. This not only saves time, it can also avoid the high likelihood of bending leads on very fine-leaded devices. Also, by allowing soldering of TQFP packages directly onto a board without sockets, it helps to avoid spending time simply checking device-to-socket-lead connections during debugging. ISR also allows for designs which can be reconfigured in the field, either by a software update or by other input from the system. The superior routability and flexible architecture of the Cypress ISR CPLDs enhance the value of all of these benefits greatly by allowing design changes to be made during prototyping, debugging, or field operation and still successfully route to the already-defined pinout, even on designs that utilize most or all of the device's resources.

This application note explains the available Cypress programming cables and the signals they use, and also covers many design techniques and considerations that show how to easily use the capabilities of ISR devices. These include: description of the logic needed when using the dual-function pins on applicable ISR devices, connecting ISR devices in a programming chain, and ISR differences between the programmable logic device families.

Designing a daisy-chain of ISR devices is straightforward provided that the designer follows the layout guidelines set forth in this document. By following the recommended layout practices, board-level problems will be eliminated and a highly reliable programming method results.

References

1. Cypress Application Note. "System Design Considerations When Using Cypress CMOS Circuits," 1993.
2. IEEE Std 1149.1-1990 Test Access Port and Boundary Scan Architecture, IEEE Computer Society, May 1990.
3. Johnson, H. and Graham M. "High-Speed Digital Design," Prentice-Hall, Inc., 1993.

Appendix A. Dual-Function Device Considerations for Ultra37000 and FLASH370i CPLDs

Only the Ultra37000 and FLASH370i CPLD families support dual-mode ISR programming pins, whereby compact packages offering these pins can function in JTAG mode or in normal I/O mode. It is suggested that the following three application notes, “An Introduction to In-System Reprogramming with the Ultra37000,” “An Introduction to In-System Reprogramming with the FLASH370i,” and “Understanding Bus-Hold—A Feature of Cypress CPLDs,” be read along with this application note to become familiar with these product families. They give a complete listing of all single- and dual-function mode devices for all members of the FLASH370i or Ultra37000 families

Single-/Dual-Function Programming Pins

The next portion of this note explains in detail how to use the IO function on the JTAG/IO pins. To address this issue, we categorize designs into three types: designs using devices with single-function pins; designs using devices with dual-function pins used in single-function mode; and designs using devices with dual-function pins in dual-function mode. The single-function/dual-function names refer to the four ISR pins and whether they share their functionality with IO pins. Single-function mode refers to the ISR pins functioning as ISR pins only. Dual-function mode refers to the ISR pins functioning in ISR mode during ISR operations as well as IO mode when the device is operating normally in the board. The three cases are explained further below.

Single-Function Pins/Single-Function Mode

Some of the ISR devices have pinout/package combinations such that the pins used for programming are single-function only; i.e., they are used as programming pins only. When the device is operating (i.e., not being programmed) they are not in use and are extra pins. Designing with these devices simply requires a direct connection from the device ISR pins to the pins on the ISR programming cable connector for full access to in-system reprogramming.

Dual-Function Pins/Dual-Function Mode

The rest of the devices in the ISR family have pinout/package combinations such that the pins used for programming have dual functionality. They are used as programming pins when the device is being programmed, and they are used as I/Os when the device is in normal operating mode. To use both of these functions, you must design interface logic to isolate programming signals from other devices on the board or incorporate some of this interface logic into the output enable design of the devices driving the ISR devices to be programmed.

Dual-Function Pins/Single-Function Mode

Alternatively, the designer can decide to use these dual-function pins as programming pins only and not connect them as I/Os for normal operation. In this case we refer to the use of a dual-function device being used in single-function mode. The design is simpler in this case as no special interface logic is required.

Designs That Use Devices With Dual-Function Programming Pins

There are two ways to design with devices that have dual-function programming pins. First, you could use the dual-function pins as single-function pins. That is, you could decide to use only the JTAG function of the pins and not use those pins as I/Os in your design. The other way to use them is as true dual-function pins, functioning both as JTAG pins in programming mode and as I/Os in normal operating mode. Most customers will use the dual-function pins only in their JTAG function.

Devices With Dual-Function Programming Pins Used in Single-Function Mode

To use the ISR devices in this way, with the dual-function pins used as programming pins only, the total number of I/Os used in your design must be equal to or less than (n–4), where n is the total number of input and I/O pins available on the device. This way is the preferred method of design. It is much easier and will save both time and components over implementing the kind of logic described in the next section for dual-function pins used in dual-function mode.

To design with the dual-function pins used in single-function mode, simply do not allow an I/O function to be placed on these dual-function pins. Two ways to do this in software are described below.

First, if you are using the Cypress *Warp*® VHDL compiler, you can use a simple synthesis directive called “pin_avoid” to make sure the compiler does not assign signals to whatever pins you specify. In this case, of course, you would specify the pin number of the dual-function pins. An example of the exact text to include in your VHDL code appears in *Figure 12*. This example assumes you are using the CY7C373i or CY7C374i in the 84-pin PLCC package where pins 14, 35, 51, 72, and 83 are the ISR pins. For a complete listing of all the ISR pins of all members of the FLASH370i or Ultra37000 families see the application notes “An Introduction to In-System Reprogramming (ISR) with the FLASH370i” or “An Introduction to In-System Reprogramming (ISR) with the Ultra37000.”

If you prefer you can also ensure the dual-function pins do not get used as I/Os in normal operating mode by explicitly assigning all of the signals to pins in your design. You just need to make sure you assign all of the signals to pins other than the dual-function pins. An example showing how to do this in *Warp* using the “pin_numbers” directive is shown in *Figure 13*. Again, this example assumes you are using the CY7C373i or CY7C374i in the 84-pin PLCC package, so pins 14, 35, 51, 72, and 83 are not being used. Notice that none of the signals used in the example in *Figure 13* are assigned to these pins. This approach can be more time-consuming than using the “pin_avoid” directive, especially if your design has a large number of I/Os. When you do this, you also need to take some device-specific resource information into account. Since the compiler can account for all of this for you automatically, it is usually easier to just use the “pin_avoid” directive. Additionally the “pin_avoid” attribute places fewer restrictions on the fitter software making it easier to fit designs.

