# 编译方法

将源码包拷贝到 …\ShellPkg\Application\ 目录下，解压，执行：

>edksetup.bat

>cd /ShellPkg/Application/ShellTestKit

>compile-debug.bat ShellPkg.dsc

*Note:若使用不同路径或者目录名，需要修改: ShellPkg.dsc中的ShellPkg/Application/ShellTestKit/ShellTestKit.inf*

# 可用库

公用库函数内容可能会随时间而逐渐增加，以下内容日期： 20180426

Compile-debug.bat 47 2017/8/27

Compile-release.bat 47 2017/8/27

Configure.h 880 2018/4/26

ShellPkg.dec 6928 2017/7/12

ShellPkg.dsc 5964 2017/8/27

ShellTestKit.c 4467 2018/4/26

ShellTestKit.inf 1448 2018/4/25

Top.h 1455 2018/4/25

ZXEyeMonitor.c 16205 2018/4/26

ZXEyeMonitor.h 1123268 2017/6/13

ZXLibFile.c 19599 2018/4/23 //文件访问操作 – Read part

ZXLibFile.h 1035 2017/9/4

ZXLibFile2.c 51 2017/9/4

ZXLibFile2.h 72 2017/9/4

ZXLibRW.c 8957 2018/4/26 //完全移植ZX BIOS 中ASIA的interface

ZXLibRW.h 2286 2018/4/26

ZXLibString.c 12158 2017/7/2 //字符串特殊处理库函数

ZXLibString.h 1156 2017/7/2

ZXPciExpress.c 5030 2018/4/26 //PCI/PCIE 方便使用的库函数

ZXPciExpress.h 1298 2018/4/26

ZXPeMcu.c 628 2017/6/13

ZXPeMcu.h 941 2018/4/26

上述红色字体部分是当前存在的一些公用库文件.以下为详述：

## ZXLibFile.c

/\*\*

Read the content of specific line

@Param:

FileName:target filename

Line:line number

Buff:the buffer to place the string we get in specific line

\*\*/

UINT8

ReadLine(IN CHAR16 \*FileName, IN UINT8 Line, IN OUT CHAR16\* Buff)

行为简述：调用者提供buffer，函数负责将文件中具体某一行的内容读取到buffer

/\*\*

@Param:

FileName: the filename of target file

@Return:

count of line in this file(From 0 to N-1)

@Note:

this routine will search the first No-String line,

and flag it as the end line of this file, so if there is a empty line(just a \n)

following with a string line, then this string line will not be counted

@Exp: the following will be calculated as 1, and this routing will return 1

xxx --- 0

yyy --- 1

zzz

\*\*/

UINT8

GetTotalLine(CHAR16 \*FileName)

行为简述：获取指定文件的总行数

/\*\*

this is a parser to get the text in the specfic Colume from the input string

@param:

input: input string, to be parsed

output: the buffer to put the text you wanted

Column: specific column number (from 1 to N)

@Exp: in following text, Comlumn 3 is 'yyy'

'xxx = yyy'

\*\*/

VOID

GetColumn(IN CHAR16\* input, IN OUT CHAR16\* output, IN UINT8 Column)

行为简述：次函数是一个文本处理函数，负责将调用者输入的text内容中指定的“列”提取出来；

例如输入文本为 “a\_bcd\_\_dfexc\_s\_\_\_\_xs\_\_sad”，其中“\_”代表空格， 则第三列的内容为“dfexc”

/\*\*

when we need a command param, calling this routine to get

the line number from the corresponding config file

@param:

FileName: the file name of the config file

Target: the CMD string we need to find

@return:

The lane number corresponding to the CMD string

\*\*/

UINTN

GetParamLineNumber(CHAR16\* FileName, CHAR16 \*Target)

行为简述：调用者提供文件名 & 命令关键字； 函数负责返回这个命令关键字在此文件中所处的具体行数

/\*\*

this routine provide the 'Data' string corresponding to the Cmd

@param:

FileName:the file name of the target file

Cmd: the cmd string we want to search

Data: the Data buffer pointer, the data corresponding to the Cmd will be put in

\*\*/

UINTN

GetParamData(CHAR16 \*FileName, IN CHAR16 \*Cmd, IN OUT CHAR16\* Data)

