ispCODE™ Software

Source Code for In-System Programming of the ispLSI®, ispGAL® and ispGDS™ Families

Features

- **C-LANGUAGE SOURCE CODE FOR IN-SYSTEM PROGRAMMING OF THE ispLSI®, ispGAL® and ispGDS™ FAMILIES**
  - **— Simplifies In-System Programming**
  - **— Pre-Defined Routines for Common Programming Functions**
  - **— Extensively Commented Code Provides Complete Reference**
  - **— Easy Modification Saves Valuable Time**
  - **— Supports Programming of Multiple ispLSI Devices on Individual Boards**
- **ACCEPTS PROGRAMMING FILES FROM pLSI® AND ispLSI® DEVELOPMENT SYSTEM**
  - **— Supports pDS® and pDS+™ Software**
  - **— Supports ispLSI 1000/E, 2000, 3000 and 6000 Families**
- **PORTABLE TO ANY HARDWARE PLATFORM**
  - **— Adaptable to Any Hardware Interface**
  - **— UNIX Systems, PCs, Testers, Embedded Systems**
  - **— ANSI-Standard C for Portability**
- **GENERATES ispSTREAM™ FORMAT FOR GREATER EFFICIENCY**
  - **— Bit-packed File Format for Storing JEDEC Fuse Map**
  - **— Requires Less Than 1/8 the Storage Space of a Standard JEDEC File**
  - **— Ideal for Use in Embedded Systems**
  - **— Includes Checksum To Assure Data Integrity**
- **USER ELECTRONIC SIGNATURE (UES) SUPPORTED**
  - **— Provides Data Storage Area In Device**
  - **— Facilitates User Identification of Program for Secured Devices**
  - **— Automatic Counter Records Number of Programming Cycles**

Introduction

The ispCODE software from Lattice Semiconductor Corporation (LSC) is designed to facilitate in-system programming of ISP devices on customer-specific hardware platforms. The ispCODE works with Lattice Semiconductor's pDS and pDS+ software to give users a powerful, fully integrated tool kit for developing logic designs and programming ISP devices "on-the-fly."

After completion of the logic design and creation of a JEDEC file by the pDS or pDS+ software (see figure 1), in-system programming can be accomplished on customer-specific hardware: UNIX systems, PCs, testers, embedded systems (see figure 2). The ispCODE software package supplies specific routines, with extensively commented code, for incorporation into user application programs. These routines provide users with flexible, easy-to-use program modules which support the programming of a single device or multiple devices on a board.

ispCODE Software

The ispCODE software consists of source files containing routines for performing all the functions needed to control the programming of Lattice Semiconductor in-system programmable devices. These routines are provided as fully-commented source code for easy inclusion with any software written in the industry-standard C language. These source code routines were designed from the ground up to be easily portable to any system that has an ANSI-standard C compiler. The majority of the code is completely independent of the hardware platform and rarely requires modification. All hardware dependent portions of the code are related to how the output ports are driven. The code supports the programming of multiple ISP devices in a daisy chain configuration.

A compiled version of the ispCODE is provided to demonstrate how to use the ispCODE 'C' source routines. The compiled code, TURBO.EXE, programs the device(s) through the PC parallel port. By making small changes to the hardware-specific source file and recompiling the 'C' routine, a command utility like TURBO.EXE can be created. Furthermore, the 'C' routines on the .EXE file can be integrated into user interface routines which can be customized for the end application.

The example program and the hardware-specific code are written to run on IBM PC or compatible microcomputers. The example program uses the standard PC parallel printer port to provide the interface to the device's in-system programming pins (see figure 3). The pinout is compatible with the isp Engineering Kit Model 100.

LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A.
Tel. (503) 681-0118; 1-800-LATTICE; FAX (503) 681-3037; http://www.latticesemi.com

Customizing ispCODE

ispCODE is intended to help customize the ISP programming process if the standard Lattice Semiconductor IBM-PC compatible software is unable to meet specific application needs. Some examples of non-standard needs are:

- Using a platform other than the parallel port of an IBM-PC
- Using a customized user interface
- Using the UES feature of ISP devices for board serialization (adding serial numbers to the UES, uniquely identifying the board).

