



HiDdrTraining

工具使用指南

文档版本 01

发布日期 2014-05-23

版权所有 © 深圳市海思半导体有限公司 2014。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址： 深圳市龙岗区坂田华为基地华为总部 邮编：518129

网址： <http://www.hisilicon.com>

客户服务邮箱： support@hisilicon.com



前 言

概述

本文档主要介绍 HiDdrTraining 的工具的界面以及使用方法。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3521	V100
Hi3531	V100
Hi3520D	V100
Hi3716M	V3XX
Hi3716C	V2XX

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 硬件开发工程师

作者信息

章节号	章节名称	作者信息
全文	全文	Y00250993/F00107764/z00194698



修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2014-01-16	00B01	第一次临时版本发布。
2014-05-23	01	增加对芯片 Hi3521 版本的支持。
2014-06-23	02	增加对 Reg.bin 文件的目录和描述。



目 录

前 言.....	iii
1 概述.....	1-1
1.1 工具介绍.....	1-1
1.2 主界面整体布局.....	1-1
2 界面及功能说明.....	2-4
2.1.1 菜单栏	2-4
2.1.2 工具栏	2-5
2.1.3 HiDdrTraining 配置视图.....	2-7
2.1.4 执行命令视图.....	2-10
2.1.5 控制台视图	2-12
3 HiDdrTraining 运行原理及脚本	3-1
3.1 HiDdrTraining 运行原理.....	3-1
3.1.1 HiDdrTraining 原理.....	3-1
3.1.2 HiDdrTraining 运行过程.....	3-5
3.2 脚本文件格式定义.....	3-9
3.2.1 节点之间关系.....	3-9
3.2.2 节点包含的属性介绍.....	3-10
3.2.3 脚本书写注意事项.....	3-11
4 FAQ.....	4-1
4.1 选择 Kernel Training 时，应注意什么？	4-1
4.2 下载 watchdog 失败，可能有什么原因呢？	4-1
4.3 错误“Address xx not found, the value will be xx”，是什么原因呢？	4-2
4.4 kernel training 的时候出现文件系统被破坏，重启也不能恢复？	4-2



插图目录

图 1-1 HiTool 解压包	1-2
图 1-2 HiDdrTraining 的按钮.....	1-2
图 1-3 HiDdrTraining 主界面.....	1-3
图 2-1 HiDdrTraining 透视图菜单栏.....	2-4
图 2-2 打开透视图对话框.....	2-5
图 2-3 HiDdrTraining 透视图工具栏.....	2-5
图 2-4 连接管理器对话框.....	2-6
图 2-5 活动连接	2-6
图 2-6 HiDdrTraining 视图.....	2-8
图 2-7 高级选项设置	2-9
图 2-8 Ko 命令配置.....	2-10
图 2-9 输入可执行命令	2-10
图 2-10 命令历史记录列表.....	2-11
图 2-11 弹出自动提示框.....	2-11
图 2-12 命令历史列表中的右键功能.....	2-11
图 3-1 DDR Training 前后时序对比.....	3-1



1 概述

1.1 工具介绍

当前 DDR 的频率越来越高，DDR 的时序窗口将越来越小，信号线上 skew 的变化更加敏感，且受到 PHY to IO 走线 skew，芯片封装 skew，单板走线的 skew 的影响。为了提高芯片 DDR 的抗干扰能力，获得更加稳定的 DDR 环境，提高 DDR 的速率，提升产品竞争力，DDR Training 软件的需求产生了。

为了提高芯片 DDR 的抗干扰能力，获得更加稳定的 DDR 环境，提高 DDR 的速率，有 2 种 Ddr Training 的方式：

- Boot 下 training
该方式单板上电时每次都会自动启动，动态配置 DDR 参数，使 DQS/DQ 的时序参数配置到中心位置。SDK 中提供的 ddr training 命令用于直观的查看 DQS/DQ 的参数；一般 Boot Training 在 boot 启动的过程中完成，Training 程序运行在 flash 或者片内的 SRAM 中，Training 的时间较短大约在 100ms 以内。
- 业务下 training
用于自动化做业务下 Training 和 boot Training 结果进行对比；有些芯片没有做 boot training，使用业务下 Training 结果作为 fastboot 表格的默认值，或者客户没有完全 copy 参考设计，表格默认值需要重新配置；HiDdrTraining 工具主要提供了相关辅助功能，下面再详细解释下工具的各个功能及页面。

1.2 主界面整体布局

在发布包的工具目录下，解压 HiTool XXX.zip（需要提前安装 jre-6u1-windows-i586-p-s.rar），解压后如点击 HiTool.exe，如图 1-1 所示。选择正确的芯片后，点击 HiDdrTraining 按钮，如图 1-2 所示。进入 HiDdrTraining 的主界面如下图 1-3，HiDdrTraining 主界面包含菜单栏，工具栏，透视图，HiDdrTraining 配置视图，控制台视图，TFTP 命令视图。



