DirectC for ProASIC^{PLUS} v1.2

User's Guide for Microprocessor Programming of ProASIC^{PLUS} Devices

Actel®

Windows®

Actel Corporation, Mountain View, CA 94043-4655

Printed in the United States of America

Part Number: 50200008-2

Release: September 2006

Trademarks

The title of this user's guide has been changed down from v1.3 to v1.2 to stay consistent with the current version number of the software.

Table of Contents

Introduction

The ProASIC^{PLUS} family of devices, Actel's second generation of Flash FPGAs, offers enhanced performance over Actel's ProASIC family. ProASIC^{PLUS} devices combine the advantages of ASICs with the benefits of programmable devices through nonvolatile Flash technology.

Because the ProASIC^{PLUS} FPGA is reprogrammable through JTAG, you can program it on various programmers or internally within the system (also known as internal ISP). To perform in system programming (ISP) for the FPGA, the system must contain control logic (a microprocessor), memory to store the programming instruction and the programming data (see Table 1 for memory requirements), and a voltage generator to generate the programming voltages. Figure 1 shows a local system overview.

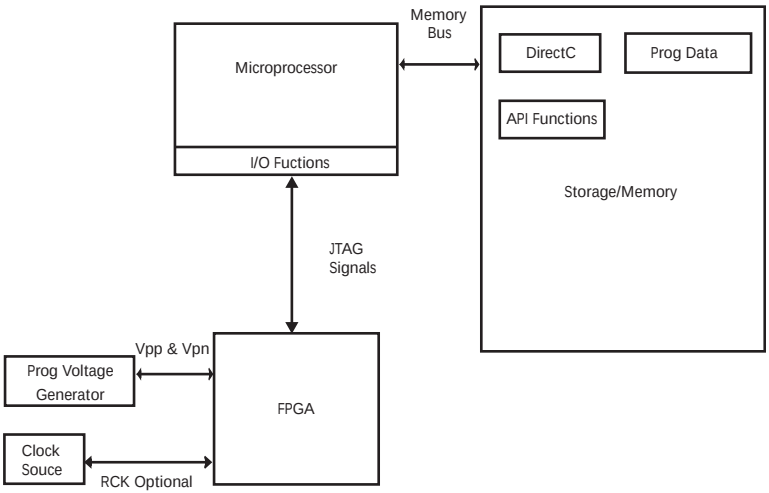

Figure 1. Local System Overview

Table 1. Memory Requirements

Code/STAPL File	Required Memory
DirectC	100KB
APA075	100KB
APA150	200KB
APA300	250KB
APA450	350KB

Table 1. Memory Requirements (Continued)

Code/STAPL File	Required Memory
APA600	700KB
APA750	1MB
APA1000	1MB

The DirectC solution is specifically designed for programming Actel's ProASIC^{PLUS} family of devices using a Microprocessor. DirectC is a set of C code that contains the ISP programming algorithm for APA devices. You can use DirectC by making minor modifications to the source code, adding the necessary API, and compiling the source code and the API together to create a binary executable. The binary executable is downloaded on to the system along with the programming data file (see Figure 2). DirectC uses a STAPL file (generated from Designer) as the programming data file. Even though the STAPL file also contains the programming algorithm, DirectC ignores it and uses only the programming data.

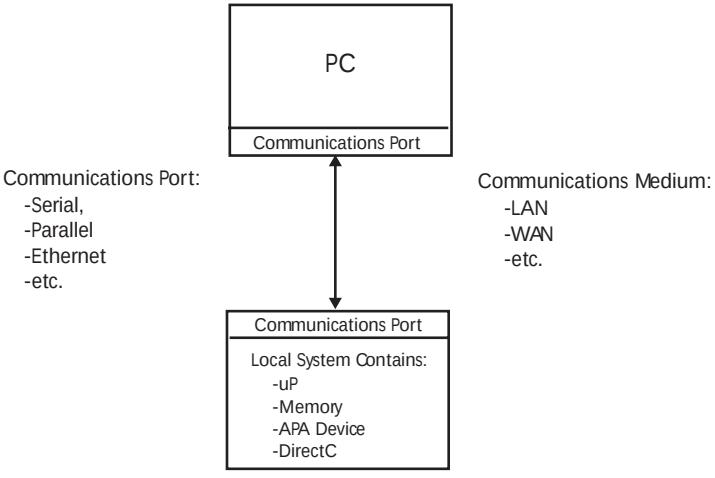

Figure 2. System Overview

Since programming time depends on microprocessors, the DirectC solution has a greater speed advantage on smaller and slower microprocessors/systems. DirectC is specifically targeted and optimized for APA devices, and therefore produces a much shorter programming time than a generic programming solution like the STAPL player. With DirectC, the algorithm is embedded into the source and compiled. Therefore, if the programming algorithm changes you can obtain a

new DirectC source ((http://www.actel.com/custsup/updates/directc/index.html)) from Actel, recompile the source, and download the binary executable on to your system. See Figure 3 for flow diagrams and algorithm changes for the STAPL player and DirectC solutions.

