The Basics of One-Wire ISP™ with an ISP-IrDA Example

Introduction

Lattice Semiconductor, the inventor of In-System Programmable devices, has been successfully programming PLDs and CPLDs in-system longer than any other company. There are currently several methods for downloading information to a Lattice ISP device. The most common programming controllers are a PC parallel port, automatic test equipment (ATE) or embedded microcontroller. Each of these methods involve a direct four- or five-wire connection from the source to the ISP target device. With this standard ISP connection, it is possible to control the Lattice device's four or five programming pins. Figure 1 shows a block diagram for a configuration using a PC and an ISP target.

ISP Overview

The ISP target can include a single device or a daisy chain of multiple devices. The required ISP signals include $\overline{\text{ispEN}}$, SDIN, MODE, SCLK and SDO. For those familiar with the four-wire IEEE 1149.1 boundary scan TAP controller, the TAP controller signals TDI, TMS, TCK and TDO can be used interchangeably with the ISP signals SDI, MODE, SCLK, and SDO respectively. Most Lattice ispLSI® devices use the ISP Enable ($\overline{\text{ispEN}}$) pin to determine whether the device is in Edit Mode (ISP programming mode) or Normal Mode (normal device operation mode). The ispGAL®22V10 and ispGDS™ enter edit mode when SDI is a logic high and a SCLK is received.

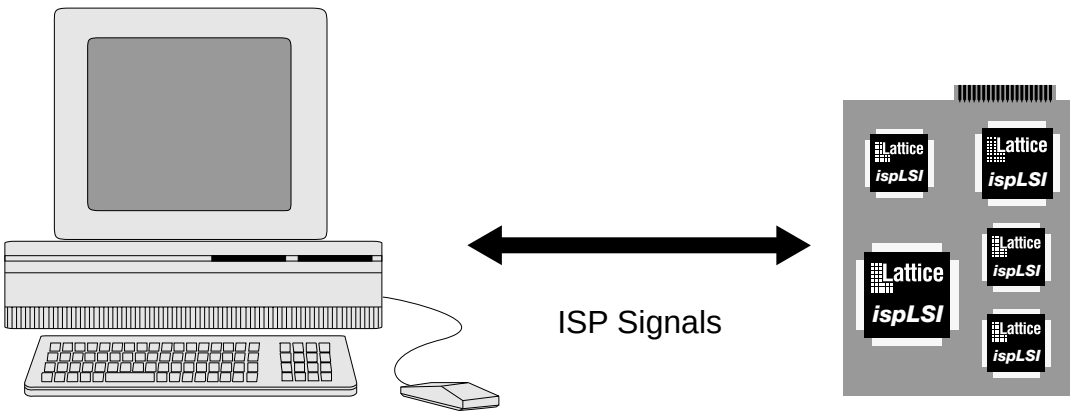
The Serial Data In (SDI) pin functions as both a data input to the serial shift registers in the device and one of two control pins for the programming state machine. The Mode pin combines with SDI to control the programming state machine. The Serial Clock (SCLK) pin is used to clock the internal serial shift registers and the ISP state machine. Serial Data Out (SDO) is connected to the output of the internal shift registers. If you are unfamiliar with the advantages of Lattice ISP, refer to the latest edition of the Lattice Semiconductor Data Book or CD-ROM for additional information.

ISP Innovation

With years of ISP experience, Lattice has developed solutions for many challenges encountered when implementing in-system programming. Customers frequently require unique solutions. For example, programming times of less than 20 seconds became essential for customers programming with Automatic Test Equipment. The solution to that challenge led to the development of Lattice Turbo ispDOWNLOAD™ programming. Turbo programming is a method in which a daisy chain of devices is programmed in parallel. The programming time of the entire daisy chain is accomplished in approximately the time it takes to program the largest device in the chain. What started as an ATE-driven solution has carried over to all programming platforms.

ISP-IrDA arose from similar customer inquiries. The Infrared Data Association (IrDA) has standardized infrared data communication. ISP-IrDA incorporates ISP with standard IrDA transmission. While many companies are still experimenting with the first steps of a four- or five-wire ISP solution, Lattice has taken ISP to the next level with one-wire ISP.

