Designing With Cypress In-System Reprogrammable™ (ISR™) CPLDs for PC Cable Programming

Introduction

This application note presents how to design with the Cypress In-System Reprogrammable™ (ISR™) families of complex PLDs, the FLASH370i™ family and the Ultra37000™ family, for programming from a PC with the ISR programming cable. 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 the two product families. This note discusses all issues related to programming and reprogramming the devices in-system through the control of a PC (i.e., while they are soldered onto a printed circuit board). These issues include: an explanation of the ISR programming cables and ISR connections required, using the ISR programming pins for both functional logic and for programming, and connecting ISR devices in a chain for programming as well as proper functional operation. For the purposes of this note the term "ISR devices" refers to both the FLASH370i and the Ultra37000 devices. "Ultra37000" refers to both Ultra37000 and Ultra37000V unless otherwise stated.

Throughout this application note, assume that the ISR devices are programmed in system by means of the ISR cable that connects the board to a PC. This ISR programming cable is provided by Cypress, and the details of the cable and the connector on the board to which it interfaces are explained in detail in this application note. There are other ways to program or reprogram the parts in-system as well, and many of the topics covered and solutions shown in this application note also apply to those methods.

ISR Programming Pins

The programming pins for the FLASH370i are named ISREn, SDI, SDO, SMODE, and SCLK. For the Ultra37000 the same pins are called JTAGen, TDI, TDO, TMS, and TCK respectively. The reason for using the standard JTAG naming convention for the Ultra37000 family is that these devices support Boundary Scan testing and are compliant with the IEEE 1149.1 (JTAG) standard. The programming interface for the FLASH370i devices also complies with 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 cable and one connector 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.

JTAGen (ISREn) Enables the 4-pin JTAG Interface

This pin is present on all packages where the JTAG pins share their functionality with IO pins. For the Ultra37000, when this pin is at a TTL HIGH level the JTAG pin functionality is selected. When it is at a TTL LOW level the IO 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 IO pin functionality is selected when it is at a TTL level between 0 and 5V. This functional difference in the JTAGen pin for the two ISR device families is important and is discussed further in this application note.

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.

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.

The ISR devices are programmed using a PC as shown in *Figure 1*. The ISR programming cable connects the parallel port of the PC to a cable header on the board where the ISR devices are soldered. The header on the board connects to the traces that go 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, header, and traces, to program the devices with the appropriate JEDEC files.

The ISR Programming Cables

Table 1. ISR Cable Used for Programming

Cable	Devices
ISRPCCABLE(rev 0.03)	FLASH370i, Ultra37000, (Ultra37000V with 5V supply)
UltraISRPCCABLE	Ultra37000, Ultra37000V

There are two ISR cables available to program ISR devices. *Table 1* shows the two cables and their appropriate usage. The ISRPCCABLE cable (ISRPCCABLE.03 - revision 0.03) can be used to program all ISR devices. The Ultra37000V devices can also be programmed with this cable provided a 5V supply voltage is used to power the cable instead of a 3.3V supply. 5V is present on the circuit board if the Ultra37000 or