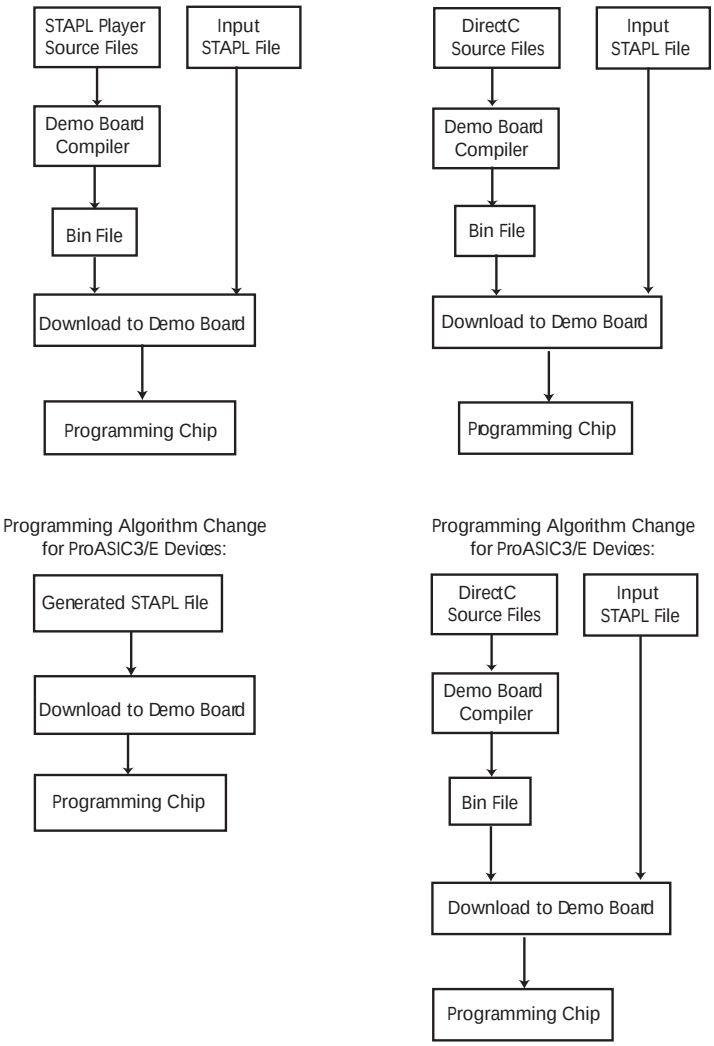

Figure 3. STAPL Player and DirectC Flow and Algorithm Change

Document Organization

This guide provides detailed information on how to use DirectC. The *DirectC User's Guide* is divided into the following chapters:

Chapter 1 - "Source Description" lists a description of the DirectC source code files.

Chapter 2 - "Required Modifications" provides information about the required file names that you need to modify.

Chapter 3 - "ISP Demo Board Tutorial" describes an example of how to use the DirectC code.

Appendix A - "Error Messages & Troubleshooting Tips" contains workarounds for error messages and warnings.

Appendix B - "Product Support"describes our support services.

Document Assumptions

The information in this manual is based on the following assumptions:

1. You are familiar with ANSI C programming.
2. You are familiar with the STAPL file generated from Designer or Libero.
3. You are familiar with the compilers provided with the microprocessor of your choice.
4. You are familiar with your development platform for microprocessors.
5. You are familiar with Microsoft Program Maintenance Utility (NMAKE.EXE) for compiling sample projects.

Actel Application Notes

Application Notes are available at our web site,

http://www.actel.com/products/tools/prog.aspx

Listed below are several Application Notes available to help you program your device:

- In-System Programming ProASIC^{PLUS} Devices Application Note
- Performing Microprocessor Programming For Actel's ProASIC^{PLUS} Devices

Actel User Manuals

Other Actel user manuals are available at our web site, under Technical Documentation http://www.actel.com/techdocs/manuals/default.aspx.

1 Source Description

The DirectC source code consists of the following files: dpalg.h, dpapa.h, dpapaalg.h, dpapadef.h, dpapads.h, dpdef.h, dpjtag.h, dpuser.h, dpalg.c, dpapaalg.c, dpjtag.c, dpjtag2.c, dpuncomp.c, dpuser.c

For detailed information about dpuser.h and dpuser.c files, see "Required Modifications" on page 19.

dpapaalg.h

dpapaalg.h file includes action name definitions, bitstream data variables and the STAPL file read function definition.

Action Name Definitions

DirectC supports the following actions in Table 1-1 for the dpalg.h file:

Table 1-1. Action Names and Definitions

Action	Definition
`#define DP_STP_AN_PROGRAM`	Program
`#define DP_STP_AN_ERASE`	Erase
`#define DP_STP_AN_READ_IDCODE`	Read_idcode
`#define DP_STP_AN_VERIFY`	Verify
`#define DP_STP_AN_DEVICE_INFO`	Device_info
`#define DP_STP_AN_QUERY_SECURITY`	Query_security
`#define DP_STP_AN_ERASE_ALL`	erase_all

Bitstream Data Variable Declaration

The following bitstream data variables are used in the dpalg.c and dpuncomp.c files:

```
extern DPUCHAR *DESIGN
extern DPUCHAR *sw
extern DPUINT CHECKSUM
extern DPUCHAR *UKEY
extern DPULONG UKEYBITLEN
```

STAPL File Read Function Declaration

DirectC accepts a STAPL file as input, extracts the bitstream data from the input STAPL file, and ignores the programming algorithm.The following function is the STAPL file read function in DirectC:

```
DPINT dp_read_stapl(DPUCHAR *f_buffer, DPULONG f_length);
```

The f_buffer points to the address of the input STAPL file, and the f_length is the total byte sizes of the input STAPL file.