Figure 1. Standard Four- or Five-Wire ISP



The Basics of One-Wire ISP with an ISP-IrDA Example

In field upgrade situations where I/O is needed for other uses or there is a need for remote access, running a four- or five-wire connection is not practical. For example, telecommunications companies want the ability to reprogram boxes on top of telephone poles without climbing poles and exposing the electronics to the elements. With ISP-IrDA, Lattice makes this a reality.

Infrared data communication is quickly becoming a popular remote data communication standard. The IrDA standard defines protocol for wireless data communication. Essentially, IrDA is half-duplex serial communication. In the first section of this application note, the basic theory involved for changing the five-wire interface to a one-wire interface is presented. The following section will describe the IrDA model in detail. After reading this application note, the reader will have a basic understanding of how to apply ISP devices for use with many remote communication standards.

One-Wire ISP Theory

The basic theory behind converting the standard four- or five-wire ISP to one-wire ISP is quite simple. A standard parallel port has eight data lines in and eight data lines out. Serial communication typically uses eight data bits. ISP uses four signals out and one signal in. Everything is synchronized by SCLK. Rather than having four or five signals connected in parallel, it is possible to compact the these signals into a serial byte. This conversion is the simplest parallel-to-serial conversion, although not necessarily the most efficient. Figure 2 shows the pinout for the ispDOWNLOAD cable and illustrates how the parallel port is used.

Figure 2. ispDOWNLOAD Cable

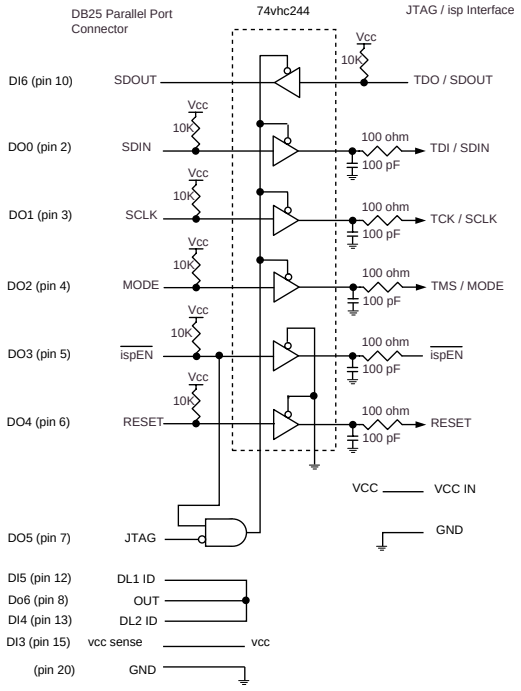


Figure 3. ISP Serial Byte

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
SDI	SCLK	MODE	$\overline{\text{ispEN}}$			SDO	

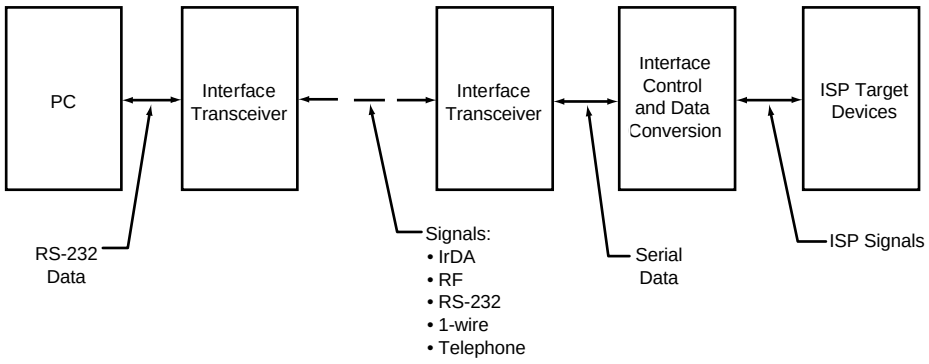
This configuration is used because the bit order format agrees with the parallel version. Each ISP signal corresponds to a specific bit location within the serial byte. This method does not allow for rapid data transmission since it takes a serial byte to change any one ISP signal. The advantage of this structure is that it is easy to decode the serial byte at the ISP target.

