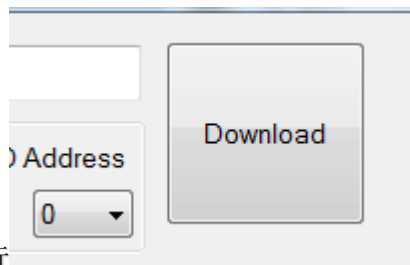


IMX6 DDR 测试工具使用

DDR 测试工具有两个版本，一个是 2.6,一个是 2.52 我一般用 2.52 版本。



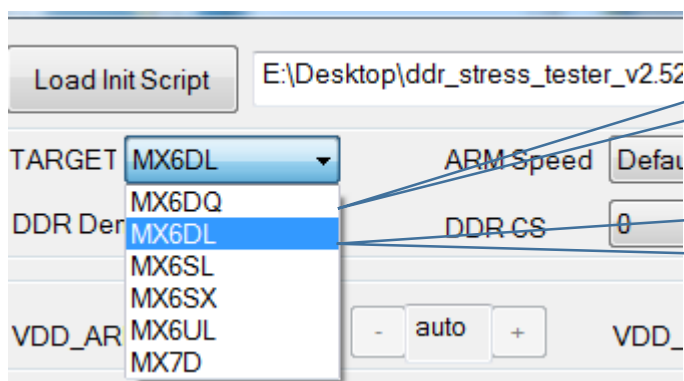
目录名字必须是英文的，否则在运行

下载会出现 ERROR

ERROR: can not open script file

所以整个路径都必须是英文

软件选项

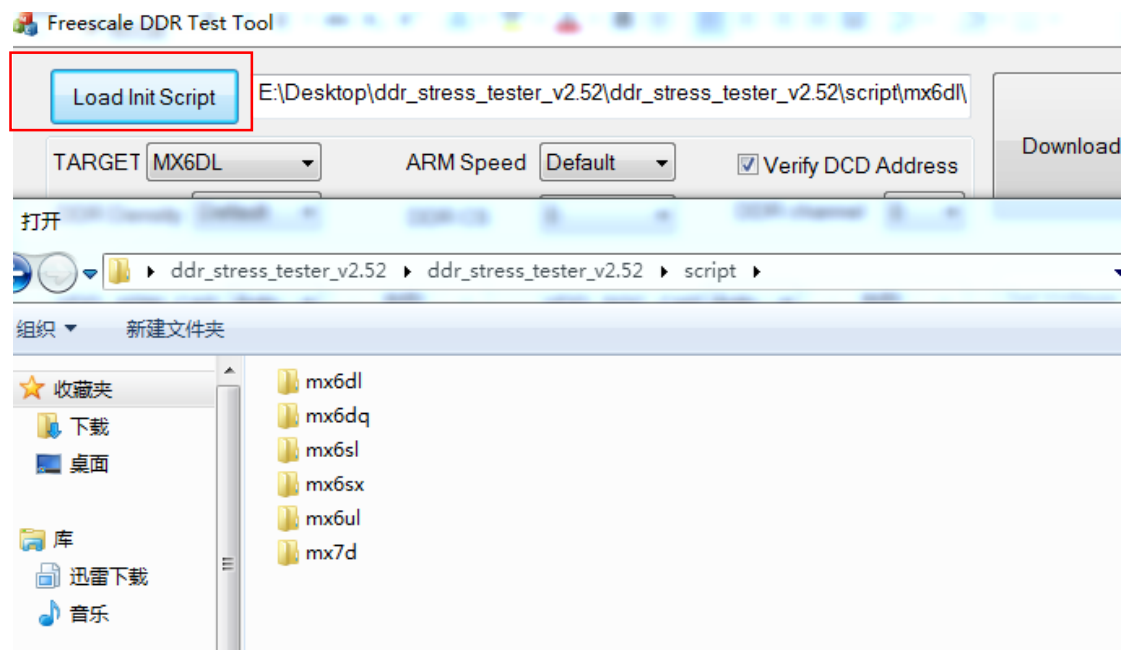


IMX6Q 芯片选择 MX6DQ

IMX6DL 芯片和 IMX6solo 芯片都选择 MX6DL

因为 IMX6DL 和 IMX6solo 都是一样

的,只是 IMX6DL 比 IMX6solo 多一个核。记住将开发板拨码开关拨到 00001100, 这样才行。



点击 Load Init Script 选择脚本，如果使用的是 IMX6dl 或者 IMX6solo 那么就选择 MX6dl，如果是 IMX6Q 就选择 MX6qd。
然后点击 DOWNload