Note: If successful, this function returns with a DPE_SUCCESS message. After this function successfully returns, the bitstream data variables DESIGN, sw, CHECKSUM, UKEY, and UKEYBITLEN are assigned with the correct data (UKEY is 0 if there is no key in the input STAPL file).

dpalg.h

dpalg.h file includes CRC check and version comparing functions.

CRC Check Function Declaration

The CRC check function checks the CRC value of the input STAPL file. The following function is the CRC check function declaration for DirectC:

```
DPINT dp_check_crc_value (DPUCHAR *f_buffer, DPULONG f_length,
DPUSHORT* pActualCrc, DPUSHORT* pClaimedCrc);
```

The f_buffer points to the address of the input STAPL file, and the f_length is the total byte sizes of the input STAPL file. The pActualCrc holds the CRC value calculated by DirectC, and the pClaimedCrc holds the CRC value listed in STAPL file.

Note: This function returns DPE_SUCCESS if the calculated CRC value matches the listed CRC value.

Version Comparing Function Declaration

The version comparing function is used to compare the input STAPL file version with the DirectC source code version. This function verifies that the algorithm embedded in DirectC is a newer version than the STAPL file. If the STAPL file has a newer algorithm version, please visit the Actel website at http://www.actel.com to download the latest version of DirectC. The following function is the version comparing function declaration for DirectC:

```
DPINT dp_version_compare(DPUCHAR *f_buffer, DPULONG f_length,
DPINT* pVerNum);
```

The f_buffer points to the address of the input STAPL file, and the f_length is the total byte sizes of the input STAPL file. The pVerNum is used to hold the ALG_VERSION number listed in the STAPL file.

Note: This function returns DPE_SUCCESS if the two version numbers match. If the two version numbers do not match, a warning message appears.

dpdef.h

The dpdef.h file contains the programming algorithm version number, the main entry function return type, the main entry function, and the utility functions.

Programming Algorithm Version Number

The programming algorithm version in DirectC is:

```
#define DP_ALG_VERSION 17
```

Main Entry Function Return Type Declaration

The main entry function return type declarations are listed in Table 1-2.

Table 1-2. Return Type Declarations

Return Type	Value
#define DPE_SUCCESS	0
#define DPE_OUT_OF_MEMORY	1
#define DPE_IDCODE_ERROR	2
#define DPE_CRC_MISMATCH	3
#define DPE_SYNTAX_ERROR	4
#define DPE_NOWRITE_ERROR	5
#define DPE_ACTION_NOT_FOUND	6
#define DPE_POWER_DOWN_RETURN	7
#define DPE_FSETPOS_ERROR	8
#define DPE_VERSION_ERROR	9
#define DPE_FACTORY_CRC_ERROR	10

Table 1-2. Return Type Declarations (Continued)

Return Type	Value
`#define DPE_VERIFY_ERROR`	11
`DPE_NO_DATA_STREAM`	12
`DPE_NO_ACTION_IN_STAPL`	13

Table 1-3 shows a description of the return values.

Table 1-3. Return Value Descriptions

Return Value	Description
0	Success
1	Out of memory
2	Read IDCODE error
3	CRC mismatch
4	Syntax error
5	Cannot write
6	Action missing
7	Power down return
8	dp_fsetpos () error
9	Version mismatch
10	Factory CRC error
11	Verify error
12	Core programming data is not available in the STAPL file
13	Requested action is not in the STAPL file

Main Entry Function Declaration

The main entry function declaration is listed below:

```
DPINT dp_top
(
    DPCHAR* pAction,
    DPUCHAR* f_buffer,
    DPULONG f_length,
    DPINT* exit_code
);
```

The function return values (DPINT) are defined in the main entry function return type declaration in the dpdef.h file.

The action names (pAction) are defined in the action name definitions in the dpalg.h file.

The f_buffer points to the address of the input STAPL file, and the f_length is the total byte sizes of the input STAPL file.

Note: The exit_code is defined on page 37 of the JEDEC Standard No. 71.

Table 1-4 lists exit codes and their definitions.

Table 1-4. Exit Code Descriptions

Exit Code	Description
0	Success
1	Checking chain failure
2	Reading IDCODE failure
3	Reading USERCODE failure
4	Reading UESCODE failure
5	Entering ISP failure
6	Unrecognized device ID
7	Device version is not supported
8	Erase failure
9	Blank check failure
10	Programming failure

Table 1-4. Exit Code Descriptions (Continued)

Exit Code	Description
11	Verify failure
12	Read failure
13	Calculating checksum failure
14	Setting security bit failure
15	Querying security bit failure
16	Exiting ISP failure
17	Performing system test failure

Utility Function Declaration

DirectC uses the following utility function in the dpuncomp.c and dpjtag.c files:

```
DPUCHAR dp_convtohexvalue(DPUCHAR ch);
```

The following function is defined in the dpuncomp.c file and used in the dpalg.c file:

```
DPINT dp_stricmp(DPUCHAR *left, DPUCHAR *right);
```

dpjtag.h

The dpjtag.h file consists of the JTAG TAP controller state declaration, pre/post data variable declaration, and the JTAG related functions.