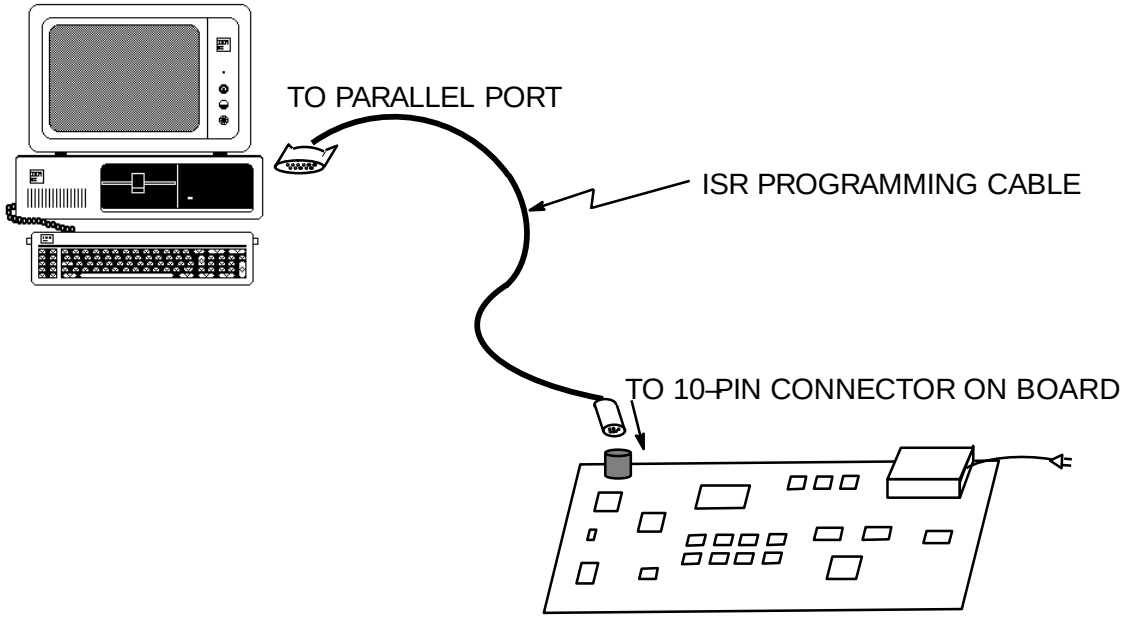


Figure 1. ISR Programming Set-up

FLASH370i devices are also present along with Ultra37000V devices. The UltraISRPCCABLE is used to program the Ultra37000 and Ultra37000V devices but not the FLASH370i devices. This cable does not have a 5V to 12V DC/DC converter since the Ultra37000 family does not require a supervoltage for programming. The ISRPCCABLE has a DC/DC converter built into it that receives 5V from the board and supplies the 12V programming voltage to the JTAGen pin. The ISRPCCABLE (revision 0.03 or greater) can be used to program the Ultra37000 devices because the JTAGen pin is 12V tolerant. The Ultra37000V devices are also 12V tolerant. This means that the application of 12V on the JTAGen pin is equivalent to placing a TTL HIGH value on the pin. To program these devices the user can either use the ISRPCCABLE (revision 0.03 or greater) or use the UltraISRPCCABLE. The latter solution is recommended if the user does not need to program FLASH370i devices.

Dimension of the Two ISR Cables

The ISRPCCABLE is a six-foot cable with one foot as a detachable, flexible ribbon cable. The ribbon cable section is connected to the 12V converter box at the end of a five-foot cable. The UltraISRPCCABLE is simply a short one foot ribbon cable with a small PC board mounted inside of the connector. The user may need a parallel port extension cable when using this cable since there may not be enough cable length from the parallel port on the back of the PC to the circuit board. Both cables plug into the female parallel port of the PC or the female end of an extension cable for the UltraISRPCCABLE.

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)

Both ISR programming cables provided by Cypress have a female end which plugs into these connectors. The position of the signal pins on the connector is shown in *Figure 2*. 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 itself. The notch on the female connector is located near the ISR* signal as shown. If the right-angle connector is used make sure the notch of the cable faces up so that it can be plugged into the boxed header.

PIN1

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

Figure 2. Layout of Connector for Cable on Board Top View

To program a single ISR device using the ISR programming cable described here, route the JTAGen, TDI, TDO, TCK, and TMS pins from the cable connector to the JTAGen, TDI, TDO, TCK, and TMS pins of the ISR device, respectively. Multiple devices can be programmed in a single ISR programming chain which is explained later in this application note. For multiple devices in the chain, the TDI and TDO pins are connect-

ed 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 5V or 12V and ISR operations, such as programming, are in process on the devices 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. 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 later in this application note. The logic LOW and logic HIGH levels on ISR* are 0 and V_{CC}, respectively. 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 using a pull-up resistor to the V_{CC} pin of the ISR connector to indicate to the circuit 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 5V to the DC/DC converter in the ISR cable. This is necessary for the ISRPCCABLE to be able to generate the 12V voltage level on JTAGen needed to program the FLASH370i devices. It also supplies power to a buffer in the cable for all other JTAG signals. On the UltraISRPCCABLE the V_{CC} simply supplies power to the buffer in the cable for the JTAG signals. 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 the 5V to V_{CC} header pin instead of the 3.3V.

Connecting the ISR Cable to the Board

Because 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 UltraISRPCCABLE cable since there is no 12V signal generated in the cable.) Once the cable is connected, the user can power up the board. This assures a normal slew rate on the JTAGen pin for all possible conditions. It is also recommended that the user run the ISR software after the cable is plugged into the users board and the 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.

