

**SQP Command Line Interface**

**Development Guide**

**Neeraj Mallampet**

**Table of Contents**

1. **Preface**
   1. Purpose
   2. Location
      1. Contents
   3. Requirements
2. **Codebase**
   1. How is the code organized?
   2. Main Class
      1. Sqp.java
   3. Input File Handler
      1. inputFileParser.java
   4. Info Parser
      1. LDFParser.java
      2. XCFParser.java
      3. edaParser.java
   5. Data Structures
      1. propStruct.java
      2. deviceInfo.java
   6. Setup
      1. createHierarchy.java
      2. trunkSetup.java
   7. Running Flow
      1. genBit.java
      2. Programmer.java
      3. Ftdi.java
   8. Utility
      1. Utils.java
3. **Compiling CodeBase**
4. **Example Updates**
   1. Handle SQP GUI save file as input file
   2. Support New Devices
5. **Preface**
   1. **Purpose** The purpose of this document is to provide the reader with details on how SQP CLI was developed. With this, the reader can quickly pinpoint where in the codebase changes need to be made in order to make the desired update or modifications in the CLI.
   2. **Location** The project is located in the following directory:  
      [\\lscfs02\LDC-svsw27\Document-Lib\fphillip\SQP\SQP\_OP\SQPCLI\_codebase](file:///\\lscfs02\LDC-svsw27\Document-Lib\fphillip\SQP\SQP_OP\SQPCLI_codebase)  
      FileName: sqpCLI.zip  
      1. **Contents**Unzipping the folder will yield the following directories/files:  
         - **sqp:-** root/parent directory  
          - **nbproject:-** contains build properties to make jar file, (makefile)  
          - **res:-** contains external resources such as trunk script, ftdi.exe programs  
          - **src/sqp:-** the codebase  
          - **test:-** test directory for NetBeans (the IDE)  
          - **build.xml:-** jar build conf file  
          **- manifest.mf:-** manifest file
   3. **Requirements  
      - Java 1.6+ (Recommended: current version of Java (1.8)  
      - NetBeans8.1 (**[**download**](https://netbeans.org/downloads/8.1/)**)  
       -** Alternative: Eclipse

NetBeans is highly recommended since it’s a great editor for Java and makes compiling all the java source files easier (instead of manually making your makefile).

1. **Codebase** 
   1. **How is the code organized?** SQP CLI is an automation tool; it needs to properly get information from an input file and create the right environment to run the SQP flow and create results. In order to achieve that, the code’s logic can be described by the following image:   
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
        
      To summarize, the main class, sqp.java, will call inputFileParser.java to handle the input file. First, the inputFileParser class extracts information from the input file. Second, the extracted information is stored in data structures. Third, the stored information is used to set up the env needed to run the SQP Flow.   
        
      The following sections describe each block more in depth:

**The utility class contains functions that are commonly used. This way instead of redefining the same function every class, they can be accessed at this central location**

**Utility**

Utils.java

**Results are sent back to create results.txt**

**After env is setup, calls to classes responsible for running SQP flow are made to run SQP flow**

**The stored information is used to create the correct environment to run SQP flow**

**Calls input file handler to handle input file**

**Stores the parsed information in the appropriate data structure**

**Calls appropriate information parsing class to parse the input file**

**Running Flow**genBit.java

Programmer.java

Ftdi.java

**Setup**

createHierarchy.java

trunkSetup.java

**Data Structures**

propStruct.java

deviceInfo.java

**Info Parser**LDFParser.java

XCFParser.java

edaParser.java

**Input Fil Handler**   
inputFileParser.java

* 1. **Main Class**
     1. **Sqp.java**   
        The purpose of this class is to determine which input file the user passed in. The idea is to give user the freedom of using the save file (sqp file) generated from the GUI or their own input file.   
          
        **For now, the current SQP CLI only supports user created input file and not the save file generated by the GUI.**  
          
        In the future, when both formats are supported, this class will determine which input file the user passed in and send it to the appropriate class for processing.
  2. **Input File Handler**

**Input Fil Handler**   
inputFileParser.java

* + 1. **inputFileParser.java**The purpose of this class is to parse the inputfile, call the appropriate classes to store the information, call the appropriate classes to create the correct env to run the SQP flow from the stored information, call the appropriate classes to run the SQP flow, and create the results file.   
         
       This class can be thought of as a manager for the suite because it uses the information given to set up the correct env to execute the SQP flow.  
         
       The constructor for this class does the following:  
       1) Stores the output location (Passed in as arg; where all the results are stored)  
       2) Parses inputfile (calls to other classes)  
       3) Creates env (calls to other classes)  
       4) Run Flows (calls to other classes)  
       5) Creates results file
  1. **Info Parser**

**Info Parser**LDFParser.java

XCFParser.java

edaParser.java

* + 1. **LDFParser.java**  
       One required parameter from the input file is the ldf file’s location. The purpose of this class is to parse the ldf file to get names to identify the implementation directory, bitfile name, and get the device name info.  
         