JTAG TAP Controller State Declaration

The JTAG TAP controller state definition follows the Merlin Programming Circuit Design Specification, GF-ICD-1032 Rev. 2.0.

typedef enum

```
{
    DREXIT2 = 0,
    DREXIT1 = 1,
    DRSHIFT = 2,
    DRPAUSE = 3,
    IRSELECT = 4,
```

```
    DRUPDATE = 5,
    DRCAPTURE = 6,
    DRSELECT = 7,
    IREXIT2 = 8,
    IREXIT1 = 9,
    IRSHIFT = 10,
    IRPAUSE = 11,
    IDLE = 12,
    IRUPDATE = 13,
    IRCAPTURE = 14,
    RESET = 15
} DPSTATE;
```

Pre/Post Data Variable Declaration

The pre/post data variable declaration variables are initialized in the dpjtag2.c file and are used in the dpjtag.c file. Their default values are 0s. You do not need to change these values if you are programming a standalone APA device. However, you must correctly set these variables if you are programming an APA device in a daisy chain. See "dpjtag2.c" on page 16 for information on how to set the variable values listed below.

```
extern DPUINT dp_preir_length;
extern DPUINT dp_predr_length;
extern DPUINT dp_postir_length;
extern DPUINT dp_postdr_length;
extern DPUCHAR* dp_initial_preir_data;
extern DPUCHAR* dp_initial_predr_data;
extern DPUCHAR* dp_initial_postir_data;
extern DPUCHAR* dp_initial_postdr_data;
```

JTAG Related Function Declaration

The JTAG related function declarations are implemented in the dpjtag.c file and are called from the dpalg.c file. You do not need to change the functions listed below:

```
void dp_jtag_reset(void);
void dp_goto_state(DPSTATE state);
void dp_wait_cycles(DPULONG cycles, DPSTATE state);
```

```
void dp_wait_microseconds(DPULONG ms, DPSTATE state);
void dp_do_irscan(DPUINT n, DPUCHAR* tdi_data, DPULONG start_bit_index);
void dp_do_irscan8(DPUCHAR opcode);
void dp_do_drscan(DPUINT n, DPUCHAR* tdi_data, DPULONG start_bit_index);
void dp_do_1bit_capture_drscan(DPUCHAR tdi, DPUCHAR* tdo);
void dp_do_capture_drscan(DPUINT n, DPUCHAR* tdi_data, DPULONG tdi_index,
DPUCHAR* tdo_data, DPUINT tdo_index);
void dp_init_jtag_variables(void);
void dp_clean_jtag_variables(void);
```

dpjtag.c

The dpjtag.c file describes how Actel implements JTAG related functions declared in the dpjtag.h file. Do not change this file.

dpjtag2.c

The following codes initialize the pre/post data variables:

```
dp_preir_length = PREIR_LENGTH_VALUE;
dp_predr_length = PREDR_LENGTH_VALUE;
dp_postir_length = POSTIR_LENGTH_VALUE;
dp_postdr_length = POSTDR_LENGTH_VALUE;
dp_initial_preir_data = PREIR_DATA_INIT_VALUE;
dp_initial_predr_data = PREDR_DATA_INIT_VALUE;
dp_initial_postir_data = POSTIR_DATA_INIT_VALUE;
dp_initial_postdr_data = POSTDR_DATA_INIT_VALUE;
```

Note: See Pre/Post Data Variable Declaration to set pre/post data.

The devices sitting in a chain between the need-programming APA device and the TDO of programming header are called pre-devices. The devices between the need-programming APA device and the TDI of the programming header are called post-devices. In Figure 1-1 on page 17, devices four and five are pre-devices, devices one and two are post-devices, and the APA 3 is the device that is programed.

Figure 1-1. Devices in the Chain

If there are N1 pre-devices and N2 post-devices in a chain, L1 is the sum of IR lengths of all the pre-devices. L2 is the sum of IR lengths of all the post devices. Table 1-5 is an example of how to set the values for the dpjtag2.c file using the following variables: N1 = 2, N2 = 3, L1 = 5, L2 = 8.

Table 1-5. Setting Values for dpjtag2

Pre/Post Data Values	Comments
#define PREIR_LENGTH_VALUE 5	L1
#define PREDR_LENGTH_VALUE 2	N1
#define POSTIR_LENGTH_VALUE 8	L2
#define POSTDR_LENGTH_VALUE 3	N2
#define PREIR_DATA_INIT_VALUE "1F"	L1's 1
#define PREDR_DATA_INIT_VALUE "3"	N1's 1
#define POSTIR_DATA_INIT_VALUE "FF"	L2's 1
#define POSTDR_DATA_INIT_VALUE "7"	N2's 1

Note: The values of PREIR_DATA_INIT_VALUE, PREDR_DATA_INIT_VALUE, POSTIR_DATA_INIT_VALUE POSTDR_DATA_INIT_VALUE are in quoted hex format without "0x" prefix. They can be zeros, and their non-zero values are the following values: “1”, “3”, “7”, “F”, “1F”, “3F”, “7F”, “FF”, “1FF”, “3FF”, “7FF”, “FFF”, …

dpalg.c

This file contains functions that checks for a STAPL file crc and for a device ID, then calls for the programming algorithm.

dpuncomp.c

The bitstream data reading routine, the CRC checking routine, and the version comparison routines are in the dpuncomp.c file.

dpapaalg.c

The programming algorithm for APA devices is located in this file.