行为简述：此函数直接从指定文件中，读取命令关键字对应的数据

如文件中内容如下：

…

…

CMDx = 12345

…

…

其中CMDx为命令关键字， 12345则是对应的数据， 其格式必须是 “X = Y”这种格式；

次函数会直接解析出12345这个字符串返回给调用者预先准备好的buffer中。

/\*\*

this routine provide the specific column of 'Data' string corresponding to the Cmd

@param:

FileName:the file name of the target file

Cmd: the cmd string we want to search

Data: the Data buffer pointer, the data corresponding to the Cmd will be put in

\*\*/

UINTN

GetParamDataX(CHAR16 \*FileName, IN CHAR16 \*Cmd, UINT8 Column,IN OUT CHAR16\* Data)

行为简述：此函数相对于GetParamData更为灵活，可以取“命令关键字”所在行的 “指定列的内容”，返回给调用者预先提供的buffer中。

/\*\*

Write 'StrWr' into 'FileName', start from offset 0 in this file

@Param:

FileName: target file

StrWr: the content to be written into the file

\*\*/

EFI\_STATUS

WriteII(CHAR16 \*FileName, CHAR16 \*StrWr)

/\*\*

Write 'StrWr' into 'FileName', the insert position is 'pos'

@Param:

FileName: target file

StrWr: the content to be written into the file

pos: the start writing offset in file

\*\*/

EFI\_STATUS

WriteIIPos(CHAR16 \*FileName, CHAR16 \*StrWr, UINT64 pos)

/\*

EDK 2:

#include <Library/ShellLib.h>

Then you can call the shell libraries for file manipulations such as reading and writing a file.

Shell File manipulation functions:

CreateFile, DeleteFile, ReadFile, WriteFile, DeleteFileByName, CloseFile, FindFiles, FindFilesInDir, GetFilePosition, SetFilePosition, GetFileInfo, SetFileInfo, FreeFileList, OpenFileByName, OpenFileList, OpenRoot, OpenRootByHandle, GetFileSize, RemoveDupInFileList

\*/

VOID

FileTest(CHAR16 \* param)

行为简述：测试函数，仅供测试

## ZXLibRW.c

此文件提供了Asia中CFG/IO/Mem的8/16/32 bit操作函数。

///////////////////////////////////////////////////////////////////////////////////CFG

VOID AsiaPcieModify8(UINT64 Bar, UINT64 Address, UINT8 Mask, UINT8 Value);

VOID AsiaPcieModify16(UINT64 Bar, UINT64 Address, UINT16 Mask, UINT16 Value);

VOID AsiaPcieModify32(UINT64 Bar, UINT64 Address, UINT32 Mask, UINT32 Value);

UINT8 AsiaPcieRead8(UINT64 Bar, UINT64 Address);

UINT16 AsiaPcieRead16(UINT64 Bar, UINT64 Address);

UINT32 AsiaPcieRead32(UINT64 Bar, UINT64 Address);

VOID AsiaPcieWrite8(UINT64 Bar, UINT64 Address, UINT8 Data);

VOID AsiaPcieWrite16(UINT64 Bar, UINT64 Address, UINT16 Data);

VOID AsiaPcieWrite32(UINT64 Bar, UINT64 Address, UINT32 Data);

///////////////////////////////////////////////////////////////////////////////////IO

VOID AsiaIoModify8(UINT16 Address, UINT8 Mask, UINT8 Value);

VOID AsiaIoModify16(UINT16 Address, UINT16 Mask, UINT16 Value);

VOID AsiaIoModify32(UINT16 Address, UINT32 Mask, UINT32 Value);

UINT8 AsiaIoRead8(UINT16 Address);

UINT16 AsiaIoRead16(UINT16 Address);

UINT32 AsiaIoRead32(UINT16 Address);

VOID AsiaIoWrite8(UINT16 Address, UINT8 Data);

VOID AsiaIoWrite16(UINT16 Address, UINT16 Data);

VOID AsiaIoWrite32(UINT16 Address, UINT32 Data);

///////////////////////////////////////////////////////////////////////////////////Memory

VOID AsiaMemoryWrite8(UINT64 Address, UINT8 Data);

VOID AsiaMemoryWrite16(UINT64 Address, UINT16 Data);

VOID AsiaMemoryWrite32(UINT64 Address, UINT32 Data);

UINT8 AsiaMemoryRead8(UINT64 Address);

UINT16 AsiaMemoryRead16(UINT64 Address);

UINT32 AsiaMemoryRead32(UINT64 Address);