If programming using the standard PC parallel port, Lattice Semiconductor recommends the use of either the Windows or DOS based download software. ISP Daisy Chain Download is provided with all new pDS and pDS+ shipments and software updates.

The files that follow are available as part of the ispCODE set:

DOS File Name	Descriptions
dld2isp.exe	Converts JEDEC files to ispSTREAM format
express.exe	Command line programming utility
ispcode.c	C source code
lattice.h	Header file for use in ispcode.c
readme.txt	A text file with information updated since this manual
turbo.exe	Compiled ispcode.c

Design Flow Using ispCODE

ispCODE reads the Lattice Semiconductor Bitstream file (ispSTREAM™) format instead of directly using JEDEC files. *dld2isp.exe* accepts a DLD file and converts the JEDEC files listed in the DLD file to a single Bitstream file. The DLD file is the file that defines the device ordering, device type, and JEDEC files for a daisy chain configuration of ISP devices. The Bitstream file should be created on a PC and transferred to the ISP device programming platform.

Figure 1. Using the ispCODE Software

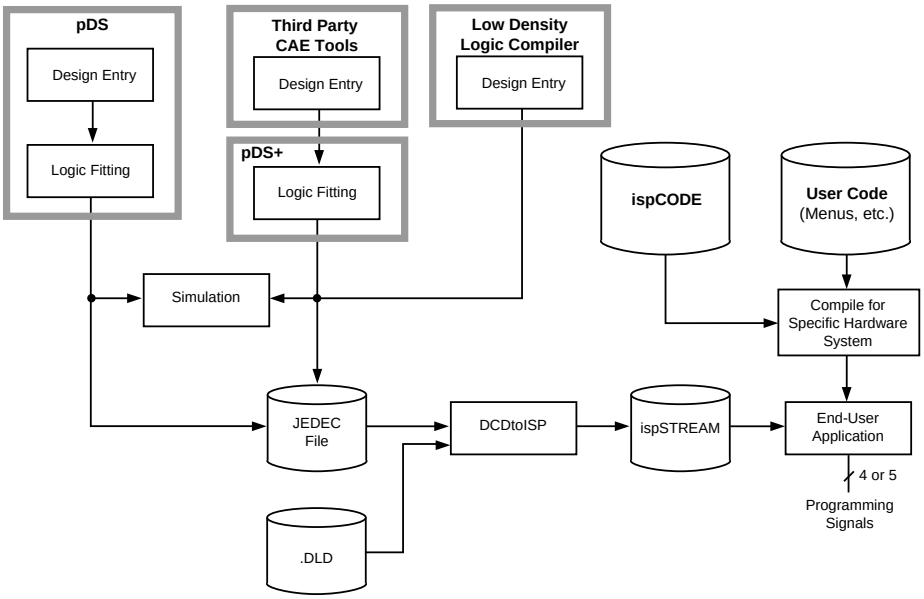


Figure 2. Configuring an ispLSI Device from an On-board Microprocessor

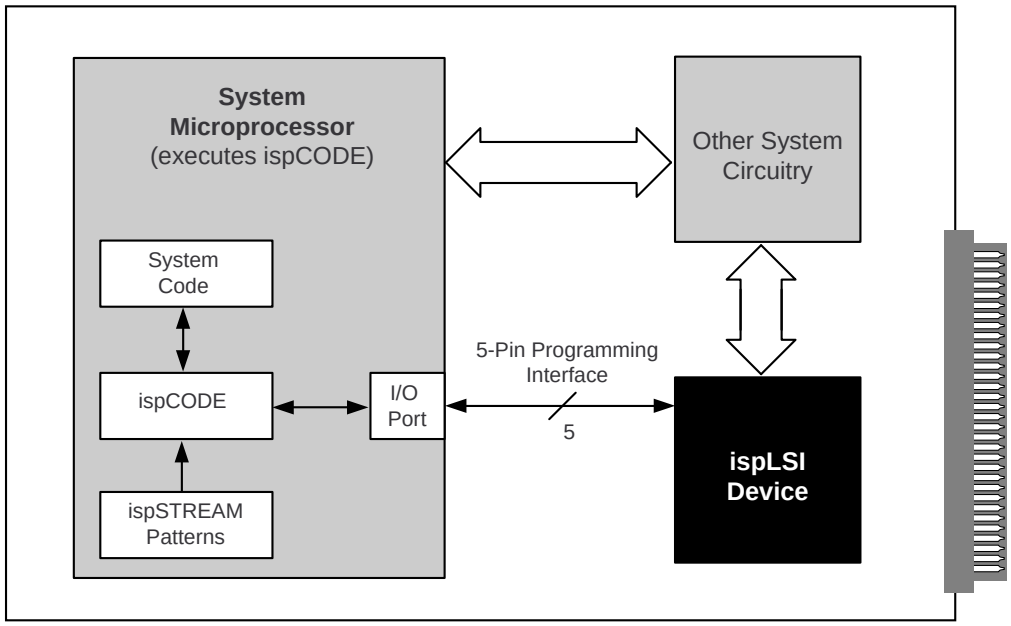
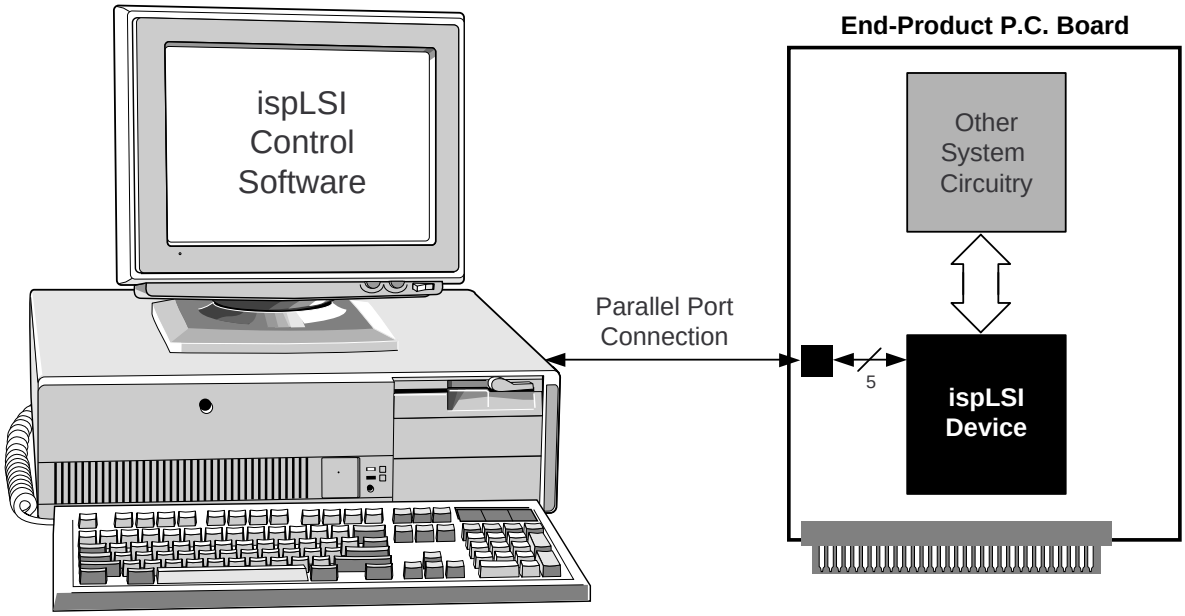


Figure 3. Configuring an ispLSI Device from a Remote System



If performing ISP programming debugging, Lattice Semiconductor recommends the use of the evaluation/design download option of the Windows or DOS based download software. If a command line version is needed, then *express.exe* supports all the same operations as the evaluation/design download of the Windows or DOS based download.