2
Required Modifications

You must modify the dpuser.h and dpuser.c files when using the DirectC source code.

dpuser.h

The dpuser.h file contains data type definitions, compile switches, constant values, and function declaration descriptions.

Data Type Definition

The data types for the dpuser.h file used in the DirectC source codes are listed below:

```
typedef unsigned char   DPUCHAR;
typedef unsigned short  DPUSHORT;
typedef unsigned int    DPUINT;
typedef unsigned long   DPULONG;
typedef unsigned char   DPBOOL;
typedef char            DPCHAR;
typedef int             DPINT;
typedef long            DPLONG;
```

Actel uses DPUCHAR, DPUINT, DPULONG, DPBOOL, DPCHAR, DPINT, and DPLONG in the DirectC source codes. Change the middle columns if you use different data type names. The definitions in Table 2-1 must conform to the following rules.

Table 2-1. Data Type Definitions and Rules

Definition	Rule
DPCHAR	-- 8-bit signed integer
DPINT	-- 16-bit signed integer
DPLONG	-- 32-bit signed integer
DPBOOL	-- boolean variable (0 or 1)
DPUCHAR	-- 8-bit unsigned integer
DPUSHORT	-- 16-bit unsigned integer

Table 2-1. Data Type Definitions and Rules (Continued)

Definition	Rule
DPUINT	-- 16-bit unsigned integer
DPULONG	-- 32-bit unsigned integer

Constant Values

The constant values are listed below:

```
#define USE_RCK_VALUE 1
#define RCK_FREQUENCY_VALUE 4
```

If you are using an RCK clock, define USE_RCK_VALUE to 1. If you are using different RCK frequency, set RCK_FREQUENCY_VALUE to the value you are using. The RCK_FREQUENCY_VALUE must be an integer.

Note: The unit is MHz. The default value is 4 MHz.

The following pre/post-ir/dr related values are calculated the same as the values for dpjtag2.c.

```
#define PREIR_LENGTH_VALUE        0
#define PREDR_LENGTH_VALUE        0
#define POSTIR_LENGTH_VALUE       0
#define POSTDR_LENGTH_VALUE       0
#define PREIR_DATA_INIT_VALUE     "0"
#define PREDR_DATA_INIT_VALUE     "0"
#define POSTIR_DATA_INIT_VALUE    "0"
#define POSTDR_DATA_INIT_VALUE    "0"
```

Function Declaration Description

Replace the following functions with your own.

```
#define DPPRINTF                  EbioHostXmt
#define ACT_DP_MALLOC             dp_malloc
#define ACT_DP_FREE               dp_free
#define ACT_DP_JTAG_TMS           dp_jtag_tms
#define ACT_DP_JTAG_TMS_TDI       dp_jtag_tms_tdi
#define ACT_DP_JTAG_TMS_TDI_TDO   dp_jtag_tms_tdi_tdo
#define ACT_DP_DELAY              dp_delay
```

Actel uses the second columns in our C source codes. Change the last column in your source codes. Table 2-2 lists a description of the function declarations. For additional information, see "dpuser.h" on page 19.

Table 2-2. Function Declarations

Function	Definition
DPPRINTF	Behaves identically as print function in C language. It prints formatted output to the standard output stream.
ACT_DP_MALLOC	Allocates memory blocks of size bytes. It returns a void pointer to the allocated space, or NULL if there is insufficient memory available.
ACT_DP_FREE	Frees the allocated memory blocks
ACT_DP_JTAG_TMS	Gives a single TCK clock cycle with TMS input
ACT_DP_JTAG_TMS_TDI	Gives a single TCK clock cycle with TMS input and TDI input. It does not care about the TDO output value
ACT_DP_JTAG_TMS_TDI_TDO	Gives a single TCK clock cycle with TMS input and TDI input, and it returns 0x80 if TDO output is 1, 0 if TDO output is 0

Table 2-2. Function Declarations (Continued)

Function	Definition
ACT_DP_DELAY	Used to indicate the elapse of real time (microseconds)
DPINT dp_set_frequency(DPLONG hertz);	The DirectC programming algorithm is based on the STAPL player generated by Designer. This function corresponds to the FREQUENCY statement in the STAPL file. It defines the maximum frequency at which the IEEE 1149.1 TCK signal operates following execution of this function. DirectC usually uses RCK as the ramp clock source (USE_RCK_VALUE =1). Return 0 in dp_set_frequency() function.

dpuser.c

The dpuser.c file is a sample describing how to implement the functions in the dpuser.h file. See Table 2-2 to implement your own functions.

3

ISP Demo Board Tutorial

This chapter describes installation instructions, eZ80 code descriptions, and a programming device example.