ARM Clock set to 1GHz

```
=====
DDR configuration
BOOT_CFG3[5-4]: 0x00, Single DDR channel.
DDR type is DDR3
Data width: 64, bank num: 8
Row size: 15, col size: 10
Chip select CSD0 is used
Density per chip select 2048MB
=====
```

这就是正常访问芯片了。

如果换了个 DDR 型号的芯片，就要对 DDR 芯片进行校正。



Write DQS5 delay: 46/256 CK
Write DQS6 delay: 71/256 CK
Write DQS7 delay: 49/256 CK

WARNING: write-leveling calibration value is greater than 1/8 CK.
Per the reference manual, WALAT must be set to 1 in the register MDMISC(0x021B0018).
This has been performed automatically.
However, in addition to updating the calibration values in your DDR initialization,
it is also REQUIRED change the value of MDMISC in their DDR initialization as follows:

MMDC_MDMISC (0x021b0018) = 0x00011740

Starting DQS gating calibration
.....

进入 DDR 测试运行状态

最后会生成一个 DDROFFSET 报表

Starting Write calibration...

ABS_OFFSET=0x00000000	result[00]=0x11111110
ABS_OFFSET=0x04040404	result[01]=0x10101000
ABS_OFFSET=0x08080808	result[02]=0x00100000
ABS_OFFSET=0x0C0C0C0C	result[03]=0x00000000
ABS_OFFSET=0x10101010	result[04]=0x00000000
ABS_OFFSET=0x14141414	result[05]=0x00000000
ABS_OFFSET=0x18181818	result[06]=0x00000000
ABS_OFFSET=0x1C1C1C1C	result[07]=0x00000000
ABS_OFFSET=0x20202020	result[08]=0x00000000
ABS_OFFSET=0x24242424	result[09]=0x00000000
ABS_OFFSET=0x28282828	result[0A]=0x00000000
ABS_OFFSET=0x2C2C2C2C	result[0B]=0x00000000
ABS_OFFSET=0x30303030	result[0C]=0x00000000

如果你没有换其他型号的内存，这里就已经结束操作了。

如果你换了其他型号的内存，那么在没有下载程序进开发板之前，就要先执行这个 DDR_stream 软件，生成上面这个 ABS_OFFSET 列表。将列表里面的内容提取出来。

注意只需要修改 ABS_OFFSET 列表里面的东西，其他不用动。

内核 uboot 文件中找到使用 ABS_OFFSET 的代码，然后按照 DDR_stream 生成的 OFFSET 对 uboot 里面的 offset 进行修改。然后从新编译 UBOOT，烧写进内核。

Write calibration
MPWRDLCTL PHY0 (0x021b0850) = 0x3832302E
MPWRDLCTL PHY1 (0x021b4850) = 0x38363432

Success: DDR calibration completed!!!

这就是 DDR 测试成功。

DDR Calibration

MR1 Value(HEX) 0000

DDR Freq(MHz) 400

Calibration Save Result

DDR Stress Test

☐ Over Night Test

Start Freq(MHz) 0

End Freq(MHz) 0

Stress Test Save Result

32bit Memory Read/Write

ADDR(HEX)

SIZE 1 WORD Read

DATA(HEX) Write

这个是 DDR 压力测试。

IMX6 DDR 内存大小修改指南

参考 MX6X_3.14.28_Uboot_V1 版本资料

下载 yocto 环境，在 yocto 的

```
root@ubuntu:/home/xiang/yocto/fsl-release-bsp/imx6qsabresd-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/u-boot-imx/2014.04-r0/git#
```

这个目录下有 uboot 源码

按照 uboot 文档的要求

```
/home/vmuser/mx6x/fsl-release-bsp/imx6qsabresd-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/u-boot-imx/2014.04-r0/git
```

```
export ARCH=arm
```

```
export
```

```
CROSS_COMPILE=/opt/poky/1.7/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi
```

```
make mx6qsabresd_config
```