The DLD file format is the Lattice Semiconductor standard approach to defining the configuration of a daisy chain. The format of the file is straightforward: each line contains the device type, followed by an operation code, followed by a JEDEC file name. An example is shown below:

```
GDS22     PV   GDS22.jed
1016      PV   1016.jed
22V10     PV   22v10.jed
1032      PV   1032.jed
```

The previous DLD file is illustrated in Figure 4.

This file can be created manually with an ASCII text editor, or automatically with Lattice Semiconductor download software. The following rules apply when creating the DLD file:

1. The lines in the DLD file correspond to the position of the device in the chain with the first device (the device whose SDI is connected to the hardware programming port) corresponding to the first line in the file.
2. The maximum number of devices supported in a daisy chain is currently set to 60 in all Lattice Semiconductor download tools.
3. Only the following set of operations can be used in the DLD file when using *dld2isp.exe* to generate the ispSTREAM file (.isp file) for ispCODE V3.02.

Operation	Description
PV	Program and verify
PU	Program and verify with UES data from sources other than the JEDEC files
V	Verify only
NOP	No operation, by-pass device

4. The valid device names are currently:

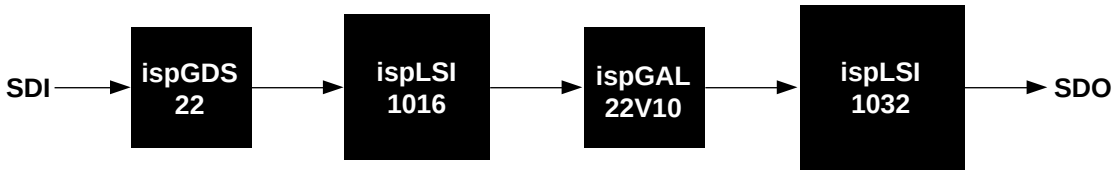
GDS14	1024	2032
GDS18	1032	2064
GDS22	1032E	2096
22V10	1048	2128
1016	1048C	3256
1016E	1048E	

To generate the ispSTREAM file, at the DOS prompt type: **dld2isp** *design_name.dld.*

The ispSTREAM file generated will be *design_name.isp.*

If the daisy chain is too large, *dld2isp.exe* displays the message “Not Enough PC Memory”. Lattice Semiconductor can supply, upon request, a *dld2isp.exe* that uses extended memory or it is possible to use a multiple DLD strategy. The multiple DLD approach requires breaking the programming operation into two or more Bitstream files, and programming different sets of devices with each Bitstream file. Using the NOP code to skip devices is the easiest way. If two Bitstream files are needed, then create two DLD files. In the first DLD file, insert NOP’s for the first set of devices in the original DLD file and in the second DLD file, insert NOP’s for the second set of devices. The first Bitstream file will only program the

Figure 4. ISP Daisy Chain



second set of devices (since the first set is NOP), and the second Bitstream file will only program the first set of devices.

The "PU" operation is available only from ispCODE version 3.01 or later. This operation allows customization of the powerful UES feature of ispLSI devices for specific applications and manufacturing flows. This field causes ispCODE to save the UES information in **char *ues**, a string that contains all the device UES information. The following shows the procedure to customize the UES:

1. Use the PU code in the DLD file for each device that will have its UES changed

2. Modify the **program_ues** routine to modify the UES with the desired value.

The **char *ues** contains the UES string that was read from the devices prior to their erasure. The UES will be written back out to the devices during the **program_ues** routine. If changes to the UES are desired, then change it at a point in the code prior to the *printf* statement in the first few lines of that routine.

Using the UES

The UES is a user-defined section of bits in the Lattice Semiconductor device that can store any sort of information, such as a board serial number or device version number. The UES is normally part of the device JEDEC file. Although the JEDEC file is an ASCII text file that can be edited with a text editor, Lattice Semiconductor recommends the use of the UES editor in the Lattice Semiconductor Windows or DOS based download software. Using the download software to edit the JEDEC file insures that any necessary corrections to the pattern and transmission checksums will be made (the UES of the ispGDS devices and ispGAL devices is included in the pattern checksum calculation, but not for ispLSI devices).