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. For a complete listing of all single- and dual-function mode devices for 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.”

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 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 3*. 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 4*. 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 4* 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 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

```
-- 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 3. 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 4. VHDL Code Fragment Showing pin_numbers Attribute

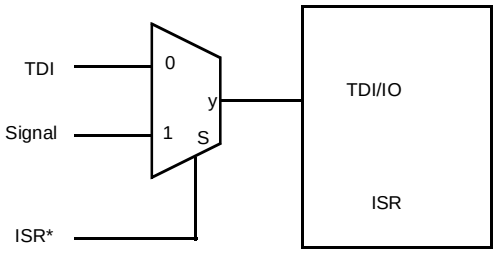
pin with a bidirectional I/O when it is a dual-function pin. These I/O pins, in turn, can be used as input only, output only, or bidirectional I/Os in any design, based upon the functionality that is described for these pins in the programmable logic chip's design description. The result is that there 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.

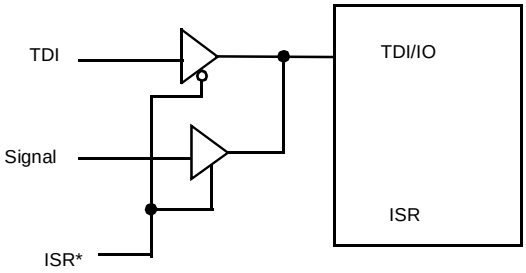
Dual-Function Mode Operation: I/O Pin Used as an Input

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 5(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 5(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 CY74FCT257T to implement the logic shown in *Figure 5(a)*. Alternatively, you could use a pair of transceivers or pass-transistors from a CY74FCT244T or a CYBUS3384 to implement the logic shown in *Figure 5(b)*. The connections for the FCT257T, FCT244T, and CYBUS3384 are shown in *Figure 5(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.

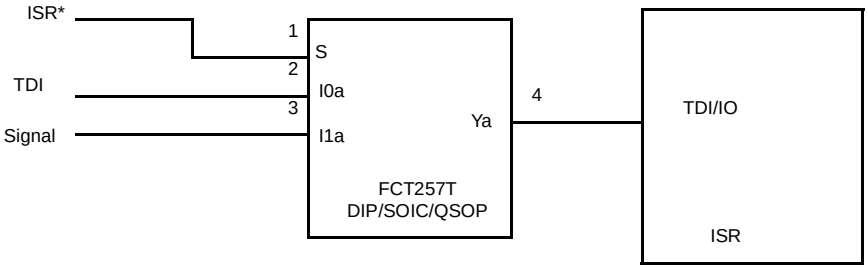
The inverter shown in *Figure 5(e)* can be eliminated by implementing an inversion within the CYBUS3384 device. This requires using only an external resistor and a few additional connections. An inversion of the connection to pin BE1* is accomplished by connecting +5V to pin A1 and connecting one end 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 ISR*. By implementing this inversion, the inverter in *Figure 5(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.



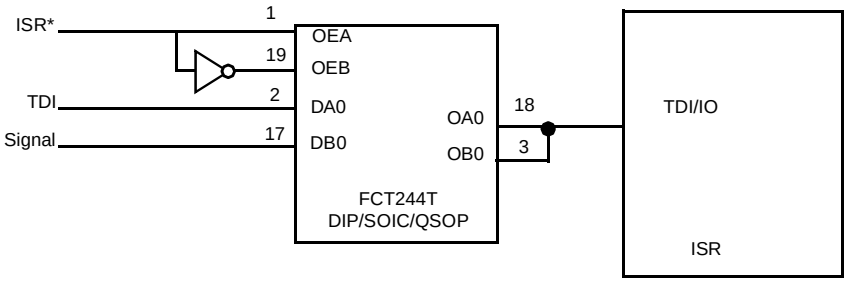
(a) Multiplexer Solution



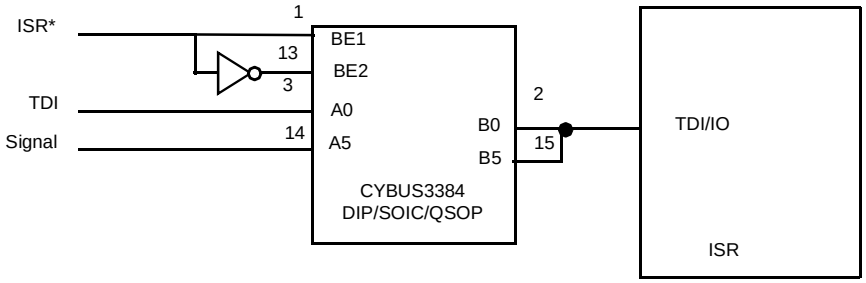
(b) Buffer Solution



(c) FCT257T Implementation



(d) FCT244T Implementation



(e) CYBUS3384 Implementation

Figure 5. Design for Dual-Function Pins: TDI/TCK/TMS used with Input