Devices With Dual-Function Programming Pins Used in Dual-Function Mode

There are cases where you may need or want to take advantage of the dual functionality of the dual-function programming pins. For example, you may not have enough I/O pins for your design if you do not use the dual-function ISR programming pins as I/Os when the device is in normal operation. Other times, you may want to use the dual-function ISR

```
-- example of using “pin_avoid” for single–function mode of
-- dual–function devices

entity cpuctl is port (
     a             :      in        bit_vector (31 downto 0);
     rd, wr        :      out       bit;
     hold          :      buffer    bit;
     status        :      out       bit_vector (7 downto 0));
attribute pin_avoid of cpuctl:entity is “14 35 51 72 83”;
end cpuctl;

-- architecture would follow
```

Figure 12. VHDL Code Fragment Showing pin_avoid Attribute

```
-- example of explicit pin assignments that avoid ISR pins
-- to facilitate single–function mode of dual–function devices

entity cpuctl is port (
     a             :      in        bit_vector (15 downto 0);
     rd, wr        :      out       bit;
     hold          :      buffer    bit;
     status        :      out       bit_vector (7 downto 0));

-- assign pins below and avoid pins 14, 35, 51, 72, and 83

attribute pin_numbers of cpuctl:entity is
“a(15):12 a(14):13 a(13):15 a(12):16 a(11):17 a(10):18 a(9):19 “ &
“a(8):24  a(7):25  a(6):26  a(5):27  a(4):28  a(3):29  a(2):30 “ &
“a(1):31 a(0):33 rd:36 wr:37 hold:38 status(7):54 status(6):55 “ &
“status(5):56 status(4):57 status(3):58 status(2):59 status(1):60 “ &
“status(0):61“;
end cpuctl;

-- architecture would follow
```

Figure 13. VHDL Code Fragment Showing pin_numbers Attribute

pins as I/Os in normal operation because their physical position makes your board layout easier. If you want to do this in your design, you can do it fairly easily; it simply requires a little bit of extra logic and some additional small components. This next section shows you how to do this.

The TDI, TCK, and TMS programming pins are all inputs to the device during programming, and they always share pins with bidirectional I/Os when they are dual-function pins. The TDO programming pin, on the other hand, is an output pin from the device during programming. It, too, always shares a pin with a bidirectional I/O when it is a dual-function pin. These I/O pins, in turn, can be used as input only, output only, or bidirectional I/Os in any design, based upon the functionality that is described for these pins in the programmable logic chip’s design description. The result is that there are six different cases to consider: An ISR input programming pin (TDI, TCK, TMS) can share a pin with a signal that is an input, an output, or an I/O; and you can have an ISR output programming pin (TDO) sharing a pin with a signal that is an input, an output, or an I/O. We next look at each of these six cases individually.

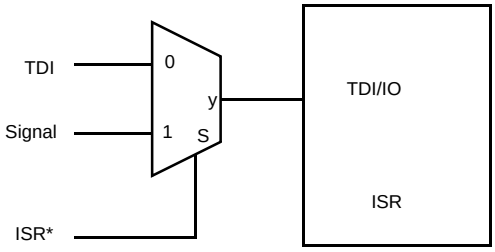
What you are trying to accomplish in all of these cases is fundamentally the same. You are trying to isolate the programming signals from the normal operating signals on the board. You do not want the programming signal to drive or affect anything else on the board when you are programming the ISR device, and you do not want the normal operating signal to drive, affect, or be affected by the programming logic when the ISR device is operating normally in the system. The basic strategy in all of the cases listed above is to use three-state buffers or multiplexers on these signals, and to have those buffers or multiplexers controlled by the ISR* signal from the programming cable. The ISR* signal, recall, is a signal from the programming cable that is a logic LOW when JTAGen is enabling the ISR interface.

Dual-Function Mode Operation: I/O Pin Used as an Input

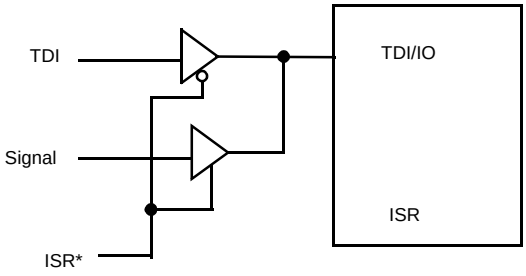
First, consider the case of the ISR programming pins that are inputs to the device during programming; TDI, TCK, and TMS. When one of these device pins is being used as an input during normal operating mode, you simply have to select between one of two inputs based on whether you are in programming mode or in operating mode. This is implemented very easily by using a 2:1 multiplexer where ISR* is the select line, as shown in *Figure 14(a)*. Alternatively, you could implement this by having two three-state buffers whose inputs are TDI (or TMS or TCK) and *signal*, whose outputs are tied together and to the TDI/IO pin, and whose enable lines are controlled by opposite values of ISR*. This is shown in *Figure 14(b)*. One way you could implement this logic is with FCT-family devices. For example, you could use one of the four 2:1 multiplexers in a 74FCT257T to implement the logic shown in *Figure 14(a)*. Alternatively, you could use a pair of transceiv-

ers or pass-transistors from a 74FCT244T or a Texas Instruments SN74CBT3384A, for example, to implement the logic shown in *Figure 14(b)*. The connections for the FCT257T, FCT244T, and SN74CBT3384A are shown in *Figure 14(c)*, *(d)*, and *(e)*, respectively. To reduce unnecessary noise it is a good idea to tie the unused inputs on the FCT devices to ground instead of letting them float.