图1-1 HiTool 解压包

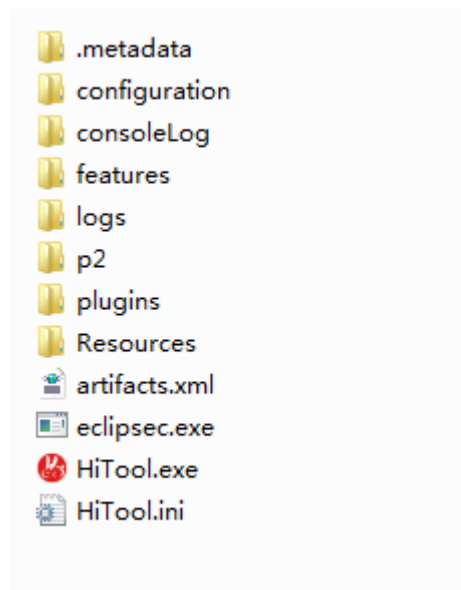
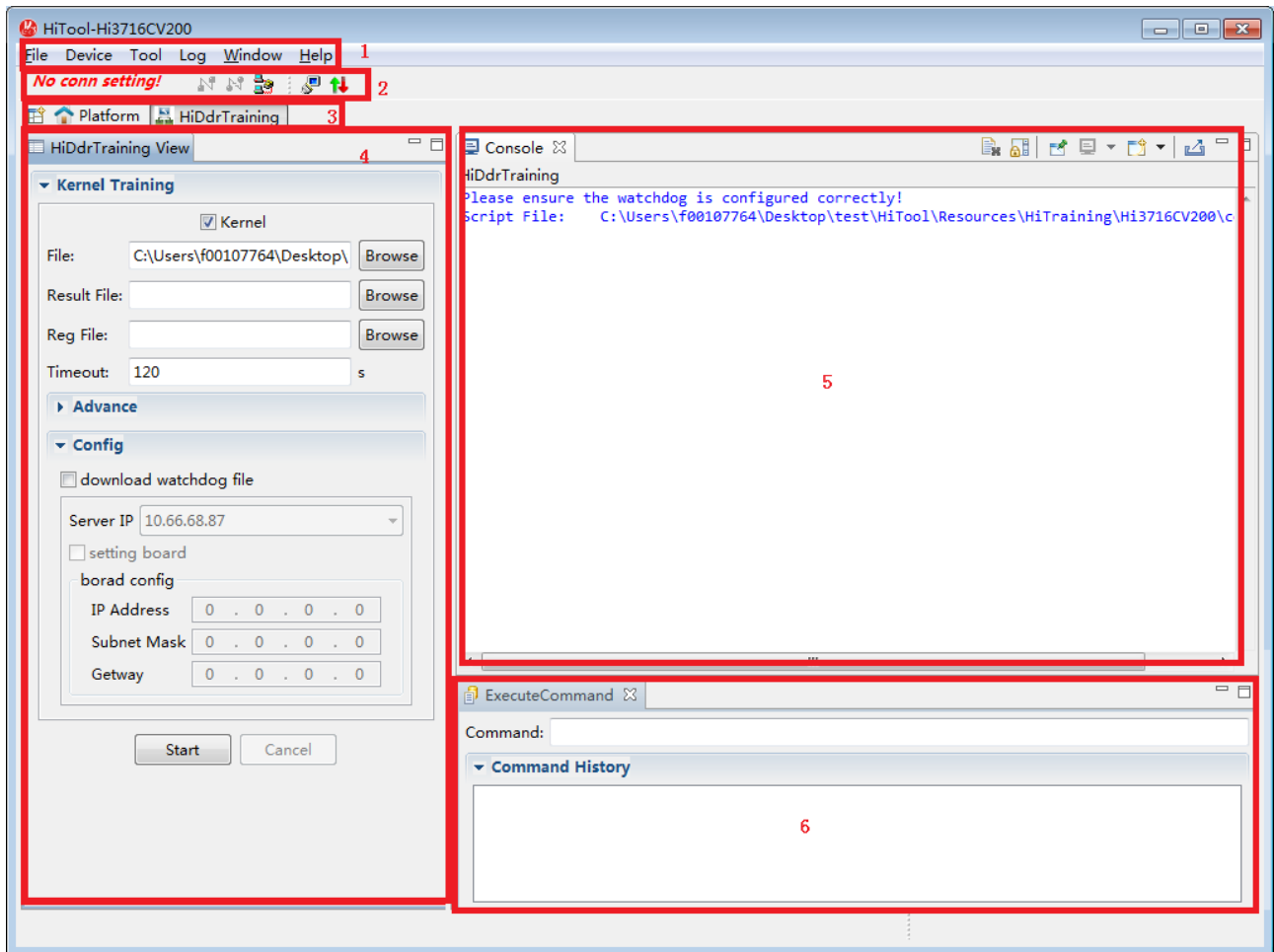