Since all the ISP data fits into a serial byte, there is simply the conversion from parallel-to-serial and then serial-to-parallel. Figure 4 illustrates the additional overhead required for one-wire ISP compared to standard ISP as illustrated in Figure 1.

The PC can convert parallel ISP signals to serial ISP bytes and initialize and control the interface transceiver. A controller is required in front of the ISP target. The controller must be able to initialize and control the interface transceiver. It must also convert the serial ISP bytes back to parallel ISP signals.

A main area of concern in changing from bi-directional parallel communication to half-duplex serial communication is avoiding data conflicts on the one-wire. However, there are three extra bits within the serial byte available (see Figure 3). The extra bits can be used for handshaking or other user-specific functions. When programming, a majority of the data is sent to the ISP target. SDO is the only signal returning from the ISP target and is used to read the device ID, verify programming and connect to other devices in the daisy chain. To reduce the complication involved with managing the half-duplex serial line, Bit 7 is used to request SDO. No additional hardware

Figure 4. One-Wire ISP Interface



overhead is required to wait for the line to become free. If Bit 7 is set to a '1', then one SDO bit is read and sent back by the interface controller. For example, when the PC wants to read the device ID, it looks for the first bit of the device ID. The PC sends out a request for SDO. Next, the PC waits for the SDO bit to be sent by the interface controller before sending the next serial byte. The next string of serial bytes shifts the next bit of the device ID to SDO. The PC again requests SDO and waits for it to return. This process continues until the device ID is read. Therefore, not a lot of additional hardware is required. Figure 5 shows the final one-wire serial byte configuration.

While quite simple in theory, converting to one-wire ISP requires knowledge of the transmission medium, interfaces and ISP programming software. Software must be modified and a hardware interface controller must be built. In the following section, these challenges have been removed with a detailed description of the ISP-IrDA interface.

ISP-IrDA Interface

The previous section describes the basic theory of conversion to one-wire ISP. In this section, the ISP-IrDA model will be discussed in detail. Covered are two areas of concern in converting from the standard four- or five-wire ISP interface: the modification of the ispDOWNLOAD software and the addition of the interface controller in front of the first ISP device. Figure 6 shows a block diagram for ISP-IrDA.

Figure 5. One-Wire Serial Byte

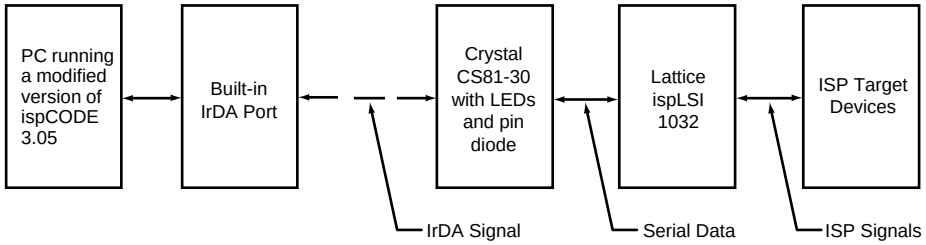
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
SDI	SCLK	MODE	$\overline{\text{ispEN}}$			SDO	Request SDO

Software

Lattice offers several versions of ispDOWNLOAD software for use in various applications or with different platforms. ISP Daisy Chain Download software is a precompiled version for Windows that includes support for turbo and one-at-a-time device programming. It has a utility to assist in creating programming vectors for various ATE manufacturers. To support ISP programming through embedded controllers or any other platform, Lattice supplies C source code called ispCODE™ software. ispCODE compiles to a DOS command line utility for Turbo ispDOWNLOAD programming a daisy chain of devices through a PC parallel port. It requires that the user specify an ispSTREAM™ file to program each device at the command line. The ispSTREAM file format is a unique Lattice format made up of the compressed JEDEC files plus instructions that the software uses to program the daisy chain. A user must also write a short text file called a DLD file showing the number of devices, their order in the chain, the operation to be performed on each device and the JEDEC file (if the operation requires one). Running *dld2isp.exe* on this DLD file creates the ispSTREAM file. Please refer to the Lattice Semiconductor Data Book or CD-ROM for more information on ispCODE software.