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 CYBUS3384 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 6*. 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.

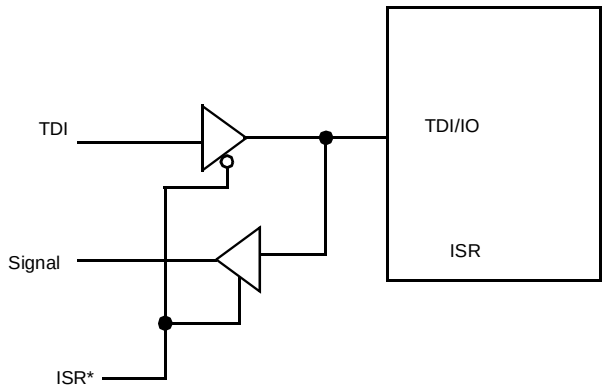


Figure 6. Design for Dual-Function Pins: TDI/TCK/TMS Used With an Output

You can also use a CYBUS3384 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 7*, 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.

To understand why this is necessary, consider just combining the logic from *Figure 5(b)* and *Figure 6*. The result would be the logic shown in *Figure 8*, which is different from *Figure 7* 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 8*, 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 7* prevent this. In the logic of *Figure 7*, 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 8.*

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 7* 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 CYBUS3384 solution of *Figure 5(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

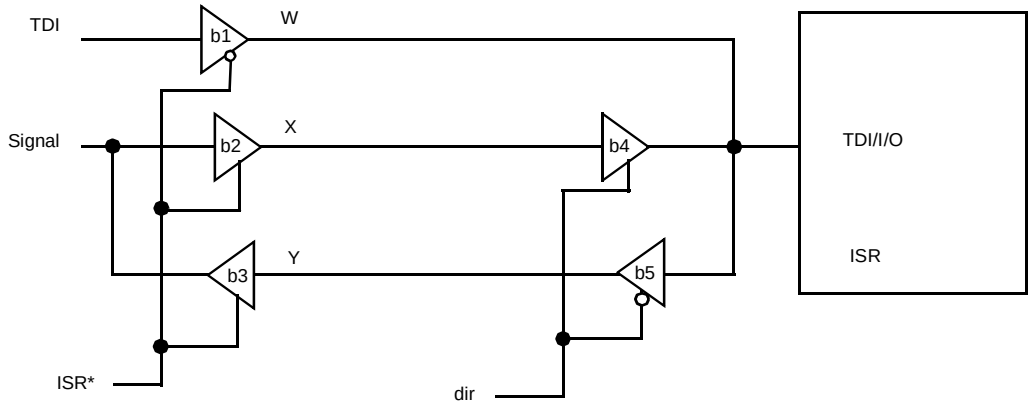


Figure 7. Design for Dual-Function Pins: TDI/TCK/TMS Used With an I/O

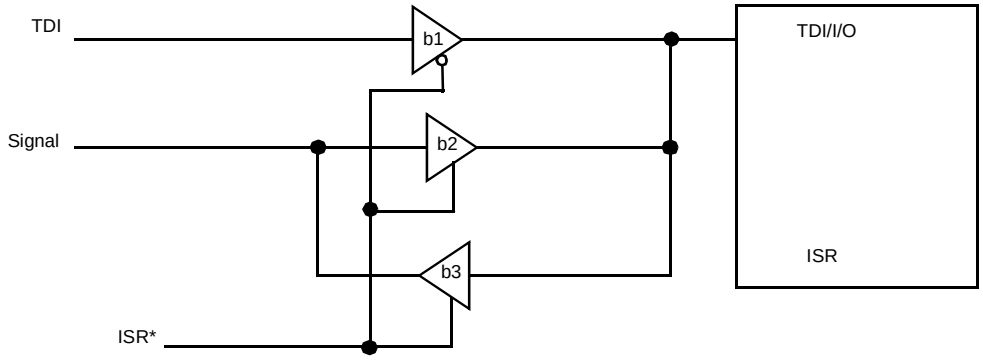


Figure 8. TDI/TCK/TMS Used With an I/O: Example of Incorrect Solution

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 9*. The logic diagram for the case where TDO is connected to an I/O used only as an output is shown in *Figure 10*. The logic diagram for the case where TDO is connected to an I/O really used as a bidirectional pin is shown in *Figure 11*. You can alternatively use the CYBUS3384 solution presented in *Figure 5(e)* in each of these three cases.

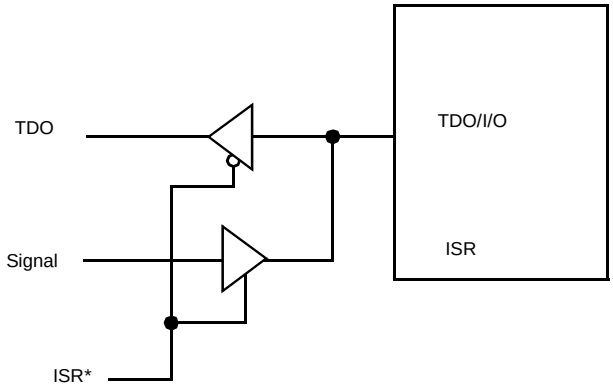


Figure 9. Design for Dual-Function Pins: TDO Used With an Input

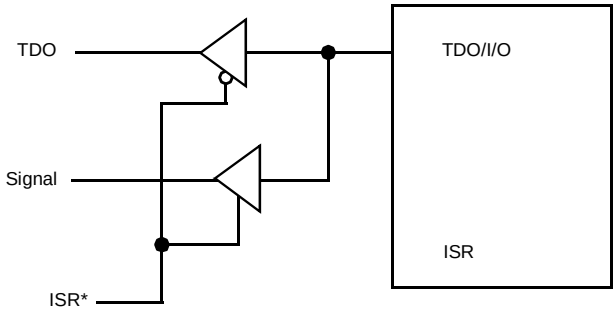


Figure 10. Design for Dual-Function Pins: TDO Used With an Output

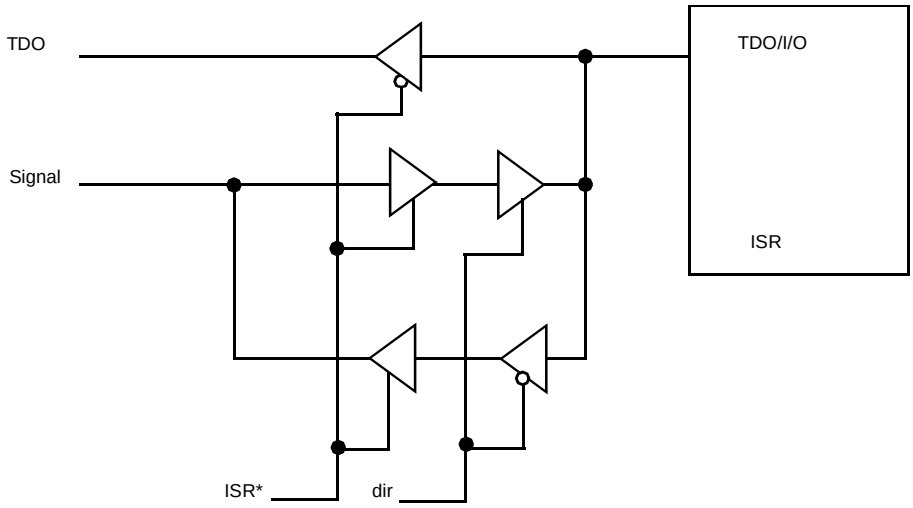


Figure 11. Design for Dual-Function Pins: TDO Used With an I/O

Dual-Function Summary

To summarize this section, there are many ways to accomplish programming using devices with dual-function pins. The easiest way to use the dual-function device is the 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 your *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.

Simple Cascading

Until now, we have talked about programming just a single ISR device in the system. 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 Dual-Function ISR Devices or Single-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 12*). In *Figure 12* 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 15* 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.