图1-2 HiDdrTraining 的按钮





图1-3 HiDdrTraining 主界面



- 1. 菜单栏
- 2. 工具栏
- 3. 透视图
- 4. HiDdrTraining 配置视图
- 5. 控制台视图
- 6. 命令视图

🔑 窍门

命令视图可通过快捷键 Ctrl+r 打开，也可通过菜单栏中窗口->显示视图->命令视图打开，命令视图默认不开启。



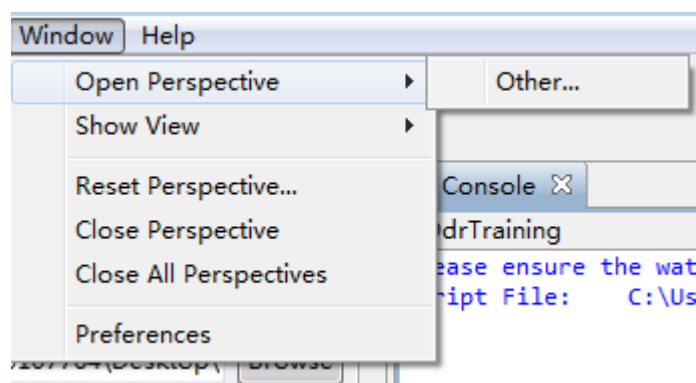
2 界面及功能说明

2.1.1 菜单栏

除了从 HiDdrTraining 按钮上可以打开工具，还可以选择菜单栏打开 HiDdrTraining 视图的，操作步骤如下：

步骤 1 选择菜单栏中窗口，点击打开透视图（O），如图 2-1 所示。

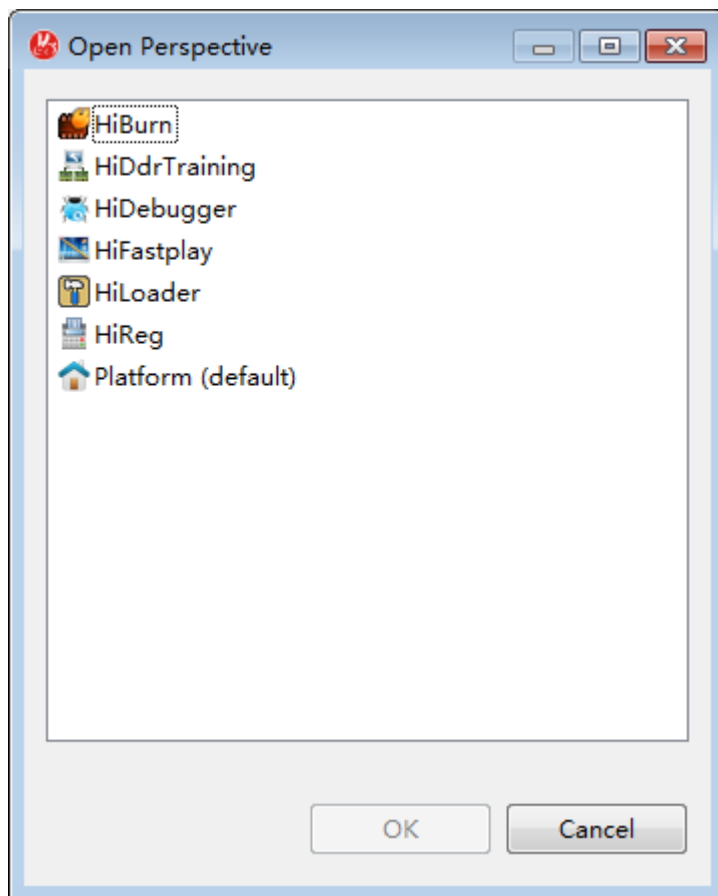
图2-1 HiDdrTraining 透视图菜单栏



步骤 2 在打开的选择框中选择 HiDdrTraining 透视图，如图 2-2 所示。



图2-2 打开透视图对话框



步骤 3 通过 HiDdrTraining 透视图工具栏切换到 HiDdrTraining 透视图，如图 2-3 所示，即可。

图2-3 HiDdrTraining 透视图工具栏



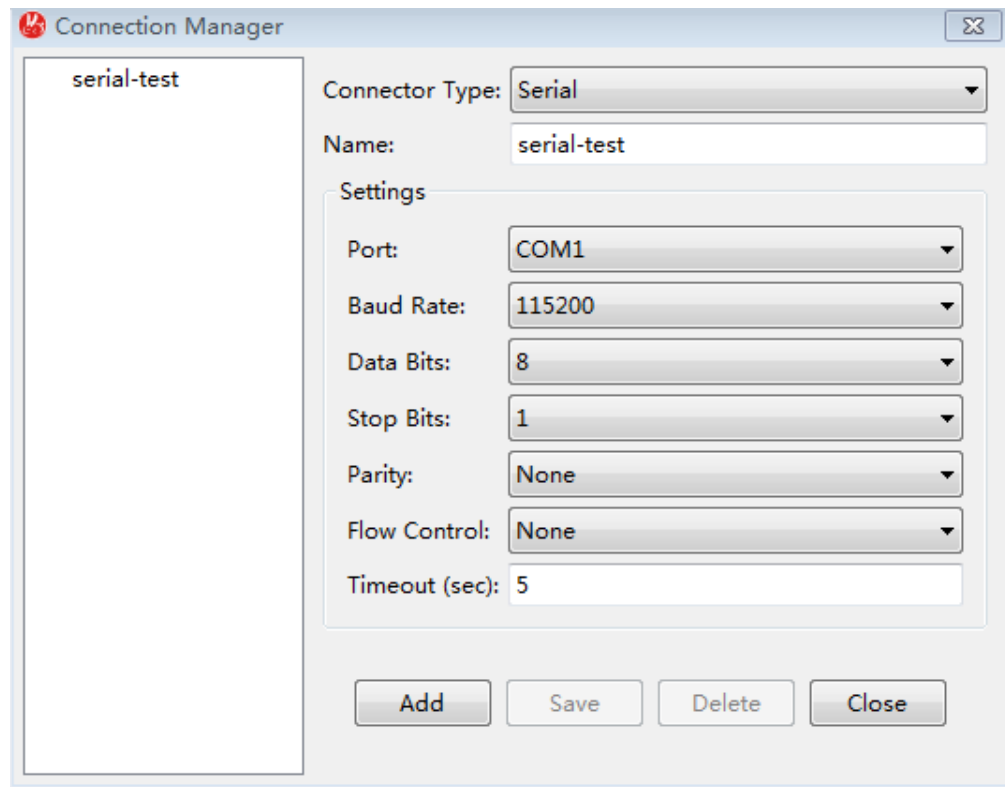
----结束

2.1.2 工具栏

此节将说明链接配置和管理的相关操作，目前 HiDdrTraining 只支持串口连接，具体操作步骤如下：

步骤 1 在工具栏中选择  按钮，打开连接管理器对话框，如图 2-4 所示，可以添加、保存、删除串口连接和网口连接，因为 HiDdrTraining 目前只支持串口连接，所以只需要添加一个串口连接。

图2-4 连接管理器对话框



步骤 2 在连接类型下拉框中，选择 Serial 项，即为串口；点击“添加”按钮，可以看见添加的串口连接已经出现在左边栏中；然后可以对其配置进行相应的修改，串口连接配置包括：

- 端口：COM1
- 波特率：115200
- 数据位：8
- 停止位：1
- 奇偶效验：None
- 数据流控制：None
- 超时：5 秒


步骤 3 配置完毕后，然后保存，如图 2-4 所示。

设置好连接后，保存后在主界面的工具栏上会显示保存的连接，如图 2-5 所示。

图2-5 活动连接





步骤 4 选择串口连接方式连接单板，如果打开主页面时，已经配置好串口，则直接在工具栏下拉框中选择相关连接，点击工具栏上的 ，与单板进行连接，反之则根据上述步骤先添加连接并保存配置信息。

----结束



注意

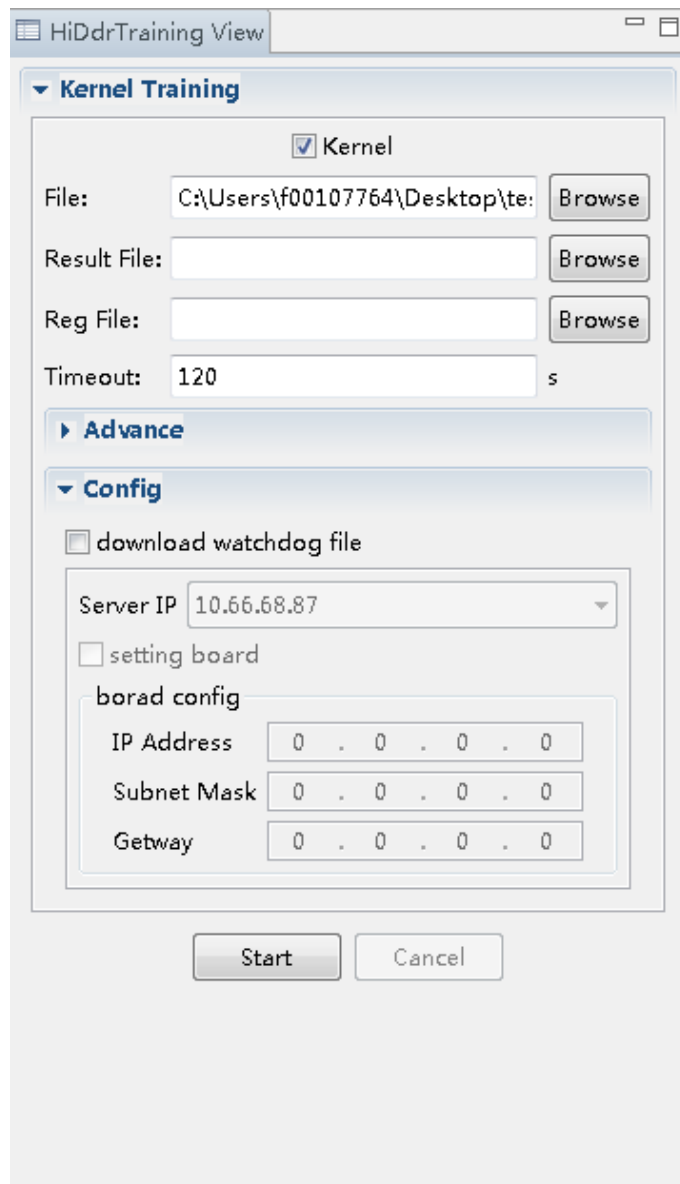
- 单板上最好运行一个业务用例，使用的业务用例一定不能独占串口，否则会导致工具下发数据失败；
- 业务测试用例尽可能使 DDR 的压力最大化，尽量多个模块同时访问 DDR。
- 客户可以通过 FAE 获取我们推荐的 DDR 测试用例。