There are considerations when manually adding the checksum. The JEDEC standard allows UES data to be entered in ASCII, HEX, and binary formats and inserted directly after the pattern checksum. A text editor can be used to edit the JEDEC files directly. Please note that while the pattern checksum is not affected, the transmission checksum will be incorrect. The transmission checksum should be changed to 0000 to comply with JEDEC standard. For example:

• UES in ASCII Format:

```
CXXXX*
UA UES IN ASCII*
<etx>0000
```

•UES in HEX Format:

```
CXXXX*
UH01234567890ABCDEF*
<etx>0000
```

•UES in Binary Format:

```
CXXXX*
U010100010*
<etx>0000
```

• This is a JEDEC compatible fuse file:

```
<stx>
DESIGN NAME:GAL61-3.lif
PART NAME  :ispLSI1048-70LQ
CREATED BY :PDS+ FUSEGEN Version 2.2
CREATED DATE:Wed Dec 22 14:23:20 1995
*QP120
*QF57600
*G0
*F0
*L00000
11......
........
11......
*CD079
uaA_MSG   <- Insert the U field here.
<etx>0000
```

Command Line Programming Utility

express.exe is a DOS command line utility that allows sequential daisy chain programming of devices. It does not support parallel programming. A standard DLD file must be created before running *express.exe*. Proper usage with all the available switches for *express.exe* can be seen by running it from the DOS prompt. The basic usage is:

express [*drive:*][*path*] *dld_filename*

The operations supported by *express.exe* are:

Operation	Description
PV	Program and verify
V	Verify only
NOP	No operation, by-pass device
C	Calculate the pattern checksum of the device
E	Erase the device only
RS	Read the pattern and UES from the device and save it in a JEDEC file.

ispCODE Source Code

Version 3 of ispCODE is set of C routines to read a Bitstream file and program a chain of ISP devices simultaneously. Since all the complex work is done by *dld2isp.exe* when the Bitstream file is created, the resulting ispCODE is relatively simple. Most of the code is a state machine which parses the Bitstream file. This state machine portion should not be modified. The only portions of ispCODE that can be modified safely are the routine to read and write from the parallel port and the routine to implement timing.

ispCODE uses the standard routines **inp** and **outp** to read from and write to the port. To redirect the I/O to a different address, create routines similar to the **inp** and **outp** routines, then link them to ispCODE. One way to do this is to make a global search and replace of "inp" and "outp" with the new I/O routine names. By renaming "inp" and "outp" to these new names, all I/O will be redirected through the new functions.

If modifying the timing routine, it is critical to meet the minimum pulse requirements specified for the different ISP devices. ispCODE calls a single routine, **pulse_width**, whenever a delay needs to be made. This routine is passed an argument that indicates the number of milliseconds to wait. Modification may be needed for this to work in a non-PC environment. It is important that the modified version of **pulse_width** guarantees that the wait period is at least as long as the argument specified. It is not critical if the pulse width is exceeded by a few hundred milliseconds. A lot of the calibration procedures the routine runs through is used to compensate for the problems that occur when running as a DOS window from Windows. If not running as a DOS window is, it may be possible to simplify the timing procedures. Of course, it is critical to insure that the minimum pulse width times are met, regardless of the method used. If minimum pulse widths are not met, device programming may not be reliable. Note that none of these routines guarantee that a pulse width will not be exceeded. If running as a DOS window or Windows application, the routine can be interrupted by another task, and the pulse time may be larger than specified.