Cascading With Other IEEE 1149.1 Compliant Devices

Other IEEE 1149.1 compliant devices can be included in the chain with ISR devices if the device has a JTAG interface, contains the standard JTAG TAP controller state machine, and supports a BYPASS instruction which uses the specified code of all ones for the instruction. The ISR programming software will insert the number of ones necessary to fill the non-Cypress device's instruction register in the appropriate place in the bitstream that is sent. While it is possible to place non-Cypress devices in the same ISR programming chain as Cypress devices if they adhere to the JTAG specification, this is not recommended as it adds complications to the chain for programming the non-Cypress devices. Separate programming chains are recommended. The next topic discusses the handling of the JTAGen pin which has slightly different functionality between the FLASH370i and the Ultra37000 families.

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

Figure 12 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 12* 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

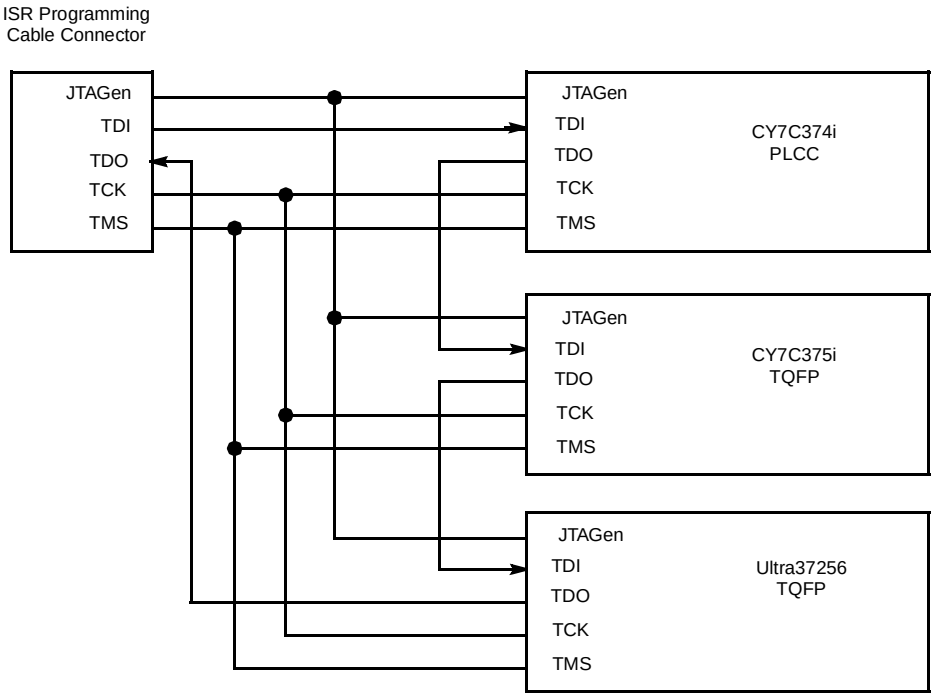


Figure 12. Simple Cascading Example - Single-Function Mode Operation

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. Unlike the FLASH370i family of devices where a TTL HIGH or LOW level on the JTAGen pin enables the I/O functionality of the dual-function devices, the same I/O functionality on the Ultra37000 dual-function devices occurs when the JTAGen pin is driven to a TTL LOW level only. For the FLASH370i it is permissible to actually let the JTAGen pin float since the bus-hold latch would latch either a LOW or a HIGH value. 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 does 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 5* 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 5* to implement this logic. Any of these solutions can be used for multiple Ultra37000 devices ganged together. *Figure 13* 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.

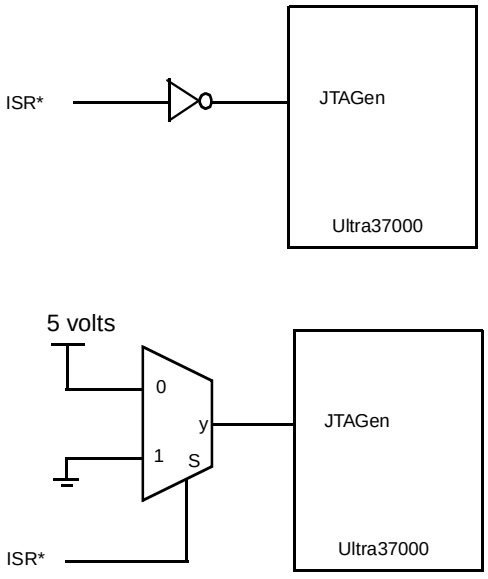


Figure 13. JTAGen Driven for the Ultra37000

Figure 14 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 15* 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.

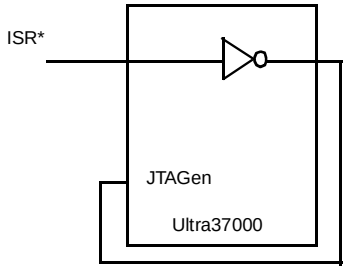


Figure 14. JTAGen Incorrectly Driven by Ultra37000 I/O