2.1.3 HiDdrTraining 配置视图

HiDdrTraining 可以自动执行 DdrTraining 脚本，计算出最优值，并将结果合入 Reg 文件，配置视图如 [图 2-6](#) 所示。



图2-6 HiDdrTraining 视图



Kernel 配置

工具打开后就会根据芯片选择默认配置好的文件，默认文件存放在 HiTool 的本地目录中([HiTool 根目录]/Resources/HiTraining/[当前芯片名]/)下。

- 脚本文件名称为：config.xml
- reg 文件的名称为：reg_info.bin
- 结果文件名称为：result.txt
- watchdog 文件名为：watchdog.bin

自定义选择文件时：



- 脚本文件输入框不能为空，且脚本文件必须存在。
- 如果 reg 文件输入框不为空时，那么输入框中指定的 reg 文件必须存在。
- reg 文件和结果文件输入框允许为空，当 reg 文件输入框为空，Kernel 方式执行 Training 完成后将不会保存最优值。
- 当结果文件输入框为空，HiTraining 会默认在 HiTool 本地目录([HiTool 根目录]/Resources/HiTraining/[当前芯片名])下自动创建一个 result.txt 文件，如果结果文件输入框指定的文件不存在，工具会自动创建该文件。



说明

Reg 文件是 boot 中包含的 ddr 参数文件,以 HiSTBLinuxV100R002C0XSPCXXX 发布包为例, reg.bin 可以在\source\boot\sysreg 下选择相应的硬件配置生成

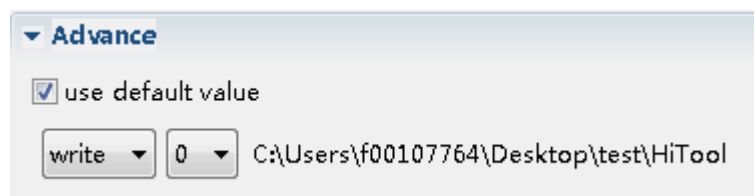
高级选项

当高级选项中的使用默认值复选框未选中时，用户可以自定义选择脚本文件、结果文件和 Reg 文件，另外通过高级选项配置，可以执行指定默认配置的 Kernel Training。高级选项中的脚本命名规则：

- 读脚本：config_read[数字].xml，如 config_read0.xml，config_read1.xml
- 写脚本：config_write[数字].xml，如 config_write0.xml，config_write1.xml

使用高级选项的默认文件配置时，高级选项中指定的脚本文件在 HiTool 的本地目录([HiTool 根目录]/Resources/HiTraining/[当前芯片名])下必须存在；Training 将不会保存寄存器的最优值和结果文件。

图2-7 高级选项设置



配置板端

只为了下载 watchdog 文件需要进行网络配置，如果板端已经存在 watchdog.bin 文件，并且可以正常工作，则不用重新配置；如不存在，可通过配置选项进行 tftp 下载，否则 DdrTraining 的测试会中断。

独占串口模式

如果业务的串口输出大量的打印，可能会影响工具数据的正常接收；所以当业务的串口输出打印量大的时候，推荐用户使用工具独占串口的 KO 模式，工具中需要进行一些配置，如下：

- 步骤 1 编译好独占串口的 Ko(mono_uart.ko)。这个模块默认是以模块的方式被编译的，对应的内核配置如下：

```
Device Drivers --->
```

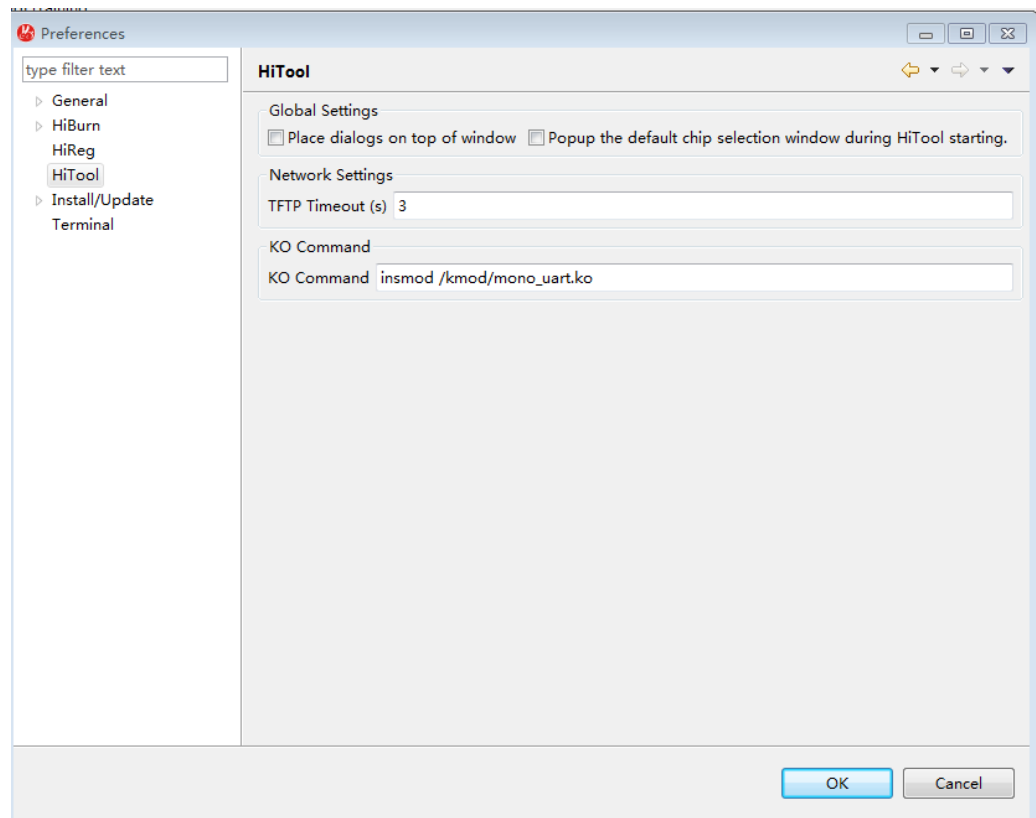


```
Character devices --->
Serial drivers --->
<M> Mono serial support
```

编译好的文件位于内核源码的 drivers/serial 目录下。

步骤 2 选择菜单栏 窗口->首选项，在首选项中，点击 HiTool->Ko 命令，如图 2-8 所示，把插入 Ko 的命令输入正确，文件的名称及路径严格区分大小写，这样工具会自动在 DdrTraining 前插入配置的 Ko。