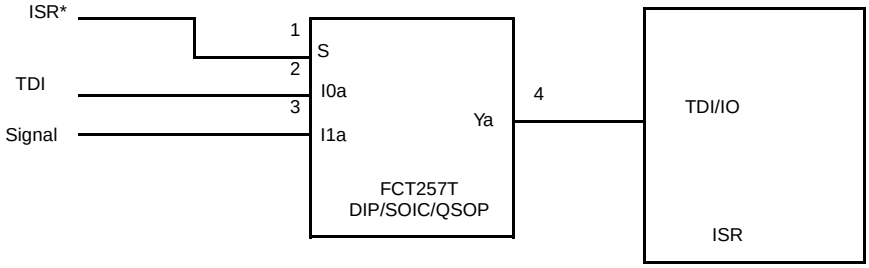
The inverter shown in *Figure 14(e)* can be eliminated by implementing an inversion within the SN74CBT3384A device. This requires using only an external resistor and a few additional connections. An inversion of the connection to pin BE1* is accomplished by connecting +5V to pin A1 and connecting one end of a resistor to GND and the other end to pin B1. B1 is then the inverse of the input connected to BE1*, which is



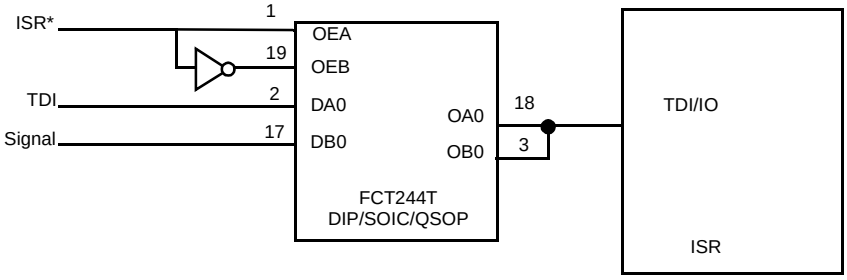
(a) Multiplexer Solution



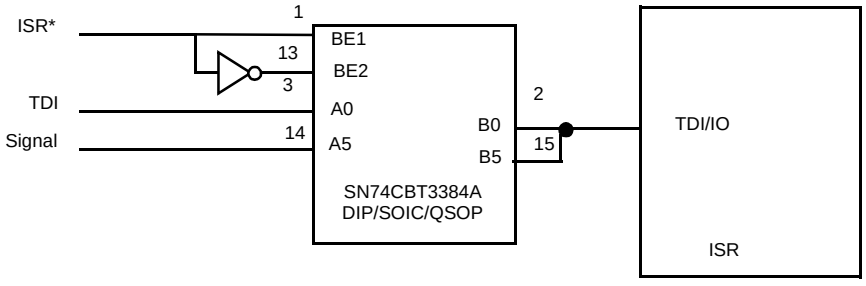
(b) Buffer Solution



(c) FCT257T Implementation



(d) FCT244T Implementation



(e) SN74CBT3384A Implementation

Figure 14. Design for Dual-Function Pins: TDI/TCK/TMS used with Input

ISR*. By implementing this inversion, the inverter in *Figure 14(e)* can be removed, and pin B1 can be connected to pin BE2*. The BE1*, A0, A5, B0, and B5 pin connections remain the same.

The FCT devices shown are just one possible way of implementing this logic, of course. There are others, including using extra pins and gates from an ASIC, FPGA, CPLD, or PAL® device already on the board. Regardless of whether the buffer or multiplexer is in an FCT device, ASIC, FPGA, or other device, there will be some additional propagation delay for the normal operating signal due to the presence of that logic. This must be accounted for in your design. Using the SN74CBT3384A provides the smallest extra delay, less than one quarter of one nanosecond. The extra delay holds true for the other cases presented next.

Dual-Function Mode Operation: I/O Pin Used as an Output

In the case of the TDI, TCK, or TMS sharing a pin with an I/O that is used only as an output during normal operating mode, the logic is slightly different. Much like the case above, you can just use a pair of three-state buffers or pass-transistors to separate the signals that are used for the two different functions. In this case, however, instead of tying the two outputs together, you tie the output of one buffer both to the dual-function pin of the ISR device and to the input of the other buffer. The input to the first buffer is the programming function signal, and the output from the other buffer is the normal operation output *Signal*. The first buffer is enabled when ISR* is asserted and is disabled otherwise, and the second buffer is enabled when ISR* is deasserted and is disabled otherwise. This is shown in *Figure 15*. Thus, when the device is being programmed, TDI (or TMS or TCK) is driving the TDI/IO pin and *Signal* is in three-state, and when the device is not being programmed, the TDI/IO pin is not driven as an input allowing the ISR output to drive *Signal*. Because *Signal* is in the three-state during programming, you may need to have a pull-up or pull-down resistor on *Signal* depending on how you use it on your board.

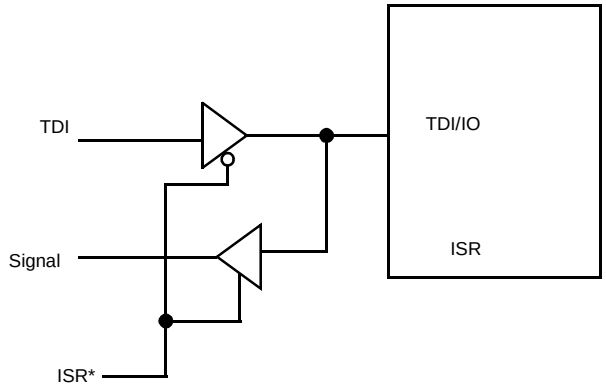


Figure 15. Design for Dual-Function Pins: TDI/TCK/TMS Used as an Output

You can also use a SN74CBT3384A as an alternative to the buffers, as shown in the previous case. This would be the most flexible solution because it would work for all configurations—the pin used as an I, O, or I/O—and allows you to decide later exactly how to use that pin.