Configuring for mx6qsabresd - Board: mx6sabresd, Options:

```
IMX_CONFIG=board/freescale/imx/ddr/mx6q_4x_mt41j128.cfg,MX6Q,DEFAULT_FDT_FILE="imx6q-sabresd.dtb",DDR_MB=1024,SYS_USE_SPINOR
```

```
make
```

选择 IMX6 型号然后 make 就会生成 u-boot.imx 文件。这个文件就是给 MFG 下载进开发板使用的 uboot 文件。

但是以上配置是官方的 CPU 为 IMX6Q，DDR 为美光的 1G 内存的下使用的 uboot。

我现在要修改 CPU 型号和内存大小：

CPU : IMX6solo

DDR：美光，但是大小为 512M

```
mx6sabresd:IMX_CONFIG=board/freescale/mx6sabresd/mx6solo_4x_mt41j128.cfg,MX6SOLO,DEFAULT_FDT_FILE="mx6dl-sabresd.dtb",DDR_MB=512,SYS_USE_SPINOR,SYS_NOSMP="nosmp",ANDROID_SUPPORT
Fabio Estevam <fabio.estevam@freescale.com>
```

2. If we change the size, need to modify the follow codes,which define in

u-boot-2014.04-r0/boards.cfg as follows:

```
Active arm armv7 mx6 freescale mx6sabresd mx6qsabresd
mx6sabresd:IMX_CONFIG=board/freescale/imx/ddr/mx6q_4x_mt41j128.cfg,MX6Q,DEFAULT_FDT_FILE="imx6q-sabresd.dtb",DDR_MB=1024,SYS_USE_SPINOR
Fabio Estevam <fabio.estevam@freescale.com>
```

Change the size will change the DDR size, and after compile, it will set in the file (include\config.h) as follows:

```
#define CONFIG_IMX_CONFIG board/freescale/imx/ddr/mx6q_4x_mt41j128.cfg
```

```
#define CONFIG_DDR_MB 1024
```

按照上面的文档要求，在 uboot 目录下找到配置文件。先做 IMX6Q 的 uboot

```
root@ubuntu:/home/xiang/yocto/fsl-release-bsp/imx6qsabresd-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/u-boot-imx/2014.04-r0/git/include# vim config.
```

打开该文件

```
1 /* Automatically generated - do not edit */
2 #define CONFIG_IMX_CONFIG board/freescale/mx6sabresd/mx6q_4x_mt41j128.cfg
3 #define CONFIG_MX6DL 1
4 #define CONFIG_DEFAULT_FDT_FILE "imx6q-sabresd.dtb"
5 #define CONFIG_DDR_MB 1024
6 #define CONFIG_SYS_USE_SPINOR 1
7 #define CONFIG_SYS_ARCH "arm"
8 #define CONFIG_SYS_CPU "armv7"
9 #define CONFIG_SYS_BOARD "mx6sabresd"
10 #define CONFIG_SYS_VENDOR "freescale"
11 #define CONFIG_SYS_SOC "mx6"
12 #define CONFIG_BOARDDIR board/freescale/mx6sabresd
13 #include <config_cmd_defaults.h>
14 #include <config_defaults.h>
15 #include <configs/mx6sabresd.h>
16 #include <asm/config.h>
17 #include <config_fallbacks.h>
18 #include <config_uncmd_spl.h>
```

Dts 是识别 IMX6Q 的 dtb

DDR 是 1G

官方的配置是 IMX6Q 的 CPU

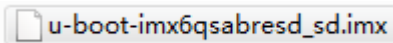
按照上面这个配置去先去 `make mx6qsabresd_config` 指定编译哪一个 CPU 型号
`make` 一下，生成的 `uboot` 文件是给 IMX6Q 使用的。

```
CFGS board/freescale/imx/ddr/mx6q_4x_mt41j128.cfg.cfgtmp
MKIMAGE u-boot.imx
OBJCOPY u-boot.srec
bot@ubuntu:/home/xiang/yocto/fsl-release-bsp/imx6qsabresd-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/u-boot-imx/2014.04-r0/git#
```

将文件拷贝到 MFG 上面。记住要把 `u-boot.imx` 改成符合 MFG 格式的名字

```
<CMD state="Updater" type="push" body="send" file="files/u-boot-imx6q%
plus%%board%.sd.imx" ifdev="MX6Q">Sending u-boot.bin</CMD>
```