图2-8 Ko 命令配置



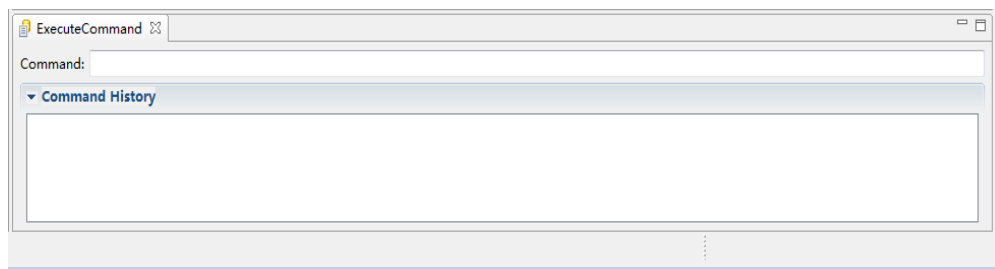
----结束

2.1.4 执行命令视图

命令视图可通过快捷键 Ctrl+r 打开，也可通过菜单栏中窗口->显示视图->命令视图打开此视图中，通过在命令文本框输入 Linux 可执行命令，按下回车后执行，可进行一些板端操作，如图 2-9 所示。

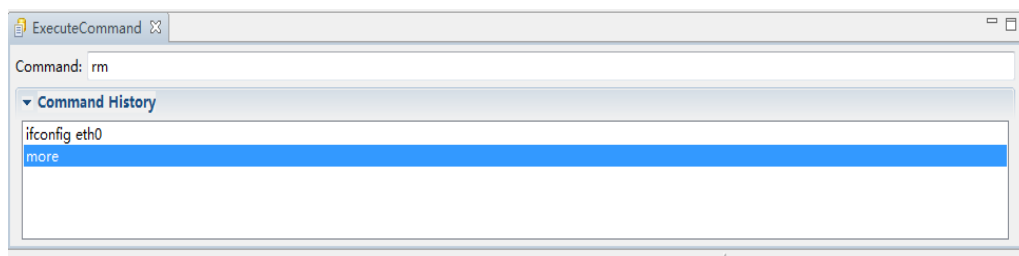


图2-9 输入可执行命令



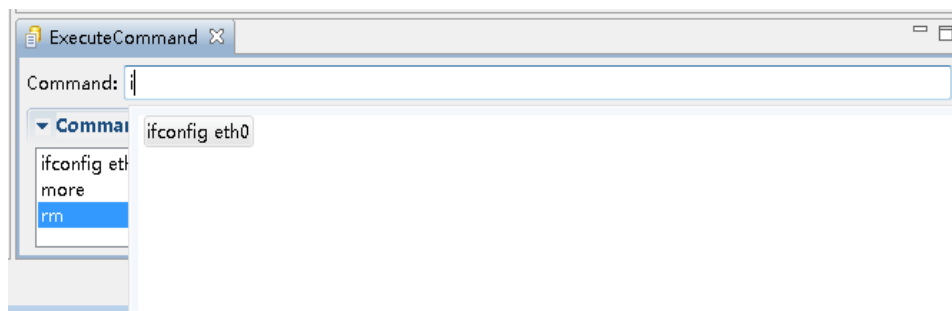
命令历史记录列表展示了输入的历史命令，如图 2-10 所示。

图2-10 命令历史记录列表



当在命令文本框中输入命令时，它会根据当前输入的字符从命令历史记录列表中匹配相似的命令，提示命令将会以弹出框形式提供（注意：在匹配中，命令历史记录中相同的命令将会被过滤），如图 2-11 所示。

图2-11 弹出自动提示框

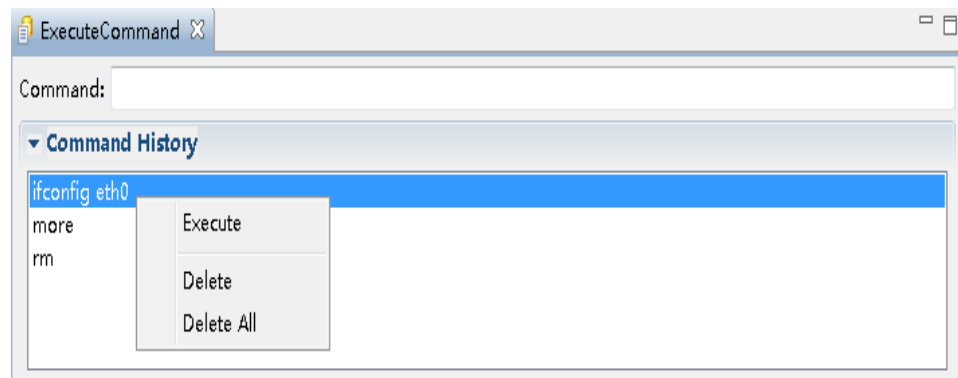


命令历史记录中的右键功能还可以进行执行和删除等操作，如图 2-12，同时当前历史命令列表中，可鼠标双击任意一条命令进行执行，利用上下键盘方向键，可翻取历史命令显示在命令输入框中。

- 执行当前历史命令列表中选中的命令。
- 删除当前历史命令列表中选中的命令。
- 删除全部当前历史命令列表中的命令。



图2-12 命令历史列表中的右键功能



2.1.5 控制台视图

当单板、网络、串口、文件等配置都根据上述配置完毕后，点击开始后，工具会把与单板所有的交互以及结果打印显示在控制台区。



3 HiDdrTraining 运行原理及脚本

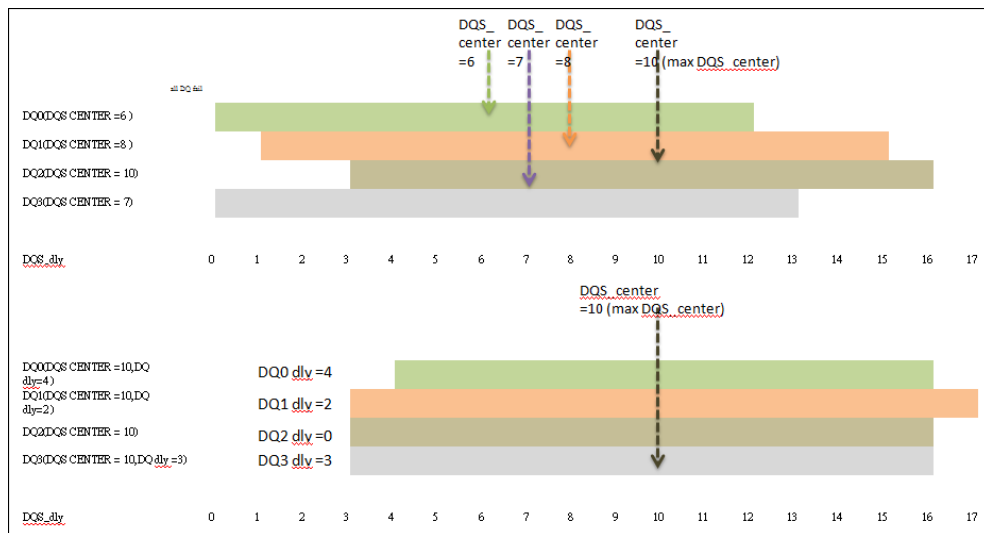
3.1 HiDdrTraining 运行原理

3.1.1 HiDdrTraining 原理

目前使用的 DDR PHY 中都有可以调整 DQ 或者 DQS 信号线上的延时功能，该延时就是用来补偿 DDR 走线不等长的问题。特别是在两层 PCB 上，为了走线方便，DQ 信号线的长度差异很大，如果芯片内部不做补偿，窗口将会是非常的小。