Installation Instructions

Use the instructions below to install the sample project.

To install the sample project:

1. **Upzip the DirectCsampleproject.zip sample project file and save it to the "c:\directc" directory.**
2. **Open the Command Prompt window and change the current directory to "ebactel":**

   ```
   c:
   cd \directc\ebactel
   ```

3. **If you installed the eZ80 C compiler in the default directory "c:\program files\ZiLOG\eZ80CC_1.02" go to step four. Otherwise, change the Include and Bin directories in the file "makefile" to their installed directories as follows (suppose you actually installed the eZ80 C compiler in the directory "c:\myeZ80CC"):**

   ```
   I = -I "c:\program files\ZiLOG\eZ80CC_1.02\include"
   ==>   I = -I "c:\myeZ80CC\include"
   C = c:\program files\ZiLOG\eZ80CC_1.02\bin
   ==>   C = c:\myeZ80CC\bin
   ```

4. **From the Command Prompt window in step 2, type the following command:**

   ```
   nmake /f makefile
   ```

For a step-by-step project example, refer to the *ProASIC^{PLUS} ISP Demo Board User's Guide*.

eZ80 Code Description

This section describes the eZ80 code.

DPPRINTF

DPPRINTF behaves identically as the printf function in C language. The print formats outputs to the standard output stream.

```
void EbioHostXmt (char *fmt, . . .);
```

dp_malloc

The dp_malloc function allocates memory blocks of size bytes. This function returns a void pointer to the allocated space or NULL if there is insufficient memory available.

```
void *dp_mallco (DPUINT size);
```

dp_free

The dp_free function frees the allocated memory blocks.

```
void dp_free (void *ptr);
```

dp_jtag_tms

The dp_jtag_tms function performs a single TCK clock cycle with TMS input (tms).

```
void dp_jtag_tms (DPBOOL tms);
```

dp_jtag_tms_tdi

The dp_jtag_tms_tdi function performs a single TCK clock cycle with TMS input (tms) and TDI input (tdi), discarding the TDO output value.

```
void dp_jtag_tms_tdi (DPBOOL tms, DPBOOL tdi);
```

dp_jtag_tms_tdi_tdo

The dp_jtag_tms_tdi_tdo function performs a single TCK clock with TMS input (tms) and TDI input (tdi). This function returns 0x80 if TDO output is 1, return 0 if TDO output is 0.

```
DPUCHAR dp_jtag_tms_tdi_tdo (DPBOOL tms, DPBOOL tdi);
```

dp_delay

The dp_delay function is an elapse of real time (“microseconds”).

```
void dp_delay (DPULONG microseconds);
```

dp_set_frequency

The dp_set_fequency function simply returns 0 if USE_RCK is defined in dpalg.c.

```
DPINT dp_set_frequency (DPLONG hertz);
```

Programming an APA750 Device Example

The following tutorial is an example of programming a standalone APA750 device using a ProASIC^{PLUS} In-System Programming (ISP) demonstration board with a Zilog eZ80190 Microprocessor, 2 MB of Flash RAM, 2 MB of SRAM, and an on-board DC to DC convertersand. To reproduce this tutorial, you will also need a eZ80 ANSI C compiler with a 1.02 Release. For more information about the ProASIC^{PLUS} In-System Programming (ISP) demonstration, go to http://www.actel.com/products/tools/demoboards/PAplusISP.asp.

The basic steps below are followed by additional steps to guide through the tutorial.

To program a standalone APA750 device:

1. **Modify the dpuser.h file.** See "Modifying the dpuser.h File" section for more information.
2. **Rewrite the dpuser.c file.** Implement all the functions defined in dpuser.h. Discard the dpuser.c file of the sample project and rewrite your own function implementations.
3. **Build the binary executable ebfw.bin.** Replace the dpuser.h and dpuser.c files in the sample project with your modified dpuser.h and dpuser.c files and issue the following commands in a Command Prompt window to generate the binary executable file ebfw.bin.

   ```
   c:
   cd \directc\ebactel
   nmake /f makefile
   ```

4. **Run the Windows host software.** See "Running the Windows Host Software" on page 27 for more information.

Modifying the dpuser.h File

Follow the steps below to modify the dpuser.h file.

To modify the dpuser.h file:

1. **Modify data type definitions if necessary.** If a microprocessor supports ANSI C programming language, then it will most likely support the following data type definitions:

   ```
   typedef unsigned char  DPUCHAR;
   typedef unsigned short DPUSHORT;
   typedef unsigned int   DPUINT;
   typedef unsigned long  DPULONG;
   typedef unsigned char  DPBOOL;
   typedef char           DPCHAR;
   typedef int            DPINT;
   typedef long           DPLONG;
   ```

If your microprocessor supports ANSI C, then you do not need to change the definitions above. If not, then you need to change the middle columns.

2. **Define the RCK clock source and its frequency value.** The ProASIC^{PLUS} device contains an internal programming controller that controls the programming of the Flash switches. Voltages used during programming are based on an internal voltage reference, but the controller needs a 1 MHz clock derived from either RCK or TCK for its time reference. The part includes an internal five bit programmable clock divider, so the external clock can be any multiple of 1 MHz, between 1 MHz and
31 MHz. The external time reference can either come from TCK or RCK. The RCK input is provided as an easy to use reference clock input. Connect an appropriate oscillator to the RCK pin and the programming controller's clock reference needs are met.