VOID AsiaMemoryModify8(UINT64 Address, UINT8 Mask, UINT8 Value);

VOID AsiaMemoryModify16(UINT64 Address, UINT16 Mask, UINT16 Value);

VOID AsiaMemoryModify32(UINT64 Address, UINT32 Mask, UINT32 Value);

///////////////////////////////////////////////////////////////////////////////////

## ZXLibString.c

/\*\*

This function will translate CHAR16 formate string to CHAR8 format string

\*\*/

char Char16Char8(IN CHAR16 \*in, OUT char \*out)

行为简述：CHAR16格式的字符串转CHAR8格式的字符串

/\*\*

CHAR8 Function

Hex Format string to Dec data

\*\*/

//'len' must be input by strlen(str);

unsigned int my\_strhex2dec(unsigned char \*str, unsigned char len)

行为简述：字符串格式的数字， 从16进制换位10进制

/\*\*

CHAR8 Function

Dec Format string to Dec data

\*\*/

//'len' must be input by strlen(str);

unsigned int my\_strdec2dec(unsigned char \*str, unsigned char len)

行为简述：十进制的字符串数字，转化为纯数字

/\*\*

CHAR8 Function

@param:

str: the string we get (CHAR8)

@return:

the number in this string

@note:

the formate of this string can be

1. Rx...(hex)

2. 0x...(hex)

3. ...(Dec)

\*\*/

//'len' must be input by strlen(str);

unsigned int my\_str2dec(unsigned char \*str, unsigned char len)

行为简述：将字符串格式表示的值，转化为串数字； 如Rx100转化为100h, 0x200转化为200h；

这有助于方便解析配置文件中不同表示方法.

/\*\*

CHAR16 function

Hex format string to Dec data

\*\*/

//'len' must be input by strlen(str);

UINT32 my\_strhex2dec\_16(CHAR16 \*str)

行为简述：十六进制的CHAR16字符串数据直接转换成十进制的值

/\*\*

CHAR16 Function

Dec format string to Dec data

\*\*/

UINT32 my\_strdec2dec\_16(CHAR16 \*str)

行为简述：十进制的CHAR16字符串数据直接转换成十进制的值

/\*\*

String to Number

@param:

str: the string we get (CHAR16)

@return:

the number in this string

@note:

the formate of this string can be

1. Rx...(hex)

2. 0x...(hex)

3. ...(Dec)

\*\*/

UINT32

Str2Num(CHAR16 \*str)

行为简述：此函数能够将各种格式表示的CHAR16字符串数值转化为纯数字。

相对my\_str2dec来讲，参数更少，更加易于使用。

/\*\*

this is a parser to get the text in the specfic Colume from the input string

the difference between this routine and Str2Num() is: this routine can auto

analyse the format like Rx70[3:0]/0x70[3:1] and output the Mask for those bit fields

@Param:

@Return:

EFI\_NOT\_FOUND: input format error

EFI\_SUCCESS;

Note:

Rx70[15:10]

local = Rx70

p1 = 4 is EndOfFirstPart

p2 = 7

p3 = 10 is EndOfSecondPart

\*\*/

EFI\_STATUS

Str2NumBitField (

IN CHAR16 \* input,

IN OUT UINT32 \*FieldMask,

IN OUT UINT8 \*BaseBit,

IN OUT UINT8 \*EndBit,

IN OUT UINT32 \*FirstPart)

行为简述：这个函数可以直接解析类似Rx70[4:1]这种表述方式

## ZXPciExpress.c

/\*\*

@Use this function to search register offset of Specific capability

@Param:

Busx,Devx,Funx: device routing ID

PcieCap: when 1, search PCIE capability, when 0,search PCI capability

CapId: target capability ID

@Return:

Offset: register offset of target capability

EFI\_STATUS: search result,

when EFI\_NOT\_FOUND, not found capability

when EFI\_DEVICE\_ERROR, device register abnormal

\*\*/

EFI\_STATUS

PCIeSearchCapOffset(

IN UINT8 Busx,

IN UINT8 Devx,

IN UINT8 Funx,

IN UINT8 PcieCap,

IN UINT16 CapId,

OUT UINT16\* pOffset)

行为简述：用于对指定function的配置空间上寻找特定PCI Capability or PCIE Extend Capability

VOID

PCIeDumpCfg(

UINT64 PcieBase,

UINT8 Busx,

UINT8 Devx,

UINT8 Funx,

UINT16 DCount)

行为简述：dump指定范围的配置空间数据

VOID

PCIeDumpMem(

UINT32 MemBase,

UINT16 DCount)

行为简述：dump指定范围的Memory空间数据

# 应用

Note: Tool中执行的代码若涉及到ZXLibFile.c中的函数，则需要先执行shell目录下的shell\_2\_0.efi. 这是因为shell\_2\_0.efi中提供了ZXLibFile.c中文件操作函数所依赖的必要protocol.