Dual-Function Mode Operation: I/O Pin Used as an Input and an Output

In the case of the TDI, TCK, or TMS sharing a pin with an I/O that really is used as a bidirectional I/O, the logic needed is a little more complicated. As seen in *Figure 16*, part of the logic is a combination of the two solutions for the two individual cases above in the way it uses ISR* to separate the programming function of TDI (or TMS or TCK) from the input and output functions of *Signal* during normal operating mode. There is more than just this logic required, however. It is also necessary to use an extra pair of buffers to separate the input and output functionality of *Signal* itself. This is required to keep from unintentionally building a feedback loop and is implemented using an extra signal that indicates the direction of the I/O pin. In this example, we assume we have a signal called *dir*, and that *dir* is HIGH when the I/O pin is being used as an input and *dir* is LOW when the I/O pin is being used as an output.

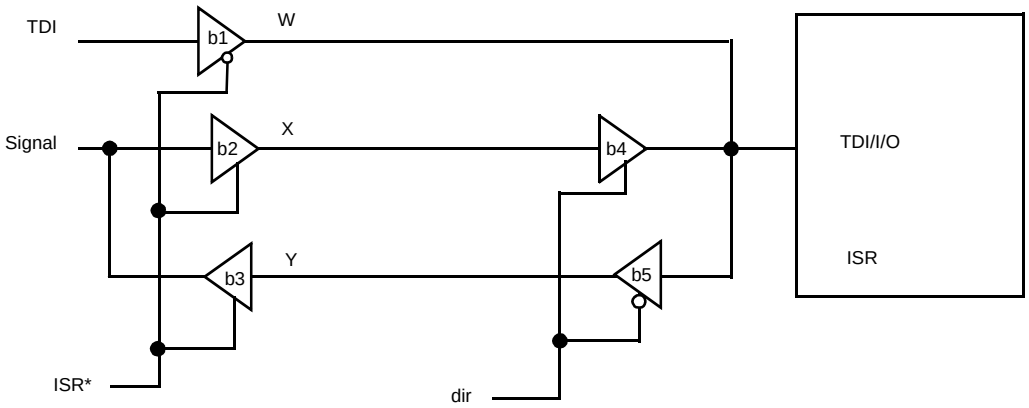


Figure 16. Design for Dual-Function Pins: TDI/TCK/TMS Used With an I/O

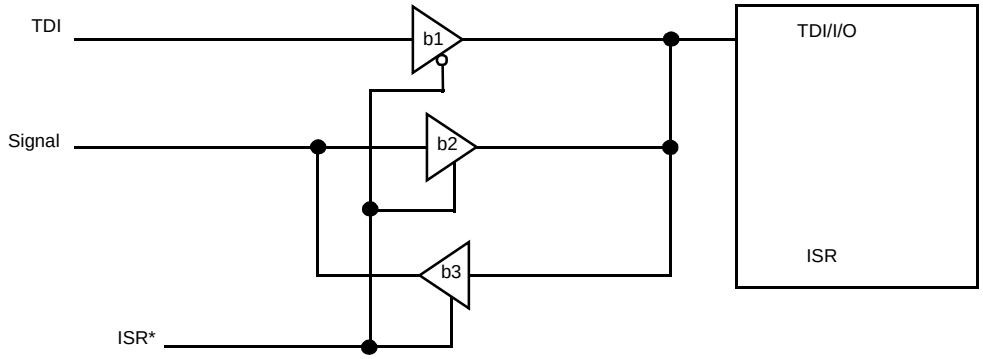


Figure 17. TDI/TCK/TMS Used With an I/O: Example of Incorrect Solution

To understand why this is necessary, consider just combining the logic from *Figure 14(b)* and *Figure 15*. The result would be the logic shown in *Figure 17*, which is different from *Figure 16* in that buffers *b4* and *b5* were eliminated and intermediate signals *w*, *x*, and *y* are now all simply connected together and to the TDI/IO pin. In the logic of *Figure 17*, when the ISR device is in normal operation mode and ISR* is HIGH, buffers *b2* and *b3* would both be enabled. If *Signal* were an input at that time, it would drive the input to buffer *b2*, whose output would drive the input to buffer *b3*. The output of buffer *b3* would be driving the input of *b2* again, resulting in a feedback loop that could produce undesired affects. The same thing would happen if *Signal* were an output at that time.

Buffers *b4* and *b5* in *Figure 16* prevent this. In the logic of *Figure 16*, when *signal* is an output from the ISR device, *b5* is enabled and *b4* is disabled; when *Signal* is an input to the device, *b4* is enabled and *b5* is disabled. In both cases, both the function and value at the pin of the device and the function and value of *Signal* are the same, correct, and only driven by one source. There is no dangerous self-driving feedback system like there is in *Figure 17.*

The limitation of this solution is that it requires the extra signal *dir*. This signal may be already available; in fact, it may be an input to the ISR device itself for use as the $\overline{OE}$-control on the pin in question. If it is not already available, you will need to generate it using other logic on the board. If you cannot do it using other logic on your board, you should certainly be able to generate it using logic inside the ISR device itself, because, as pointed out above, it should be the same signal as the $\overline{OE}$ used on that pin internally. To get the signal out of the ISR, however, requires an additional pin. If you are using the logic in *Figure 16* to save a pin, having to use a pin on the device to generate *dir* will not gain you anything. If generating one *dir* will help you save two or three pins by allowing you to use two or three of TDI, TCK, and TMS as dual-function pins, then you will still have a net savings of one or two pins and it may be worth it.

As was mentioned in the case where the TDI (or TMS or TCK) dual-function pin was being used with an input-only pin or with an output-only pin, you can also use the SN74CBT3384A solution of *Figure 14(e)* when trying to use the TDI (or TMS or TCK) dual-function pin as a bidirectional I/O pin in normal operating mode.