如图 3-1 所示，DDR 的时序窗口为各个 DQ 信号的交集部分，如果不做 DDR Training 交集会比较小，做过 DDR Training 后时序窗口将会增大。

图3-1 DDR Training 前后时序对比



HiDdrTraining 工具调整的原理与 Fastboot 下调整的原理相似，不同点就是把 boot 下调整使用的 DDRT 或者其它业务模块(VEDU)更换为实际的业务，这样芯片内部的干扰和 DDR 的压力更大，Training 得到的结果也更准确。

以 Hi3716CV200 为例，Training 时按照 byte 为一组，因为 DQS 信号的调整会影响到该 byte 内部的所有窗口，且 DQ 信号的调整范围比较大（32 级 BDL(bit delay line)，每个



级别 25ps)，因此不对 DQS 信号进行调整，直接使用 boot 表格中的 DQS 信号的配置。只调整 DQ 信号的 BDL 值。

调整的步骤如下：

- 步骤 1** 在 linux 环境下配置测试业务：PIP+TDE+GPU+DDRT，设置开机自动启动业务。开启 watchdog 功能设置 5 秒喂狗一次，如果没有喂狗则 watchdog 会对单板进行重启。
- 步骤 2** 读取 DQ 信号的 BDL 的默认配置。
- 步骤 3** 从默认的 DQ BDL 开始，增加 BDL 值，等待 30s（时间可调）。
- 如果 30s 没有检测到单板重启，记录该 BDL OK。
 - 如果 30s 检测到单板重启，记录该 BDL 错误。
- 步骤 4** 重复**步骤 3**直到 BDL 的最大值。
- 步骤 5** 从默认的 DQ BDL 开始，逐渐减小 BDL 直到单板重启，等待 30s。
- 如果 30s 内没有检测到单板重启，记录该 BDL OK。
 - 如果 30s 内检测到单板重启，记录该 BDL 错误。
- 步骤 6** 重复**步骤 5**直到 BDL 的最小值。
- 步骤 7** 统计记录 OK 的 BDL 级数作为该 DQ 信号的窗口大小，取中间的 BDL 值作为最优配置。
- Hi3716CV200 DQ 信号可调 32 级 BDL 窗口，不需要调整 DQS 的 BDL 窗口；
 - Hi3716MV300 DQ 与 DQS 只可以调整 8 级 BDL 窗口，所以调整时使用 DQS 与 DQ 的组合以后的 15 级 BDL 窗口；
 - Hi3716CV100 和 Hi3531 读写方向调整的级别不一样，Hi3716CV100 写方向可以调整 29 级 BDL 窗口，读方向可以调整 15 级 BDL 窗口。

表3-1 Hi3716CV100 DQ/DQS 组合 BDL 级别

写窗口级别	读窗口级别	DQS 级别	DQ 级别
0	-	关	7
1	-	关	6
2	-	关	5
3	-	关	4
4	-	关	3
5	-	关	2
6	-	关	1
7	0	0	7
8	1	0	6
9	2	0	5



写窗口级别	读窗口级别	DQS 级别	DQ 级别
10	3	0	4
11	4	0	3
12	5	0	2
13	6	0	1
14	7	0	0
15	8	1	0
16	9	2	0
17	10	3	0
18	11	4	0
19	12	5	0
20	13	6	0
21	14	7	0
22	-	1	关
23	-	2	关
24	-	3	关
25	-	4	关
26	-	5	关
27	-	6	关
28	-	7	关

表3-2 Hi3716MV300 DQ/DQS 组合 BDL 级别

写窗口级别	读窗口级别	DQS 级别	DQ 级别
0	0	7	0
1	1	6	0
2	2	5	0
3	3	4	0
4	4	3	0
5	5	2	0
6	6	1	0



写窗口级别	读窗口级别	DQS 级别	DQ 级别
7	7	0	1
8	8	0	2
9	9	0	3
10	10	0	4
11	11	0	5
12	12	0	6
13	13	0	7

表3-3 Hi3716CV200 DQ BDL 级别

写窗口级别	读窗口级别	DQ 级别
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15
16	16	16
17	17	17
18	18	18
19	19	19



写窗口级别	读窗口级别	DQ 级别
20	20	20
21	21	21
22	22	22
23	23	23
24	24	24
25	25	25
26	26	26
27	27	27
28	28	28
29	29	29
30	30	30
31	31	31

----结束

3.1.2 HiDdrTraining 运行过程

由于现在所有的芯片已经预置了脚本，可供用户直接使用，如果需要更新脚本的用户可以参考工具的详细执行流程来自行开发脚本，运行的步骤如下：



说明

- 当 registergroup 中只定义了一个 register 时，以下称为 Single。
- 当 registergroup 中定义了多个 register 时，以下称为 Multiple。

步骤 1 开始读取配置文件。

步骤 2 启动 watchdog。

步骤 3 执行 kocommand，kocommand 从首选项配置中加载，执行 regtool 命令。

步骤 4 读取需要遍历的寄存器保存起来，记为初始值。

步骤 5 处理 loops(loops 对应于一个 byte)，然后循环处理每一个 loops 里面的 loop(loop 对应于 dq 信号)。

步骤 6 处理 loop_level_begin。

判断当前 registergroup 的 initvalue 是否为 true，为 true，则打印当前 loop 中所有寄存器的初始值并保存。

- 处理 single。
 1. 当 defaultvalue 等于-1



当设置了 **minvalue**(最小 DQ BDL 级别):

- 循环更新 **register** 的值(**register** 值即为 DQ BDL 级别), 写回设备, 并检测设备是否重启。
- 更新规则: 当前值=当前值-1 (当前值的初始值根据配置的位范围以及寄存器从板端读取的值确定。)
- 循环条件: 当前值 \geq **minvalue**。
- 循环执行一次, 检测设备是否重启, 等待时间可在 UI 界面上的超时时间文本框控制, 并输出当次执行结果。
- 循环结束后, 当前 **registergroup** 中的所有寄存器会进行复位。

当设置了 **maxvalue**: (最大 DQ BDL 级别)

- 循环更新 **register** 的值, 写回设备, 并检测设备是否重启。
- 更新规则: 当前值=当前值+1 (当前值的初始值根据配置的位范围以及寄存器从板端读取的值确定。)
- 循环条件: 当前值 \leq **maxvalue**。
- 循环执行一次, 检测设备是否重启, 等待时间可在 UI 界面上的超时时间文本框控制, 并输出当次执行结果。
- 循环结束后, 当前 **registergroup** 中的所有寄存器会进行复位。