根据 MFG 要求应该要改成这样的名字，但是不要加%号。这里有个问题？
实际的 MFG file 和 fireware 文件里面名字是这样的。



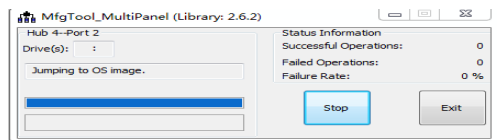
证明了%%里面的内容是可以修改的。MFG 索引的应

该是%号之前和%之后的内容，两个%之间的内容可以修改。

然后下载程序进开发板。

正常运行，你会发现 IMX6DL 和 IMX6Q 开发板之间的 `uboot` 可以相互使用。

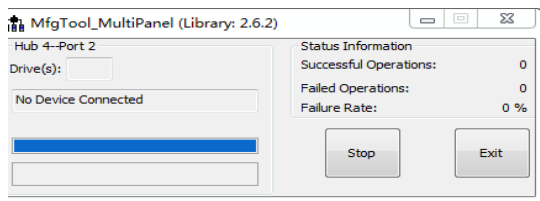
如果将 IMX6Q 的 UBOOT 烧写进 imx6solo DDR 512M 的开发板就会出现下面的错误：



```
i2c: ready
DRAM: 512 MiB
MMC: FSL_SDHC: 0, FSL_SDHC: 1, FSL_SDHC: 2
MMC: no card present
MMC init failed
Using default environment

No panel detected: default to Hannstar-XGA
Display: Hannstar-XGA (1024x768)
In: serial
Out: serial
Err: serial
Found PFUZE100 deviceid=10, revid=21
MMC: no card present
mmc1(part 0) is current device
Net: Phy not found
PHY reset timed out
FEC [PRIME]
Warning: failed to set MAC address

Boot from USB for mfgtools
Use default environment for mfgtools
Run bootcmd_mfg: run mfgtool_args;bootz ${loadaddr} ${
Hit any key to stop autoboot: 0
Kernel image @ 0x12000000 [ 0x000000 - 0x521260 ]
## Loading init Ramdisk from Legacy Image at 12c00000
Image Name: fsl-image-mfgtool-initramfs-imx6
Image Type: ARM Linux RAMDisk Image (gzip compres
Data Size: 7508220 Bytes = 7.2 MiB
Load Address: 00000000
Entry Point: 00000000
Verifying Checksum ... OK
## Flattened Device Tree blob at 18000000
Booting using the fdt blob at 0x18000000
Using Device Tree in place at 18000000, end 1800d9b
switch to ldo_bypass mode!
Starting kernel ...
```



```
ci_hdrc ci_hdrc.1: EHCI Host Controller
ci_hdrc ci_hdrc.1: new USB bus registered, assigned
ci_hdrc ci_hdrc.1: USB 2.0 started, EHCI 1.00
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
mousedev: PS/2 mouse device common for all mice
egalax_ts 1-0004: Failed to read firmware version
egalax_ts: probe of 1-0004 failed with error -5
egalax_ts 2-0004: Failed to read firmware version
egalax_ts: probe of 2-0004 failed with error -5
snvs_rtc 20cc034.snvs-rtc-lp: can't get snvs-rtc clk
snvs_rtc 20cc034.snvs-rtc-lp: rtc core: registered 1
i2c /dev entries driver
IR NEC protocol handler initialized
IR RC5(x) protocol handler initialized
IR RC6 protocol handler initialized
IR JVC protocol handler initialized
IR Sony protocol handler initialized
IR RC5 (streamzap) protocol handler initialized
IR SANYO protocol handler initialized
IR MCE Keyboard/mouse protocol handler initialized
mx6_v4l2_output v4l2_out.31: V4L2 device registered
mx6_v4l2_output v4l2_out.31: V4L2 device registered
mx6_v4l2_output v4l2_out.31: V4L2 device registered
imx2-wdt 20c0000.wdog: IMX2+ Watchdog Timer enabled.
sdhci: Secure Digital Host Controller Interface dri
sdhci: Copyright (c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
mmc1: no vqmmc regulator found
mmc1: no vqmmc regulator found
mmc1: SDHCI controller on 2194000.usdhc [2194000.us
mmc2: no vqmmc regulator found
mmc2: no vqmmc regulator found
mmc2: SDHCI controller on 2198000.usdhc [2198000.us
mmc3: no vqmmc regulator found
mmc3: no vqmmc regulator found
mmc3: SDHCI controller on 219c000.usdhc [219c000.us
galcore: clk_get vg clock failed, disable vg!
Galcore version 5.0.11.33433
```