The logic for using the dual-functionality of the TDO/IO pin is essentially the same as is shown in the above three cases. The only difference is that TDO is an output during programming mode instead of an input. Therefore, the only difference in the logic is the orientation of some of the buffers. The solutions for the TDO case are presented without further explanation. The logic diagram for the case where TDO is connected to an I/O used only as an input is shown in *Figure 18*. The logic diagram for the case where TDO is connected to an I/O used only as an output is shown in *Figure 19*. The logic diagram for the case where TDO is connected to an I/O really used as a bidirectional pin is shown in *Figure 20*. You can alternatively use the SN74CBT3384A solution presented in *Figure 14(e)* in each of these three cases.

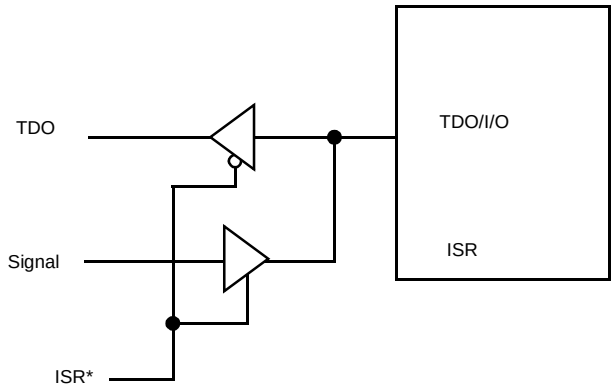


Figure 18. Design for Dual-Function Pins: TDO Used With an Input

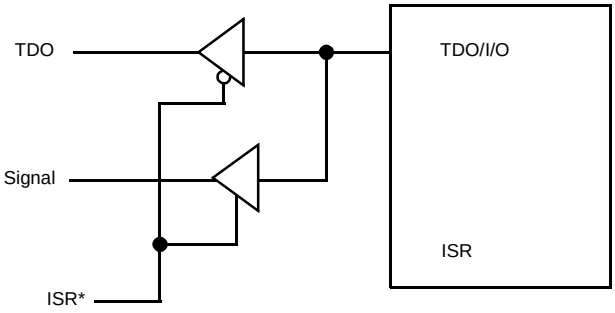


Figure 19. Design for Dual-Function Pins: TDO Used With an Output

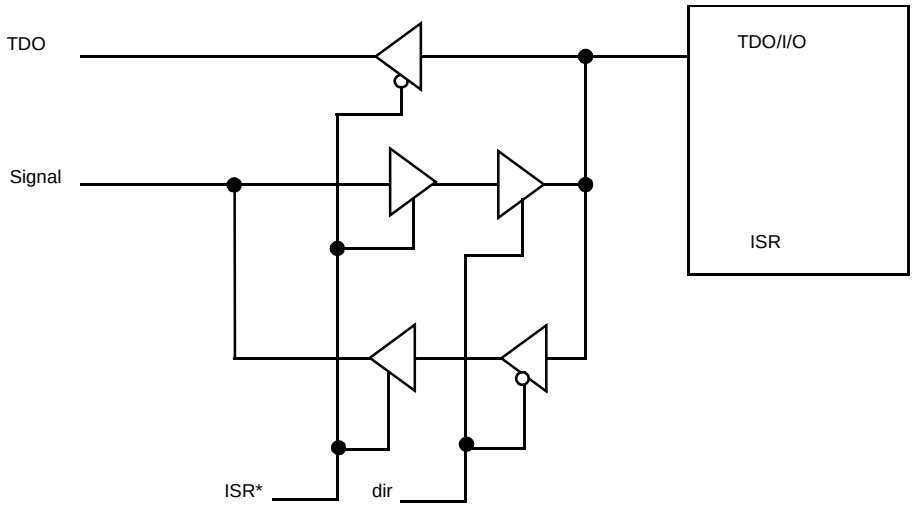


Figure 20. Design for Dual-Function Pins: TDO Used With an I/O

Dual-Function Summary

To summarize this section, there are many ways to accomplish programming when using devices with dual-function pins. The easiest way to use the dual-function device is in single-function mode. This uses the dual-function pins as programming pins only, and is easily accomplished using the pin_avoid and pin_numbers directives in the *Warp* design file. There are also going to be cases where you will want to use the dual-functionality, most likely because you need some or all of the four ISR programming pins as inputs, outputs, or I/Os during normal operation to get all the signals you need into and out of the device for your design. The circuits needed to share these pins are relatively straightforward and require only buffers or pass-transistors. These are circuits you can implement using FCT or other logic, or you may be able to implement them using extra gates and pins of an ASIC, FPGA, or another PLD you already have on the board.

Appendix B. Simple Cascading Considerations for Ultra37000 and FLASH370i CPLDs

You can cascade many ISR devices in a system. That is, you can daisy-chain the devices together and connect their programming pins in such a way that the devices can be programmed from a single connection to the ISR programming cable.

Cascading Single-Function ISR Devices or Dual-Function Devices in Single-Function Mode

To do this, you simply tie all of the JTAGen, TCK and TMS pins of each device to those same pins, respectively, on all of the other devices, and then connect them to the corresponding pins of the ISR cable connector. You then connect the TDI pin from the cable connector to the TDI pin of the first device in the chain, then connect the TDO output of that device to the TDI input of the next device in the chain, then connect the TDO output of that device to the TDI input of the next device in the chain, and so forth, until you finally connect the TDO output of the last device in the chain to the TDO pin of the cable connector (see *Figure 21*). In *Figure 21* many of the Ultra37000 devices are single-function mode devices, therefore the JTAGen pin does not exist at all for these devices.

Cascading Dual-Function ISR Devices in Dual-Function Mode

In addition to the extra circuitry needed for the dual-function pins per ISR device, the JTAGen pin must also be connected differently on the Ultra37000 devices than the FLASH370i devices. This is necessary because of its different functionality with a TTL HIGH input level as previously mentioned. This is explained further in the rest of this application note. *Figure 24* shows an example of two dual-function mode devices operating in dual-function mode on the TMS signal of two of the three devices in the chain.