Obviously, some software changes are required in order to program a daisy chain of devices over an IrDA link. Fortunately, most newer laptops include a built-in IrDA port that is accessed as a secondary serial port. This simplifies the initialization at the PC end because the IrDA transceiver is automatically initialized when the serial port address used by the IrDA port is configured. With ispCODE version 3.05 or later, many of the modifications are minor. In the main body of ispCODE, the Vcc and cable sensing for the parallel port connector is removed. Selections for a COM port and baud rate are

Figure 6. ISP-IrDA-Based Programming Block Diagram



added. Once the COM port and baud rate are known, the proper UART is initialized for the given baud rate plus eight data bits, no parity, and one stop bit. Variables are added to *lattice.h* for COM port addresses, UART flags, UART registers and the SDO request bit.

The larger changes come in the two functions for data output and data input. The *isp_setpin* function is used to change one ISP signal at a time and output the change to the data port. Because of the relatively slow transmission speed of a serial port compared to a PC, care has to be taken to not overwrite any data before it can be sent. It is important to check that the UART's data transmission register is empty before writing to it. The LSR UART flag signifies when the data register is empty. Once the data register is ready, the serial byte can be written to the port. It is best to wait for the byte to be sent before continuing by waiting for LSR to signal that the data register is empty again.

The *isp_SDO* function is called when one bit from SDO is required. With five-wire ISP, SDO is always available because it has a dedicated line. The isp_SDO function had to be changed from a simple read of the SDO line to a request for SDO to be sent followed by a read of that data. The flow for the function changed to clearing the UART receive buffer, sending out a byte with the request_SDO bit (Bit 7) high, waiting for a byte to return, reading that byte, extracting the SDO bit (Bit 6) from that byte, and resetting the request_SDO bit back to '0' for the next byte. Because IrDA streams can be interrupted easily by a drastic change in lighting or a physical blockage of the signal, it is necessary to add a time-out when waiting for SDO to return. A minimum of two seconds is given to receive SDO before timing out and exiting the program.

Hardware

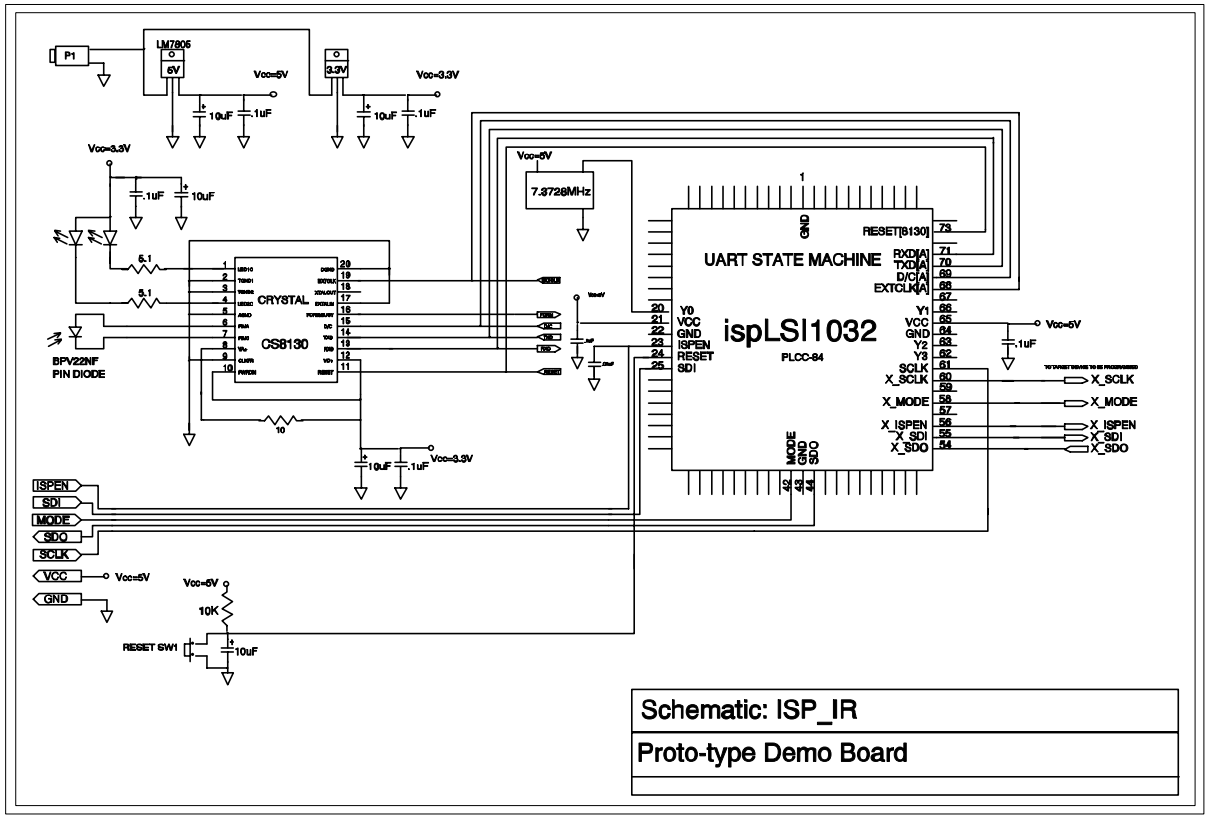
Figure 6 shows the block diagram for the ISP-IrDA model. The format of the serialized ISP signals was defined in the previous section. The preliminary feasibility tests done on the IrDA interface used a 8051 microcontroller for the interface control and serial-to-parallel conversion. However, the decision to use a Lattice ispLSI 1032 was made for several reasons. The large volume of data being transmitted caused the programming time to be directly related to the baud rate. The maximum baud rate for most systems is 115,200 baud. The logic to initialize the Crystal CS8130 IrDA transceiver, convert the serial-to-parallel, and handle the SDO contingency is quite simple. Also, the ispLSI1032 in the thin quad flat pack (TQFP) package is small and uses little PCB space.