2. 当 **defaultvalue** 不等于-1

将默认值写入设备, 检测设备是否重启, 等待时间可在 UI 界面上的超时时间文本框控制, 并输出当次执行结果。写入成功后, 当前 **registergroup** 中的寄存器会等待其所在的 **group** 中所有的 **registerGroup** 内全部寄存器执行结束后, 这个寄存器的值才会进行复位。

● 处理 Multiple

1. 当设置了 **minvalue**:

- 循环更新 **register** 的值, 写回设备, 并检测设备是否重启。
- 更新规则: 当前值=当前值-1 (当前值的初始值根据配置的位范围以及寄存器从板端读取的值确定, 每一个寄存器的当前值是独立的)。
- 循环条件: 寄存器中当前值的最小值 $>$ **minvalue**。
- 循环执行一次, 检测设备是否重启, 等待时间可在 UI 界面上的超时时间文本框控制, 并输出当次执行结果。
- 循环结束后, 当前 **registergroup** 中的所有寄存器会进行复位。

2. 当设置了 **maxvalue**:

- 循环更新 **register** 的值, 写回设备, 并检测设备是否重启。
- 更新规则: 当前值=当前值+1 (当前值的初始值根据配置的位范围以及寄存器从板端读取的值确定, 每一个寄存器的当前值是独立的)。
- 循环条件: 寄存器中当前值的最大值 $<$ **maxvalue**。
- 循环执行一次, 检测设备是否重启, 等待时间可在 UI 界面上的超时时间文本框控制, 并输出当次执行结果。
- 循环结束后, 当前 **registergroup** 中的所有寄存器会进行复位。



说明

如果在执行过程中 **boot** 没有自动重启, 工具将会提示手动重启。



步骤 7 所有 loops 中 loop_level 全部执行完毕后，查找合并 reg 文件，处理并打印保存最优值结果。详细可参考下面说明。

- 求解最优值的规则



说明

根据每一个 loop 的执行结果集合，每一个 loop 取一个最优值。

- 执行结果状态无 Fail 时，取结果集合最中间值作为 best 值。
- 存在一个 Fail 时，判断 Fail 距开头的间距和 Fail 距结尾的间距的大小，哪个间距最大，得到间距最大部分，取最大部分的中间值作为 best 值。
- 存在多个 Fail 时，判断开头离最近 Fail 的间距，结尾离最近 Fail 的间距，以及每个 Fail 之间的间距，哪个间距最大，得到间距最大部分，取最大部分的中间值作为 best 值。

- 处理结果

- 如果 Training 类型为 byteTraining，工具将直接把每个 loop 的执行结果以及最优值打印到控制台上。
- 如果 Training 类型为 bitTraining，工具将会根据脚本的配置对每个 loop 的最优值进行处理。具体处理步骤如下，分两种情况处理：

loops 指定了最大值寄存器。

- 取到 loops 指定寄存器在所有 loop 中的最优值的最大值。
- 取到每个 loop 指定的寄存器的关键值。
- 计算得到最后整个 loops 的最优值组合。最优值组合包含了每个 loop 的关键最优值，关键最优值的计算公式为：（每个 loop 的关键最优值=最大值-每个 loop 中 loops 指定寄存器的最优值+每个 loop 的关键值）
- 计算 DM 值，把每个 loop 的关键最优值相加，取平均值，小数情况下四舍五入取整。

loops 没有指定最大值寄存器。

- 取到每个 loop 指定的寄存器的关键值。
- 计算得到最后整个 loops 的最优值组合。最优值组合包含了每个 loop 的关键最优值，每个 loop 的关键最优值=每个 loop 的关键值。
- 计算 DM 值，把每个 loop 的关键最优值相加，取平均值，小数情况下四舍五入取整。

最后把 bittraining 的执行结果打印出来，打印内容包括：每一个 loops 的最优值组合以及 DM 值。

- 结果格式



说明

只是作为打印结果格式的参考，执行数据应以工具执行的数据为准。dqs0, dq0, dq1, dq2, dq3, dq4, dq5, dq6, dq7 表示寄存器的名称。寄存器名称必须在 loop 的 description 中存在定义。

每次执行一个 loop，先打印 description。以 single 循环，默认的 dqs 为 3 为例，



表3-4 以 single 循环，默认的 dqs 为 3 举例

dqs0	dq0	dq1	dq2	dq3	dq4	dq5	dq6	dq7	Result
7	0	0	0	0	2	2	2	1	fail
6	0	0	0	0	2	2	2	1	ok
5	0	0	0	0	2	2	2	1	ok
4	0	0	0	0	2	2	2	1	ok

表3-5 以 Multiple 循环举例

dqs0	dq0	dq1	dq2	dq3	dq4	dq5	dq6	dq7	Result
3	1	1	1	1	3	3	3	2	ok
3	2	2	2	2	4	4	4	3	ok
3	3	3	3	3	5	5	5	4	fail

到第一个 fail 的就结束这个小项的循环，取中间一行为最佳值，也就是 4 排（0 0 0 0 2 2 2 1）组，同时打印出最佳值和默认值，如表 3-6。

表3-6 输出最佳值和默认值

值类型	dqs0	dq0	dq1	dq2	dq3	dq4	dq5	dq6	dq7
默认值	3	0	0	0	0	2	2	2	1
最佳值	4	0	0	0	0	2	2	2	1

当 training 类型为 bittraining 时，还会打印以下内容，无最大值时，将不会打印最大值那一列。

- MaxRegister: loops 指定的寄存器名称
- MaxValue: loops 指定的寄存器的最大值

表3-7 有效值组合举例（最大值列和每个 loop 的关键寄存器值）

loop 名称	最大值	关键值 1	关键值 2
Loop1	0	0	-
Loop2	0	-	0



表3-8 整个 loops 的最佳值组合举例

值类型	最大值	关键值 1	关键值 2
最佳值:	0	0	0

DM: 0

- 查找合并
从输入 Reg 文件中查找匹配相同地址进行合并，并且更新合并后的值。
保存结果:



注意

training 执行失败或中途取消操作，将不会保存结果到结果文件。

training 正确执行成功后，保存 training 的最优结果到一个 reg.bin 文件，reg.bin 文件的格式有下面结构体连接而成，如我们要把 rddqs0 和 rddqs1 的值都设为 4 (address="0x10100b4c" start="0" end="2", name="rdqs1" address="0x10100b4c" start="3" end="5"), 那么寄存器 0x10100b4c 的值就应该修改为 0x24。

```
struct regentry {  
    unsigned int reg_addr;  ---- 寄存器地址  
    unsigned int value;  -----寄存器的值  
    unsigned int delay;  
    unsigned int attr;  
};
```