The Borland compiler supplies a DOS routine called **delay** that is accurate to within 1 ms. The **pulse_width** code can instead be replaced with a call to **delay PROVIDING** that the application is never run as a DOS window. Lattice Semiconductor has measured the **delay** function and proven it does not work reliably when used in an application that is running as a DOS window. It is possible to put a check in one's code to help insure it is not run as a DOS window. When running as a DOS window under Windows 3.1, the environmental variable "windir" is set (as lowercase) by Windows. Therefore, by using getenv("windir") to see if the code is running as a DOS window, it is possible to insure that **delay** is not used in that case.

If running only under the Windows environment, Windows 3.1 provides the multimedia services to give high resolution timing under Windows. If compiling as a Windows application, then include the "mmsystem.h" header file, and implement a delay function like this:

```
start_time=time(NULL);
      // get starting time in seconds
timeBeginPeriod(1);
      // set to one millisecond
current_time=timeGetTime();
      // get the current value
         while((timeGetTime()-
current_time)<delay_time){NULL;
      // hog cpu time until finished}
timeEndPeriod(1);
      // free up this timer
```

A somewhat more complicated way of controlling timing when running as a DOS window is to use the Virtual Timer Devices(VTD) services available from Windows. Note that this procedure will not work when running strictly as a DOS application. Again, one can use the getenv("windir") approach explained above to see if the program is running as a DOS or DOS window application. A 32-bit time count, incremented every millisecond, is available from the VTD services by using an assembly routine like the following:

```
.MODEL large
        .DATA
vtd_addr        DD 0
        .CODE
        .386
        PUBLIC _read_timer_doswin

_read_timer_doswin     PROC

;
; this routine avoids the problems with
Windows virtualizing
; the timer ports. Instead, this routine
uses the VTD to get a 32
; timer value.
;
```

```
       mov ax, 1684h                    ;
get VTD address
       mov bx, 5h
       int 2fh
       mov word ptr [vtd_addr],di       ;
save the address
       mov word ptr [vtd_addr+2],es
       mov ax, 0101h                    ;
get current system
                                        ;
time in ms
       call DWORD PTR [vtd_addr]

       ;
       ; return the values in eax in dx,
ax
       ;

       mov edx, eax
       shr edx, 16                      ;
return 32 bit time count
       ret

_read_timer_doswin       ENDP
       END
```

An example C program that uses this routine is shown below:

```
#include <stdio.h>
#include <dos.h>
void delay(unsigned int);
extern "C" unsigned long int
read_timer_doswin(void);

// use extern to prevent name mangling,
if C++ compiler used

main (){
int i;

// generate an 80ms square wave on the
parallel port

     for(i=0;i<=1000;i++){
          outportb(0x378,0xff);
          delay(80);
          outportb(0x378,0x00);
          delay(80);
     }
}
```

```
void delay(unsigned int wait_ms){
unsigned long int start,stop;
     start=read_timer_doswin();
     printf("%lu\n",start);
     while (read_timer_doswin()-start <
wait_ms){
          NULL;
     }
}
```

Devices Supported

The ispCODE library of routines will support all LSC in-system programmable devices. As new devices are developed and released, the ispCODE library will be updated to include them.

Hardware Requirements

The ispCODE routines are designed to be portable to any hardware platform with an ANSI C compiler available. The code is written such that accessing the pins of the Lattice device is done by writing to a memory or I/O port address.

In addition to driving the pins, a method of controlling timing in the millisecond range is required. The best approach for this is a hardware-based method, such as a timer chip that can be read. Most micro controllers have a timer built in, and most other systems have some way of keeping time that may be used. The ispCODE source files include an example of reading the timer chip on a PC to accurately time the programming pulses.

ispCODE Ordering Information

A copy of the ispCODE is included with all LSC Fitter purchases.

Technical Support Assistance

Hotline:	1-800-LATTICE (Domestic) 1-408-428-6414 (International)
BBS:	1-408-428-6417
FAX:	1-408-944-8450
email:	apps@latticesemi.com

LATTICE SEMICONDUCTOR CORPORATION
5555 Northeast Moore Court
Hillsboro, Oregon 97124 U.S.A.
Tel.: (503) 681-0118
FAX: (503) 681-3037
http://www.latticesemi.com

November 1996