# 快速使用指南

## Help文件

直接输入ShellTestKit.efi 可以显示help信息。

## 使用Eye monitor Mode3测试眼图

BIOS: 任意， 不需要PEMCU FW

Example:

> ShellTestKit.efi –cmd half

>ShellTestKit.efi –mode3a 0xFEB14000 3 2 10 0 fast

–mode3a: Mode3 Eye scan half-rate

0xFEB14000: EPHY base address, CHX002 single socket EPHY base address is 0xFEB14000

3: Gen3 speed test (need to fill in current link speed)

2: EPHY lane2 to be tested(for CHX002, can be 0 to 15)

10: scan 10us for each point

0: threshold is 0, when the error count >0, then treat it as FAIL point

fast: fast dump mode

## 使用Eye monitor Mode2测试眼图

BIOS: 需要R04以后的BIOS， Setup中Enable PCIE FW load

Example:

Step1: Use HalfRate Mode

> ShellTestKit.efi –cmd half

Step2: Create cfg file, after create EyeCmd.cfg file, the default parameters no need to changed.

> ShellTestKit.efi –cmd createcfg

Step3: do test

> ShellTestKit.efi –cmd lane 4 1 0

lane: test in mode2

4: PE4

1: PE4 Lane1 (Note it’s not EPHY lane number, Just to use the lane number on the port)

0: threshold, when the error count >0, then treat it as FAIL point

Note：

Step2: 只需要第一次使用tool的时候执行一次

Step1: 只有每次system boot后需要执行一次

Step3: 每次测试时执行

## 快速测试Redo EQ

Tool中带有测试Eye monitor EQ的命令

如下内容是一个脚本在循环对CHX002所有PCIE port做redoEQ，循环100次

脚本： Scan.nsh

for %a run (1 100 1)

echo %a

ShellTestKit.efi -EMEQTEST 0 0 0 100000 >> EQ.log

endfor

分析dump的信息可以在log中搜索”EMEQ\_” 这个关键字符串。

分别得到的是：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 测试耗时 | | | | | | | | | |  | tune后具体信息 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Link Speed & Width After Tune | | | | | | | | COR Error | | | | | | | | | UNCOR ERROR |
|  |  |  | 4 | 0 | 10 | 0 | 9 | 0 | 0 | 11 |  | PE0 | | | | PE1 | | | | PE2 | | | | PE3 | | | | PE4 | | | | PE5 | | | | PE6 | | | | PE7 | | | |  |  |  |  |  |  |  |  |  | 0 |  | 0 | 0 | 0 |  | 0 | 0 |  | 00 0 0 0 0 |
|  |  |  | PE0 | PE1 | PE2 | PE3 | PE4 | PE5 | PE6 | PE7 |  | pre | cur | post | EW | pre | cur | post | EW | pre | cur | post | EW | pre | cur | post | EW | pre | cur | post | EW | pre | cur | post | EW | pre | cur | post | EW | pre | cur | post | EW | PE0 | PE1 | PE2 | PE3 | PE4 | PE5 | PE6 | PE7 |  | PE0 | PE1 | PE2 | PE3 | PE4 | PE5 | PE6 | PE7 |  | PE0PE1 PE2 PE3 PE4 PE5 PE6 PE7 |

# 测试脚本附录

## RxEQ测试

for %a run (1 100 1)

echo %a

ShellTestKit.efi -RXEQTEST 0 0 0 100000 >> RxEQ.log

ShellTestKit.efi -cmd lane 4 0 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 4 1 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 4 2 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 4 3 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 4 4 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 4 5 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 4 6 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 4 7 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 2 0 0 >> RxEQ.log

ShellTestKit.efi -cmd lane 7 0 0 >> RxEQ.log

endfor

上述两条质量搭配，-RXEQTEST可以输出PEMCU RxEQ的时候测试出来的Eye width(EW);