Driving the JTAGen Pin on the Ultra37000 Dual-Function Mode Devices When Being Used in Single-Function Mode

Figure 21 shows that the user does not need to worry about driving the JTAGen pin to a particular level if the JTAG pins are only used for ISR operations. This example shows that it is possible for the dual-function pins to operate in the I/O mode for the Ultra37256 device when the ISR programming cable is removed since a TTL LOW level could exist on the JTAGen pin. Observation of *Figure 21* shows that I/O contention problems are possible through the ISR connections from one ISR device to another if multiple Ultra37000 devices were connected in the same chain and the JTAG pins are used in their I/O function. Output contention between devices is not a problem if the output from all devices is the same polarity. When and if further ISR operations need to be performed the ISR cable drives the JTAGen pin HIGH which three-states the I/O function and re-selects the JTAG function on the dual-function pins so there is no contention problem possible between the ISR device and the ISR cable.

Driving the JTAGen Pin on the Ultra37000 Dual-Function Mode Devices When Being Used in Dual-Function Mode

In many designs the user will elect to either use a single-function mode Ultra37000 device or use a dual-function mode Ultra37000 device in single-function mode. For these cases the user can ignore the following discussion. Similar to the FLASH370i family of devices, a TTL LOW level on the JTAGen pin enables the I/O functionality of the Ultra37000 dual-func-

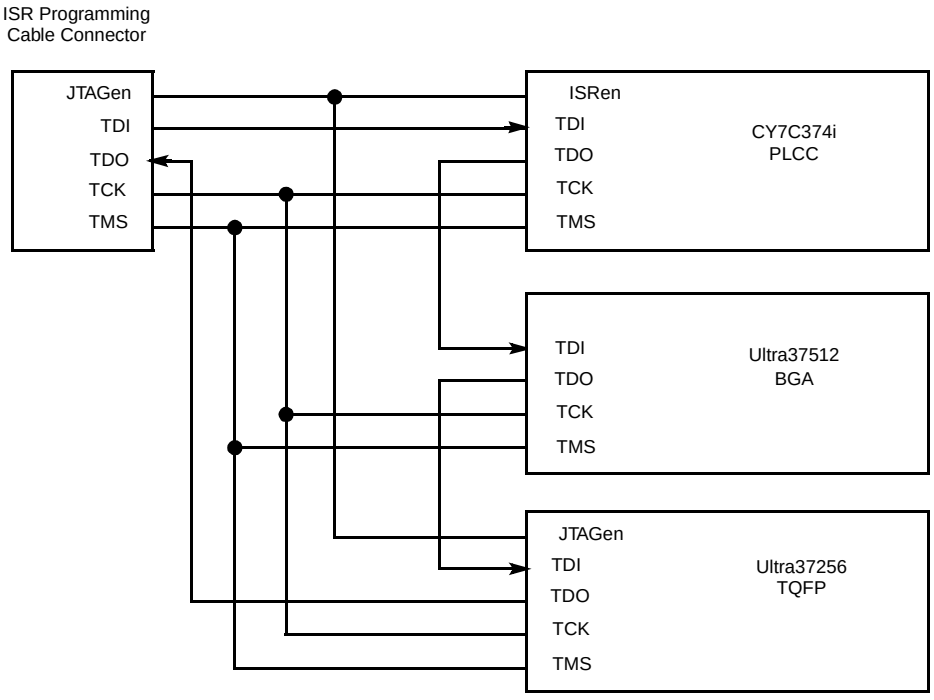


Figure 21. Simple Cascading Example - Single-Function Mode Operation

tion devices. For the FLASH370i it is permissible to actually let the JTAGen pin float to retain the I/O capability of dual-mode pins. For the Ultra37000 devices this pin must be driven to a LOW level to ensure the I/O functionality. For Ultra37128 and smaller devices a weak pull-down device replaces the bus-hold latch on the JTAGen pin. This pull-down device (see the data sheet parameter I_{JTAG}, which shows the strength of the pull-down device) keeps the pin in the LOW state after programming and prevents the need for external biasing on the JTAGen pin if a Ultra37000 device replaces a FLASH370i device. Early Ultra37256P160 silicon did not have this pull down device on the JTAGen pin but more recent silicon has this device. If the pull-down device is not employed then a bus-hold latch is employed and this latch could hold a HIGH value on the pin. With multiple devices in the chain, some devices being Ultra37128 and smaller and some being Ultra37256P160 early silicon or FLASH370i devices, it is still possible for the JTAGen pin to be pulled into the HIGH state. Two methods for driving the JTAGen pin LOW, using a external pull-down resistor and using an external component are presented.

Using a Pull-down Resistor to Drive the JTAGen Pin LOW to Use the I/O Function of the Dual-Function Pins

The JTAGen pin can be held at a TTL LOW by using a pull-down resistor to ground. This works fine provided there are not too many devices in the ISR chain. The problem with the simple resistor is that the bus-hold latches on the JTAGen pins may disturb the pull-down operation if there are many other FLASH370i devices in the chain. This is because the bus-hold latches that are connected in parallel produce a lower source impedance. The bus-hold latch differences between the two ISR family members are discussed further in this note. The resistor must be able to provide a low enough resistance to overpower the combined bus-hold latches in parallel such that the voltage on the JTAGen pin drops below the trip point (V_{trip}) of the bus-hold latches. Once the JTAGen pin voltage drops below V_{trip} all the bus-hold latches connected in parallel will flip to the desired LOW state. The maximum resistance that is guaranteed to overpower N FLASH370i bus-hold latches in parallel is given by the formula:

$$R_{pulldown} = V_{trip}/(N*(I_{BHHO}))$$

where V_{trip} is 1.5V and I_{BHHO} is –500 µA (maximum current that is guaranteed to invert the state of a single bus-hold latch). $R_{pulldown}$ is 3 kΩ for 1 device, 600Ω for 5 devices, and 300Ω for 10 devices in the chain. It is recommended that the number of devices in the chain be limited to 5 devices if the other devices in the chain are all FLASH370i devices. The above limitation of 5 devices in the chain is suggested because the resistor value would need to be reduced which would place too high a DC current load on the JTAGen signal driven to 12V from the ISR cable. With a pull-down resistance of 600Ω, the DC current load on the JTAGen pin is 12/600 or 20 mA, which is an acceptable load. Resistor values less than 600Ω would require a higher wattage resistor than the standard 1/4 watt rating, assuming 12V is needed for programming FLASH370i devices in the same chain. If only Ultra37256P160 early silicon devices are in the chain and the UltraISRPCCABLE is used then the JTAGen pin only goes to a TTL HIGH so the number of devices in the chain can be increased to 10. The above restriction of the pull-down resistor can be avoided by using an external component to choose between a TTL HIGH level and a TTL LOW level on the JTAGen pin. Again the need for external biasing may not be needed in many case because of the pull-down device incorporated on the Ultra37128 devices and smaller and to be included on the Ultra37256 as well.

Using an External Component to Drive the JTAGen Pin LOW to Use the I/O Function of the Dual-Function Pins

The JTAGen pin can be driven LOW using any of the solutions already presented in *Figure 14* regarding using the dual-function pins in dual-function mode. The only difference is that the JTAG input is JTAGen, which is tied to V_{CC} instead of one of the four JTAG pins and the “signal” input is connected to ground. You may be able to use unused resources in the external component solutions presented in *Figure 14* to implement this logic. Any of these solutions can be used for multiple Ultra37000 devices ganged together. *Figure 22* shows two examples for driving the JTAGen pin from the control of signal ISR*. As mentioned before, a pull-up resistor is also needed on the ISR* signal at the 10-pin header connector.

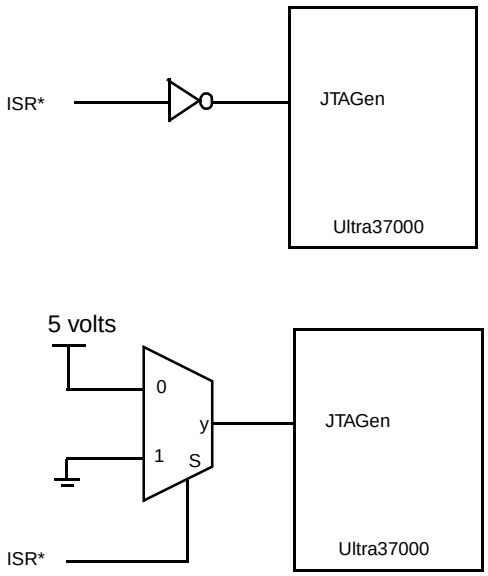


Figure 22. JTAGen Driven for the Ultra37000

Figure 23 shows the JTAGen pin driven by an extra I/O pin of the Ultra37000 device by simply inverting the ISR* control signal. This would seem to be a simple alternative to adding extra components. This solution, however, won’t work. The problem is that the I/O pins of the device enter three-state when the ISR mode is enabled (JTAGen driven HIGH). Because the I/O is three-stated there is no way to drive the JTAGen pin LOW. The dual-function pins are stuck in the JTAG function because the bus-hold latch, which is always enabled, has latched a HIGH level on the pin. *Figure 24* shows how to combine dual-function FLASH370i and Ultra37000 devices in the same chain with the appropriate JTAGen connections assuming the user wants to use one dual-function pin in dual-function mode for each of the Ultra37000 devices. The figure shows that one mux can be used for two Ultra37000 devices but many more devices can use the same mux output signal. In this example the TMS pin on both Ultra37000 devices is used in dual-function mode where the I/Os are used as JTAG pins and as input pins. The

10-kΩ resistor on the ISR* signal is required to keep the mux input HIGH when the ISR cable is removed from the board to keep the JTAGen input LOW. Remember that, if it is not necessary to use the dual-function pins in dual-function mode, the mux can be removed and the JTAGen pin can be tied to V_{CC} or left connected to the JTAGen pin from the 10-pin connector.

At this point the user should understand how to connect FLASH370i and or Ultra37000 devices in an ISR chain and how to bias the ISR interface for his programming and functional needs. There are some remaining minor functional differences between the two device families related to the placement of the bus-hold latches on the five JTAG interface pins, which was addressed in the Device-Specific ISR Design Considerations section of this application note.