表3-9 Reg 文件格式

Struct	Struct	Struct	Struct
regentry	regentry	regentry	Struct

3.2 脚本文件格式定义

3.2.1 节点之间关系

- training、commands 只能出现一次。
- training 节点为根节点，training 包含 commands 和多个 loops 节点。
- commands 节点里面可以包含多个 command 节点。



- loops 里面包含多个 loop 节点，loop 里面包含多个 group 节点。
- group 里面包含多个 registergroup 节点。
- registergroup 包含多个 register 节点。

3.2.2 节点包含的属性介绍

- training 节点包括 mode 属性，mode 属性定义了 Training 模式，mode 属性值可选 "byte" 和 "bit"。两者只能选其一。
说明：当 training 节点上不包含 mode 属性时，默认 mode 类型为 "byte"。
- command 节点包括 value 属性，value 属性定义了执行的命令字符串，value 属性值不能为空，当命令中包括符号时，必须使用此符号的 html 转义符。
- loops 节点上包括 max 属性，max 属性定义了 loops 指定的最大寄存器的名称。
说明：
 - 当 mode 为 "bit" 时，loops 节点可包含 max 属性，也可不包含。当包含 max 属性时，定义的属性值不能为空。max 属性的存在性会使工具对 bittraining 结果有不同的处理。



注意

max 属性值定义的最大寄存器必须在当前 loops 的每一个 loop 都存在定义。

- 当 mode 为 "byte" 时，因为 max 属性只在 bit 模式时才起作用，所以 byte 模式下 max 属性可定义也可不定义。
 - loop 节点包含 key 和 description 属性，不能为空。
 - key 属性定义了当前 loop 指定的关键寄存器名称；
 - description 属性定义了当前 loop 节点下的所有寄存器名称，每个 loop 节点中所有寄存器的名称必须在 loop 节点上的 description 中存在定义，否则无效。
 - description 属性值中定义的每个寄存器名称之间用英文逗号分隔，第一个名称将被忽略，名称不能存在重复值。如：第一个 "read" 将被忽略，实际的寄存器名称为："rdqs0, rdq0, rdq1, rdq2, rdq3, rdq4, rdq5, rdq6, rdq7"。
- 说明：
- 当 mode 为 "bit" 时，loop 节点上的 key 属性必须定义，且 保证定义的 key 属性值在当前 loop 中的任意一个 registergroup 中的 register 列表中存在定义。
 - 当 mode 为 "byte" 时，因为 key 属性只在 bit 模式时才起作用，所以 byte 模式下 key 属性可定义也可不定义。
 - group 节点无属性值。
 - registergroup 节点包括 initvalue、sequence、maxvalue、minvalue 和 defaultvalue 属性。如下：
 - initvalue 值指定当前 registergroup 是否只是输出初始值，值可取 true 或 false，默认为 false，当为 true 时，该 registergroup 的其他属性全部无效，加载时不做校验；否则，加载并验证其他属性配置。



- sequence 值控制 registergroup 结果的排列输出顺序，值不能为空；值可取 true 或 false，为 true 时，顺序排列输出，否则，逆序排列输入；
- maxvalue 定义了当前节点下所有寄存器循环执行的最大值；
- minvalue 定义了当前节点下所有寄存器循环执行的最小值；
- defaultvalue 定义了当前节点下所有寄存器的默认值；

说明：

- registergroup 节点中的 maxvalue 和 minvalue 属性定义时，只能二选一。即 registergroup 节点上 maxvalue 和 minvalue 属性任何时候只能存在一个。
- 除 initvalue 为 true 外，maxvalue，minvalue 和 defaultvalue 三者必须定义一个。
- register 节点下包括 name、address、start 和 end 属性，name 定义了当前寄存器的名称；address 定义了寄存器地址；start 定义了截取寄存器的起始位置；end 定义了截取寄存器的结束位置。

3.2.3 脚本书写注意事项

- 必须包含 xml 的声明，编码格式为 UTF-8。
- 脚本文件所有节点均是成对出现，且脚本中所有节点名称为小写，书写必须符合 xml 规范。
- 在 loop 节点下，当两个寄存器的 address，start 和 end 属性值都相同时，那么这两个寄存器的 name 属性值也必须相同，否则，当有一项不同时，两个寄存器的 name 属性值必须不同。
- 脚本文件后缀名建议为 “.xml”，脚本文件可以参考已经提供的 xml 文件作为样例参考。



4 FAQ

4.1 选择 Kernel Training 时，应注意什么？

问题描述

Kernel Training 的时候，环境和操作有什么特别需要注意的吗？

解决办法

- 工具需要 watchdog.bin 用于自动重启单板，故需要保证板端已经存在 watchdog.bin 文件，并且可以正常工作，可以自动运行业务；
- 工具是通过串口与板端进行交互,所以需要保证业务程序不能独占串口。需要加载一个名为 mono_uart.ko 的模块，这个模块默认是以模块的方式被编译的，对应的内核配置如下：

```
Device Drivers --->
  Character devices --->
    Serial drivers --->
      <M> Mono serial support
```

编译好的文件位于内核源码的 drivers/serial 目录下。

4.2 下载 watchdog 脚本失败，可能有什么原因呢？

问题描述

收到类似“Nothing received from the device for the command: tftp -r watchdog.bin X.X.X.X -g Remote no responded, stop executing training.”报错，一般是下载 watchdog 脚本失败，那有哪些可能的原因呢？

解决办法

- 可能是由于在 HiTool 本地存放目录（HiTool /Resources /HiTraining/ 当前运行的芯片名）下没有 watchdog.bin 文件，或者在板端配置时板端 IP，子网掩码和网关的配置有冲突。



- 如果报 “Failed to reset init value Remote no responded”，是因为硬件没有打开 watchdog。

4.3 错误 “Address xx not found, the value will be xx”，是什么原因呢？

问题描述

错误 “Address xx not found, the value will be xx”，是什么原因呢？

解决办法

脚本文件中的寄存器地址没有在 `reg_info.bin` 中没有匹配的寄存器，合并失败，所以请务必保证 `reg_info.bin` 文件和脚本文件相匹配。

4.4 kernel training 的时候出现文件系统被破坏，重启也不能恢复？

问题描述

kernel training 的时候出现文件系统被破坏，重启也不能恢复？

问题分析

这种情况多出现在 nand 的 yaffs 的文件系统，且双核出现的几率较高，主要原因是因为 yaffs 文件系统本身系统会定时刷新 ddr 的数据到 flash 中，我们 kernel training 的过程中 ddr 是处于不稳定状态，当在双核这种并行系统下，就容易出现文件系统被写破坏。

解决办法

建议可以做如下的处理：把文件系统 remount 只读，例如：`mount / -o remount,ro -o noatime`。可以使用 `spl/jffs2` 系统。



A 缩略语

I		
IP	Internet Protocol	计算机网络相互连接进行通信而设计的协议。
T		
TFTP	Trivial File Transfer Protocol	TCP/IP 协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议。