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 is addressed in the next section of the note. A few other miscellaneous board-level concerns are also covered.

Other Considerations

Other board-level design issues to be addressed in this note are: the state of the ISR programming cable's pins when not programming and the state of the ISR programming pins when the cable is not connected, bus-hold pin differences between the FLASH370i and the Ultra37000, and handling the 12V signal on the board for FLASH370i devices. These issues are now addressed.

State of the 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. Furthermore, the I/Os (except TDO) are also all three-stated during device programming, that is, when JTAGen is enabling the ISR interface. This allows you to solder ISR devices directly on your board without having to erase them first.

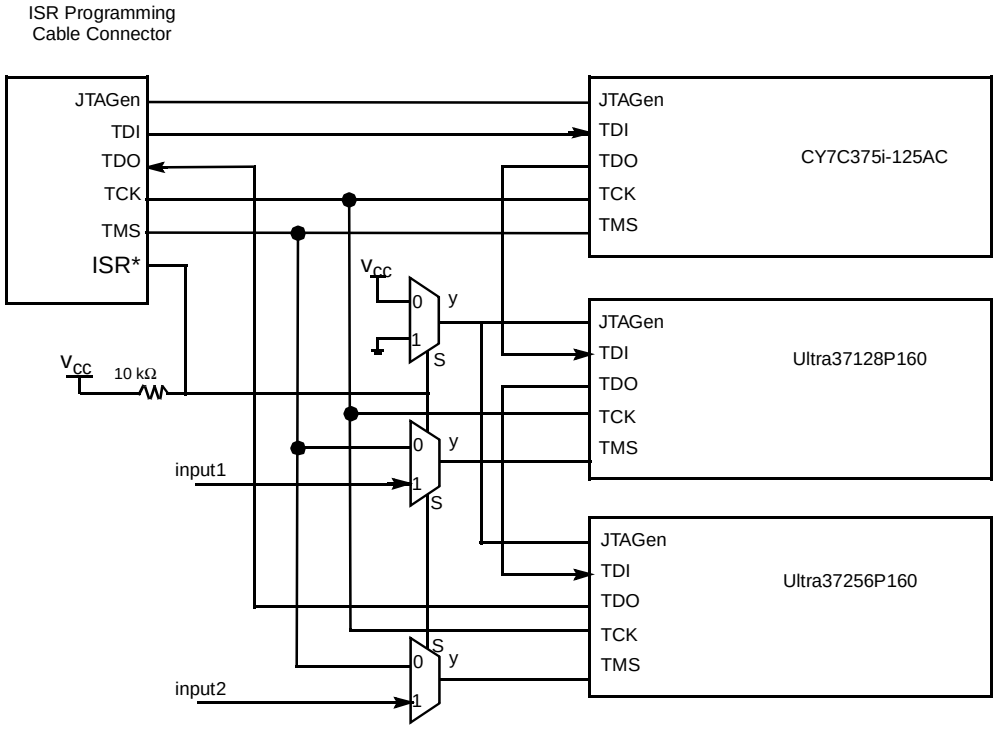


Figure 15. Cascading Dual-Function Ultra37000 and FLASH370i Devices in the Same Chain

It will allow you to power-up your board and program the ISR devices on it without having to worry about their initial, non-blank state causing 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 ISR cable to the board and the parallel port of the PC, turn on the power to the board, and then program the devices for the first time using the ISR programming cable connected to a PC. Since many of the I/Os on the ISR device(s) to be programmed 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 having them also be guaranteed to be three-stated during ISR programming, you are assured that the ISR device will not be also trying to drive those pins. This prevents bus contention, and prevents the ISR devices or other devices on the board from being damaged. When the JTAGen signal disables the ISR interface, the ISR device will start driving some of its output pins, based upon the functionality of the design. For the Ultra37000 single-mode devices, which do not have a JTAGen pin, this occurs when the ISR enable register is loaded with a LOW thus disabling the interface. At this point, it is no different from powering up a board with preprogrammed non-ISR PAL devices, PLDs, or CPLDs devices.

The next section discusses the state of the ISR pins when the ISR cable is disconnected. Also discussed are the differences that exist between the FLASH370i and Ultra37000 families due to differences in the placement of the bus-hold latches only on the JTAG pins. In some cases an external pull-down resistor is needed in the ISR chain specifically for the Ultra37000 devices on the TCK pin.