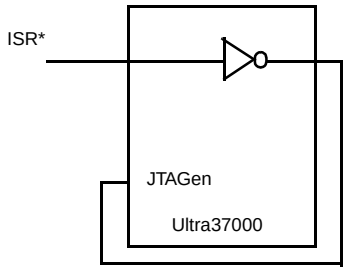


Figure 23. JTAGen Incorrectly Driven by Ultra37000 I/O

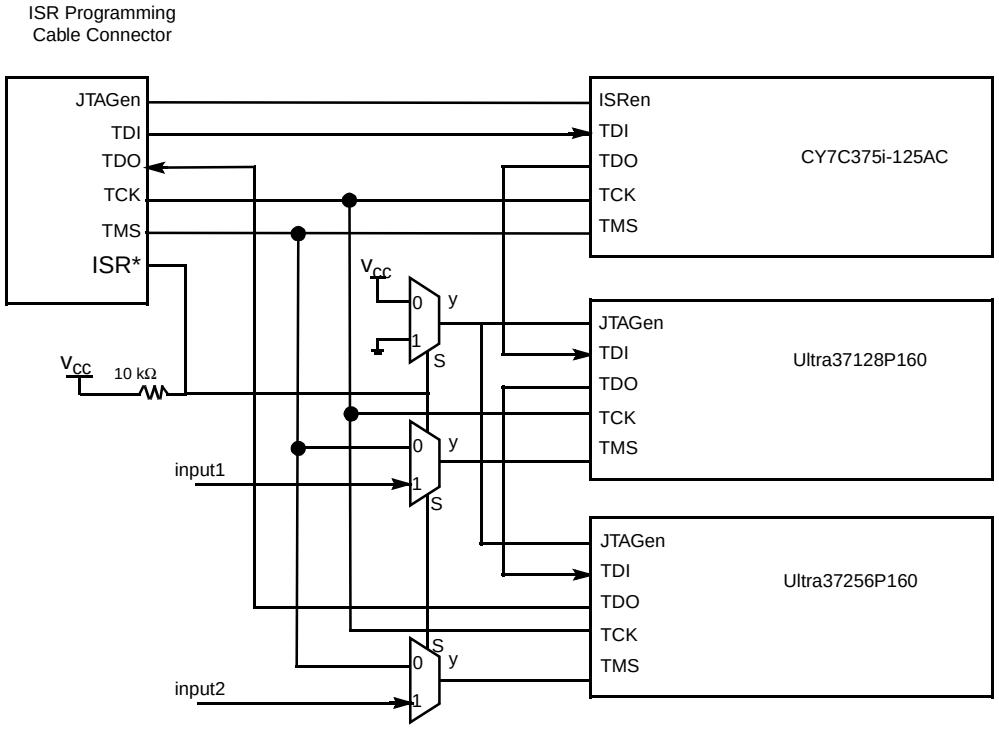


Figure 24. Cascading Dual-Function Ultra37000 and FLASH370i Devices in the Same Chain

Appendix C. Additional FLASH370i Family Design Considerations

This appendix describes additional FLASH370i device and board-level design considerations not covered in the FLASH370i device-specific section of this application note.

State of ISR Programming Pins When the ISR Pins Float for FLASH370i Devices

The ISR pins can be floating if, for example, a programming cable is not attached to the on-board ISR header. The ISR programming cable is only plugged into the connector on your board during programming; however, it is acceptable to float the ISR pins for the FLASH370i devices when the board is powered up and operating. This is acceptable because the ISR devices have been designed with bus-hold structures on every input, input/clock, and I/O pin, including the programming pins for both dual-function and single-function devices. The exception to this is the TDO pin on single-function devices, which is an output only and the JTAGen pin, which may incorporate a pull-down device instead of a bus-hold latch. The bus-hold latch eliminates the need to use external pull-up resistors or any other technique for handling the case where the ISR programming pins are left floating due to the ISR programming cable being disconnected.

Bus-hold structures enable a pin to maintain its most recent logic value even when it is three-stated, whether that value was being driven in as an input pin or driven out as an output pin. This is done with a weak latch connected to the pin. Since the ISR programming pins have bus-hold structures, when the ISR programming cable is disconnected and the board is powered on, the ISR programming pins all maintain a logic LOW or logic HIGH value even though they are no longer being driven. If the ISR programming cable is disconnected when the board is powered-down, or if the board is powered-down and then back up after the cable has been disconnected, there is still no problem. The ISR bus-hold structures have been designed to always power-up with a logic HIGH level maintained on the pins to emulate an internal pull-up.

The utility of the bus-hold structures applies both to the case of ISR programming pins used as single-function pins and as dual-function pins. The bus-hold latch does not interfere with the normal function of the pin because of the relative weakness of the latch to the output driver. The latch is more than 50 times weaker than the ISR device's output driver.

Note that the ISR* pin from the ISR programming cable connector does not connect to an ISR device pin. Since it does not, you must use a pull-up on the ISR* signal on your board so that it is not left floating when the ISR programming cable is disconnected. This signal can be used to support dual-mode JTAG/IO pins or to simply monitor when programming operations are taking place.

State of the ISR Programming Pins When the ISR Programming Cable is Not Attached for Both Ultra37000 Devices and FLASH370i Devices in the Same ISR Chain

For the case where both single-function mode Ultra37000 and FLASH370i devices are in the ISR programming chain, no external resistor is needed even on the TCK pin. This is because these pins are connected in parallel to all devices in the chain and the bus-hold latches, which are always present on the FLASH370i devices, hold the pin to a logic LOW or HIGH level preventing it from floating.

Handling the 12V Signal on the Board for FLASH370i Devices

Unlike the Ultra37000 family, in which the JTAGen signal serves only to choose either JTAG or I/O function on the dual-function pins, FLASH370i devices uses this pin as the high-voltage, low-impedance path required for programming the device. There are two requirements on the ISRen programming voltage that necessitate special handling when programming FLASH370i devices. The first is that its voltage must be in the range $11.4V \leq ISRen \leq 12.6V$ during programming, and the second is that its maximum transient current is 40 mA per ISR device during programming. The 5V/12V DC/DC converter and other components in the ISR programming cable described in this application note have been chosen to ensure that these specifications are met. Therefore it is suggested that the 12V supplied by the ISR cable be used for programming rather than another 12V supply that may be available on the board.

Due to the higher than usual current and voltage requirements on the ISRen signal, the trace on the printed-circuit board connecting the ISRen pin from the ISR programming cable to the ISRen pin on the FLASH370i device(s) also deserves special attention. First, to handle the current, the trace should be double the width of the standard traces. Second, the trace should be kept as short as possible. In general, this means the connector for the ISR programming cable should be placed as close as possible to the FLASH370i devices on the board. Since the connector is small, it is much easier to move the connector closer to the devices than change the whole board layout to place the devices close to the chosen spot for the connector.

Decoupling the ISRen Pin to Ground for FLASH370i Devices

One further board layout recommendation is to place a 10-nF capacitor located at the 10-pin header connector on the circuit board on the JTAGen pin to ground if there are FLASH370i devices to be programmed on the board. If proper ISR cable connection procedures explained in this application note are followed this capacitor is not needed. If the ISR cable is hot-socket connected to the user's board, which is not recommended, then this capacitor insures that a proper, slow ramping 12V is applied to the devices to be programmed under all operating conditions with no risk of damage to the JTAGen pin. Since the ISR cable could easily be hot-socket connected by accident, it is advisable and simple to incorporate this decoupling capacitor.