DirectC prefers to use the RCK input during programming. Do not change the following definition in the dpuser.h file.

```
#define USE_RCK_VALUE 1
```

If TCK is set at a constant frequency that is an integral multiple of 1 MHz and never stopped during programming then it can be used as the reference clock instead of RCK. If you are going to use TCK as the reference clock, then set USE_RCK_VALUE to 0.

The default defined RCK reference clock's frequency is 4 MHz. So DirectC defines the following values in dpuser.h file. The unit is MHz.

```
#define RCK_FREQUENCY_VALUE 4
```

If you are using a different RCK frequency, which must be between 1 MHz and 31 MHz, then change this value.

3. **Define daisy chain related variables.** Do not change the following definitions because you are programming a standalone APA device.

```
#define PREIR_LENGTH_VALUE        0
#define PREDR_LENGTH_VALUE        0
#define POSTIR_LENGTH_VALUE       0
#define POSTDR_LENGTH_VALUE       0
#define PREIR_DATA_INIT_VALUE     "0"
#define PREDR_DATA_INIT_VALUE     "0"
#define POSTIR_DATA_INIT_VALUE    "0"
#define POSTDR_DATA_INIT_VALUE    "0"
```

If you are programming an APA device in a daisy chain, then set these values according to the information in the section “dpjtag2.c” on page 16.

4. **Define the following functions and implement them in the dpuser.c file.** For detailed information about these functions, refer to "dpuser.h" on page 19.

```
#define DPPRINTF EbioHostXmt
#define ACT_DP_MALLOC dp_malloc
#define ACT_DP_FREE dp_free
#define ACT_DP_JTAG_TMS dp_jtag_tms
#define ACT_DP_JTAG_TMS_TDI dp_jtag_tms_tdi
#define ACT_DP_JTAG_TMS_TDI_TDO dp_jtag_tms_tdi_tdo
#define ACT_DP_DELAY dp_delay
void EbioHostXmt(char *fmt, ...);
void *dp_malloc(DPUINT size);
void dp_free(void *ptr);
void dp_jtag_tms(DPBOOL tms);
void dp_jtag_tms_tdi(DPBOOL tms, DPBOOL tdi);
DPUCHAR dp_jtag_tms_tdi_tdo(DPBOOL tms, DPBOOL tdi);
void dp_delay(DPULONG microseconds);
DPINT dp_set_frequency(DPLONG hertz);
```

For example, if the print function in your microprocessor is called printf, then make the changes below in the dpuser.h file and implement the printf function in the dpuser.c file:

```
#define DPPRINTF printf
void printf (char *fmt, ...);
```

Running the Windows Host Software

For more detailed information about the evaluation board, see the ebactel-usermanual.pdf in the sample project.

To run the Windows host software:

1. **Type the following commands from the Command Prompt window:**

```
c:
cd \directc\ebactel
cliebactel.exe ebstartup.tcl
```

The Actel ProASIC^{PLUS} Evaluation Board window displays. The default communication port in the sample project is COM2. If you use COM1, then type openport com1: in the display window.

2. **Download and reprogram the firmware image in the board.** Type the commands below in the Actel ProASIC^{PLUS} Evaluation Board window. Then, follow the on-screen instructions.

   ```
   updateprogramflash ebfw.bin
   ```

3. **Copy the STAPL file from your PC to the data flash on the board and select it.**

   ```
   download <filename.stp>
   ```

4. **Program the FPGA from the selected file.**

   ```
   programfpga
   ```

A

Error Messages & Troubleshooting Tips

The information in this chapter may help you solve or identify a problem when using DirectC code. If you have a problem that you cannot solve, visit the Actel website at http://www.actel.com/custsup/search.html or contact Actel Customer Technical Support at tech@actel.com or call our hotline 1-800-262-1060. This chapter contains information on the following error messages:

Exit 0

This message means success. This does not indicate an error.

Exit 6

JEDEC standard message. The IDCODE of the target device does not match the expected value in the STAPL file.

Possible Causes:

- You loaded an APA150 STAPL file to program an APA300.
- You selected wrong device.
- Device TRST pin is grounded.
- Noise or reflections on one or more of the JTAG pins caused by the IR Bits reading it back incorrectly.

Solutions:

- Choose the correct STAPL file and select the correct device.
- Measure JTAG pins and noise or reflection. TRST should be floating or tied high.
- Cut down the extra length of ground connection.

Exit 7

Unknown algorithm: alg=x, prev=x
Invalid data read from device

Possible Causes:

- In the factory row, the factory writes the algorithm revision the part is calibrated with. This error occurs with current STAPL files when the revision written into the factory row is not rev 2 for ProASIC devices. The STAPL files from last year may "exit 7" with newer devices or the older revision may cause this failure if the STAPL file used is from latest version. This can occur if you are using Engineering Sample parts that are no longer supported, such as ProASIC Engineering Sample parts.
- Programming -F ProASIC^{PLUS} device with old STAPL file.
- Connect V_{PP} and V_{PN} the wrong way around.
- No bypass Caps on V_{PP} V_{PN}, which damaged the device.