State of the ISR Programming Pins When the ISR Programming Cable is Not Attached for the FLASH370i Devices

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 OK 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. Because 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 necessarily connect to an ISR I/O 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.

State of the ISR's Programming Pins When the ISR Programming Cable is Not Attached for the Ultra37000 Devices

The Ultra37000 family differs from the FLASH370i family with respect to the connection of the bus-hold latches on the ISR interface pins only. Specifically, the bus-hold latches are disconnected from the JTAG pins TCK, TMS, TDI, and TDO when the JTAGen pin is HIGH, enabling the JTAG port, and connected when the JTAGen pin is LOW, 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 are permanently disabled. The reason for these differences is that the Ultra37000 family supports JTAG Boundary Scan testing and the bus-hold latches, if placed on these pins, can cause significant DC loading on the JTAG drivers to TCK and TMS depending on the source impedance of the drivers and the number of devices connected in the ISR chain. This can occur because some of 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. The reason for this change is conformance to the 1149.1 specification and is necessary to place the JTAG device in a known benign state such as BYPASS 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 a 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 1 kΩ.

State of the ISR's 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, that are always present on the FLASH370i devices, hold the pin to a logic LOW or HIGH level preventing it from floating. The last issue to discuss in this application note pertains only to FLASH370i devices and programming board layout concerns.

Handling the 12V Signal on the Board for FLASH370i Devices

Unlike the Ultra37000, where the JTAGen signal serves only to choose either the ISR interface or the I/O function on the dual function pins, for the FLASH370i devices it is also the high-voltage, low-impedance path required for programming the device. There are two requirements on the JTAGen programming voltage that necessitate special handling when programming FLASH370i devices. The first is that its voltage must be in the range 11.4V $\leq$ JTAGen $\leq$ 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.