过了 `jump OS` 后就无法下载内核和文件系统了。

所以使用 IMX6solo DDR 大小为 512M 时。就需要对 config.h 文件进行修改了。

```
1 /* Automatically generated - do not edit */
2 #define CONFIG_IMX_CONFIG board/freescale/imx/ddr/mx6solo_4x_mt41j128.cfg
3 #define CONFIG_MX6Q 1
4 #define CONFIG_DEFAULT_FDT_FILE "imx6dl-sabresd.dtb"
5 #define CONFIG_DDR_MB 512
6 #define CONFIG_SYS_USE_SPINOR 1
7 #define CONFIG_SYS_ARCH "arm"
8 #define CONFIG_SYS_CPU "armv7"
9 #define CONFIG_SYS_BOARD "mx6sabresd"
10 #define CONFIG_SYS_VENDOR "freescale"
11 #define CONFIG_SYS_SOC "mx6"
12 #define CONFIG_BOARDDIR board/freescale/mx6sabresd
13 #include <config_cmd_defaults.h>
14 #include <config_defaults.h>
15 #include <configs/mx6sabresd.h>
16 #include <asm/config.h>
17 #include <config_fallbacks.h>
18 #include <config_uncmd_spl.h>
```

这里用 imx6dl 的设备树，因为 dl 和 solo 是兼容的

这里一定要写成 CPU 的型号

DDR 大小 512M

开始 make 编译

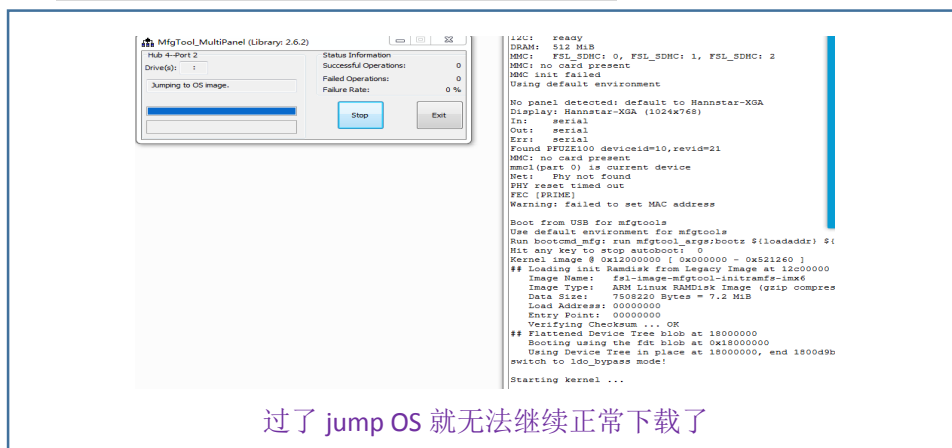
```
make[1]: *** No rule to make target 'board/freescale/imx/ddr/mx6solo_4x_mt41j128.cfg', needed by 'board/freescale/imx/ddr/mx6solo_4x_mt41j128.cfg.tmp'. Stop.
make: *** [u-boot.imx] Error 2
root@ubuntu:/home/xiang/yocto/fsl-release-bsp/imx6qsabresd-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/u-boot-imx/2014.04-r0/git#
```

编译报错。因为你最先 make 配置的是 IMX6Q 的 CPU 型号，现在你要用 IMX6solo，那么你就需要重新指定 make 配置型号。

make mx6solosabresd_config

这里一定要写成 mx6solosabresd_config 而不是 mx6qsabresd_config 或者 mx6dlsabresd_config

否则在下载程序的时候会出现和上面一样的情况



过了 jump OS 就无法继续正常下载了

执行 make mx6solosabresd_config 之后，linux 终端会回显

```
oot@ubuntu:/home/xiang/yocto/fsl-release-bsp/imx6qsabresd-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/u-boot-imx/2014.04-r0/git# make mx6solosabresd_config
configuring for mx6solosabresd - Board: mx6sabresd, Options: IMX_CONFIG=board/freescale/mx6sabresd/mx6solo_4x_mt41j128.cfg,MX6SOLO,DEFAULT_FDT_FILE=mx6dl-sabresd.dtb,DDR_MB=512,SYS_USE_SPINOR,SYS_NOSMP="nosmp"
oot@ubuntu:/home/xiang/yocto/fsl-release-bsp/imx6qsabresd-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/u-boot-imx/2014.04-r0/git#
```