Solutions:

- Re-generate STAPL file from Libero v2.3 SP3 or Designer R1-2003SP3.
- Double check V_{PP} and V_{PN} connections.
- Make sure V_{PP} and V_{PN} have correct bypass caps.

Exit 8

This message occurs when the FPGA failed during the Erase operation.

Possible Causes: The device is secured, and the corresponding STAPL file is not loaded. The device has been permanently secured and cannot be unlocked.

Solution: Load the correct STAPL file.

Exit 11

The message occurs when the FPGA failed verify.

Possible Causes

- The device is secured, and the corresponding STAPL file is not loaded.
- You used the Libero software v. 2.3 or earlier or the Designer R1-2003 software or earlier to generate the STAPL file.
- V_{PN} caps were soldered in the wrong polarity.

Solutions:

- Load the correct STAPL file.
- Use later software versions-- at least Libero v2.3 SP1 and Designer R1-2003 SP1.
- Double check the V_{PN} bypass caps polarity.

Exit 12

Occurs when security is enabled.

Possible Causes:

- The device is secured and the wrong key/STAPL file was entered.
- The device is damaged.
- The verification was interrupted and therefore fails, causing it to think the device is secure.

Exit 15

This message is a factory Calibration Data CRC Error. During program, erase, or verify, the programmer must read back Calibration Data from the FPGA. The Data contains a CRC and the programmer uses the CRC to ensure the data is not corrupted/wrong.

Possible Causes:

- The device is damaged.
- Noise on the JTAG signals causes the programmer to read back wrong data.

Exit 17

This message means that the device has been secured and write-security enabled.

Possible Causes:

- The device is secured and the wrong key/STAPL file was entered.
- The device is damaged.

Solution: Please load the correct STAPL file.

B

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0)1276.401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650. 318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/contact/offices/index.html.

Worldwide Sales Offices

Headquarters

Actel Corporation
2051 Stierlin Ct.
Mountain View, California
94043
Toll Free: 888.99.ACTEL

Tel: 650.318.4200
Fax: 650.318.4600

US Sales Offices

California

Bay Area
Tel: 650.318.4200
Fax: 650.318.4600

Irvine
Tel: 949.788.0980
Fax: 949.788.0822

Newbury Park
Tel: 805.375.5769
Fax: 805.375.5749

Colorado

Tel: 303.420.4335
Fax: 303.420.4336

Florida

Tel: 407.977.6846
Fax: 407.977.6847

Georgia

Tel: 770.623.3960
Fax: 770.831.0055

Illinois

Tel: 847.259.1501
Fax: 847.259.1572

Massachusetts

Tel: 978.244.3800
Fax: 978.244.3820

Minnesota

Tel: 651.917.9116
Fax: 651.917.9114

North Carolina

Tel: 919.654.4529
Fax: 919.674.0055

Pennsylvania

Tel: 215.794.2050
Fax: 215.706.0680

Texas

Tel: 972.312.8700
Fax: 972.312.8707

International Sales Offices

Canada
Suite 106
235 Stafford Rd. West,
Nepean, Ontario K2H 9C1

Tel: 613.726.7575
Fax: 613.726.8666

France
361 Avenue General de Gaulle
92147 Clamart Cedex

Tel: +33 (0)1.40.83.11.00
Fax: +33 (0)1.40.94.11.04

Germany
Actel GmbH

Lohweg 27,
D-85375 Neufahrn, Germany
Phone: +49.(0)81.659.584.0
Fax: +49.(0)81.659.584.10

Hong Kong
Suite 2114,
Two Pacific Place,
88 Queensway

Admiralty, Hong Kong
Tel: +852 2185 6460
Fax: +852 2185 6488

Italy
Via Giovanni da Udine No. 34
20156 Milano

Tel: +39 (0)2.3809.3259
Fax: +39 (0)2.3809.3260

Japan
EXOS Ebisu Building 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150

Tel: +81 (0)3.3445.7671
Fax: +81 (0)3.3445.7668

Korea
30th Floor, ASEM Tower,
159-1 Samsung-dong
Kangnam-ku, Seoul

Tel: +82.2.6001.3382
Fa x: +82.2.6001.3030

Nordic
Sveavagen 49
Djursholm 18262
Sweden

Tel: +46.8544.99025
Fax: +46.8544.99026

Taiwan
4F-3, No. 75, Sec. 1,
Hsin-Tai-Wu Road,
Hsi-chih, Taipei, 221

Tel: +886.928.273190
Fax: +886.226.626251

United Kingdom
Dunlop House,
Riverside Way
Camberley,
Surrey GU15 3YL

Tel: +44 (0)1276.401450
Fax: +44 (0)1276.401490

Index

For more information about Actel's products, visit our website at http://www.actel.com

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 USA*
Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

Actel Europe Ltd*. • Dunlop House, Riverside Way • Camberley, Surrey GU15 3YL • United Kingdom*
Phone +44 (0) 1276 401 450 • Fax +44 (0) 1276 401 490

Actel Japan *• EXOS Ebisu Bldg. 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • www.jp.actel.com

Actel Hong Kong *• Suite 2114, Two Pacific Place • 88 Queensway, Admiralty Hong Kong*
Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn

50200008-2/09.06