       The device name is needed to retrieve the correct device info for Diamond from [diamond\_location]/data/vmdata/database/xpga/….
    2. **XCFParser.java**The purpose of this class is to get and set information in the XCF File.  
         
       **Setting Information:**This class allows user to set the Port address, device information, and bitfile locations  
         
       **Getting Information:**This class also allows users to obtain previously used port address (if an xcf file is provided initially). This is used to run programmer on the right port.
    3. **edaParser.java**The purpose of this class is to get the correct environment variable from the provided eda SW name in the inputfile.   
         
       Example:   
       input file contains line “eda=diamond”, this class would parse that and return the corresponding env variable which is “EXTERNAL\_DIAMOND\_PATH”.   
         
       This is used to correctly run the trunk script.
  1. **Data Structures**

**Data Structures**

propStruct.java

deviceInfo.java

* + 1. **propStruct.java**The purpose of this class is to store the information obtained from the input file, and allowing easy access to this information. Each instance of this class can be identified by its suite and testcase name.   
         
       This information is used to set up the SQP env to properly run the SQP flow.
    2. **deviceInfo.java**The purpose of this class is to get and store the device information to correctly create the XCF file which is used to run programmer correctly.   
         
       This class first uses the ldf file to get the device name, and, if the eda specified is diamond, uses the device name to get the vendor, family, idCode and pon information. This information is obtained, stored, and later used to set the XCF file.
  1. **Setup**

**Setup**

createHierarchy.java

trunkSetup.java

* + 1. **createHierarchy.java**The purpose of this class is to create the proper directory structure (hierarchy) in the output file.   
         
       For example, if the input file has a suite name “daisyChainSuite” with 2 testcases “case1” and “case2”. The output directory will have the suite directory “daisyChainSuite” and that directory will have its testcase directories.
    2. **trunkSetup.java**The purpose of this class is to properly setup each testcase with the right files/directories: bqs.info, bqs.conf, logs/, sim/, programmer/test.xcf, programmer/post\_process.py to run the SQP flow.   
         
       This class also creates the appropriate xcf file depending on whether or not the suite is a daisy case.
  1. **Running Flow**

**Running Flow**genBit.java

Programmer.java

Ftdi.java

* + 1. **genBit.java**The purpose of this class is to call on the trunk script and generate bitgen from the provided design location. This is ran as a separate thread  
         
       The trunck script’s output is stored in the log file “trunkLog.txt” and can be found in the log directory.
    2. **Programmer.java**The purpose of this class is to run programmer, from the provided eda location, using the genereated xcf file. If the programmer fails, this class will rerun programmer on a different port.   
         
       The programmer’s output is stored in the log file “ProgrammerOutput.txt” and can be found in the log directory.
    3. **Ftdi.java**The purpose of this program is to send the stimulus to the connected board by calling on a program written in C called “ftdi.exe”. This program also generates an output file which is stored in the sim/output\_files/ directory.   
         
       The ftdi’s output is stored in the log file “ftdiLog.txt” and can be found in the log directory
  1. **Utility**

**Utility**

Utils.java

* + 1. **Utils.java**This class contains functions that are used by more than one class. Instead of redefining the function for each class, they have been defined here.

1. **Compiling Codebase**The codebase was compiled using NetBeans, please make sure you have NetBeans installed before proceeding.

To open the project in NetBeans and compile, follow the following steps:

1. Open Netbeans
2. Go to File 🡪 Import Project 🡪 From ZIP…
3. Here you will be prompted to choose the ZIP file and the destination. After filling both fields, click on Import.
4. Now we have to add the library file:
   1. On the left panel, the project sqp, with a coffee icon next to it, should appear. Right click on it and go to properties
   2. In the properties panal, click on Libraries on the left pane.
   3. In the right panal, click on Add JAR/Folder and add all the contents from the jarlib-CLI folder. (This folder is located in the location of the codebase)
5. To compile:
   1. Click on the run button (the green right arrow on top) or F6
6. **Example Updates**
   1. **Handle SQP GUI save file as input file**For example, if the user wants to update the current CLI to also handle the SQP GUI save file (sqp file), they would have to make the following modifications:  
        
      1) Update the main class, sqp.java, to differentiate between the two input file. Calls the correct handler for each input file.   
        
      2) Create new input file parser class to parse the SQP GUI save file.   
      2.1)Store the information in the provided data structures using the provided file parsers  
        
      3) Use the information to make calls to setup and run SQP flow. (Everything from here should be the same)
   2. **Support New devices**To support new devices, go to the utils.java and edit the function   
        
      static public String XPGAFamilyLUT(String deviceName) {…}  
        
      In this function, the deviceName is analyzed to find out which family folder to go in the {diamond\_install\_path}/data/vmdata/database/xpga directory. In this directory, we need to go to the correct family directory to get the proper device information so we can update the xcf (for programmer) to run correctly. The xdf file is used in the family directory to get the proper device information.   
        
      For example, this function categorizes any deviceName that follows the regex pattern “L[A-Z]EC” as an ECP device.   
        
      To support a new device, do the same as mentioned above. Examples can be found in the function itself.