Because of the higher than usual current and voltage requirements on the JTAGen signal, the trace on the printed-circuit board connecting the JTAGen pin from the ISR programming cable to the JTAGen 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 JTAGen 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 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 socketed connected by accident, it is advisable and simple to incorporate this decoupling capacitor.

Conclusion

In-system reprogrammability (ISR) in a CPLD has several benefits. It allows engineering development and debugging without having to socket the CPLDs and without having to remove them and reprogram them in a device programmer. This saves time regardless of the package type you decide to use. ISR is especially valuable when you decide to use fine-pitch packages like TQFPs. As before, it allows you to use them without sockets, which means, again, no handling of

devices for reprogramming. Not only does that save time, but in this case, it also avoids the higher potential for bending leads on very fine-leaded devices. Also, by allowing you to solder TQFP packages directly onto a board without sockets, it helps you 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 you to actually make design changes during prototyping, debugging, or field operation and still successfully route to the already-defined pinout, even on designs that are utilizing most or all of many of the device's resources.

This application note shows how to take advantage of the in-system reprogrammability of the FLASH370i and Ultra37000 ISR families of devices by using a cable connected to the parallel port of a PC for programming the CPLDs on the board. This typical usage during development and debugging, which, as described above, is an area where the ISR architecture and routability are particularly useful. This application note explains all of the details of the programming cables and the signals it uses, and it also covers many design techniques and considerations that show how to most easily use the desired capabilities of these parts. These include, description of the logic designs for using the dual-function pins on ISR devices whose programming signals share pins with I/O signals, connecting ISR devices in the programming chain, bus-hold and JTAG enable differences between the FLASH370i and Ultra37000 device families, and tips for handling the 12V programming signal on your board for the FLASH370i devices.