The interface controller is made up of two state machines, as well as serial transmit and receive shift registers. At power-up, the controller must initialize the CS8130 at 9600 baud. Once completed, it waits for serial data to come in from the CS8130. After a serial byte has been received, the request_SDO bit must be checked. If SDO is not required, then the byte contains programming signals. Bit 0 through Bit 3 contain the ISP signals and are sent out to the ISP target devices. If SDO is required, the controller reads SDO from the ISP target, places the SDO bit into Bit 6 of a serial byte and sends the byte to the CS8130.

In the IrDA example, the ispLSI 1032 initializes the CS8130 by turning on transmit and receive, setting Ir receive sensitivity to 23.4nA, and changing the baud rate to 115,200 baud. Additionally, the ispLSI 1032 uses a 7.3728MHz clock which is divided by two for the CS8130 clock. The schematic for the IrDA module is shown in Figure 7.

Compared to standard five-wire ispTURBO programming from the Windows environment, the ISP-IrDA-based programming in this example is approximately three times slower if running at 115,200 baud. Lattice's ispLSI demoboard is a small PCB with a four-device daisy chain, five seven-segment LEDs and ISP connections. The

Figure 7. Schematic ISP-IrDA Module



devices include an ispLSI 2064-100LT, an ispGAL22V10C-15LJ, an ispLSI 1016-60LT and an ispGDS22-7J. The ISP-IrDA module is capable of plugging onto the demoboard's eight-pin AMP connector while the demoboard is powered up. If using a laptop with an IrDA port, you are now ready to download your design files. Contact your local Lattice sales office for a demonstration.

Summary

Many ideas for making ISP-IrDA practical for a production environment have already been uncovered. Imagine a hand-held IrDA download box in front of a cart full of boards. By typing in the code for a particular board, the technician is able to program the devices on that board quickly and easily.

Building a more intelligent controller or incorporating memory to speed data transfer in order to achieve programming times similar to those of the parallel five-wire ISP are also being investigated. Other remote transmission protocols such as modem-to-modem and RF are being developed. If system manufacturers are able to update their remote hardware with only a modem and a phone line, imagine the savings in time and human effort.