然后你在 make

```
CFGS board/freescale/mx6sabresd/mx6solo_4x_mt41j128.cfgtmp
MKIMAGE u-boot.imx
OBJCOPY u-boot.srec
root@ubuntu:/home/xiang/yocto/fsl-release-bsp/imx6qsabresd-fb/tmp/work/imx6qsabresd-poky-linux-gnueabi/u-boot-imx/2014.04-r0/git# cp u-boot.imx /mnt/
```

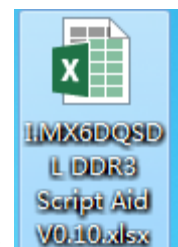
这样就对了

这样就是正常的 IMX6solo 单核，DDR 大小 512M 的配置，下载进开发板就可以了。

用 IMX6solo 单核对 ddr 512M 内存大小进行调试

我们现在将主板换成 IMX6solo 单核，DDR 还是美光的，只是将 1G 改成了

512M，我们用 DDR_Strem 软件对 512M 两片 128M*16DDR 进行压力测试。



1. 我们先打开 I.MX6DQSDL DDR3 Script Aid V0.10 excel 格式的脚本文件

TRAS Min (ns)	35	
System Information		
i.Mx Part	i.Mx6Q	Arik
Bus Width	64	
Density per chip select (Gb)	8	
Number of Chip Selects used	1	If only one CS is used, that must be CS0
Total DRAM Density (Gb)	8	
DRAM Clock Freq (MHz)	528	
DRAM Clock Cycle Time (ns)	1.894	
Address Mirror (for CS1)	Disable	
SI Configuration		
DRAM DSE Setting - DQ/DQM (ohm)	48	40ohm is used in FSL reference design.
Readme Register Configuration RealView .inc DS-5.ds Calibration +		

选择 regisert 配置

2. 修改 excel 里面的参数

针对 i.MX6X 的 MMDC 配置如下：

- Bus width=64(i.MX6S 仅支持 32 bit, i.MX6Q/D/DL 可以支持 64 或 32 bit)
- Number of Chip Selects used=1(我们的 SDP 设计使用了一个 CS0 来连接 4 片 DDR3)
- DRAM Clock Freq(Mhz)=528(i.MX6Q/D 工作在 528Mhz, i.MX6DL/S 工作在 400Mhz)
- DRAM Clock Cycle Time(ns)=1.894(i.MX6Q/D=1/528Mhz=1.894ns, i.MX6DL/S=1/400Mhz=2.5ns)

注意 i.MX6X 的 MMDC 仅支持工作在 528Mhz 或 400Mhz，所以虽然我们选择的 DDR3 可以工作在 800Mhz(DDR3-1600)，所以我们在选择 DDR3 的时序参考时，我们使用 DDR3-1600 的时间参数(单位 ns)，而不是时钟周期参数(clock)，因为时钟周期是按照 800MHz 来计算的，不正确。

Device Information	
Manufacturer:	Micron
Memory part number:	MT41K128M16JT-125
Memory type:	DDR3-1600
DRAM density (Gb)	2
DRAM Bus Width	16
Number of Banks	8
Number of ROW Addresses	14
Number of COLUMN Addresses	10
Page Size (K)	2
Self-Refresh Temperature (SRT)	Normal
tRCD=tRP=CL (ns)	13.75
tRC Min (ns)	48.75
tRAS Min (ns)	35
System Information	
i.Mx Part	i.Mx6Q
Bus Width	64
Density per chip select (Gb)	8
Number of Chip Selects used	1
Total DRAM Density (Gb)	8
DRAM Clock Freq (MHz)	528
DRAM Clock Cycle Time (ns)	1.894
Address Mirror (for CS1)	Disable
SI Configuration	
DRAM DSE Setting - DQ/DQM (ohm)	48
DRAM DSE Setting - ADDR/CMD/CTL (ohm)	48
DRAM DSE Setting - CK (ohm)	48
DRAM DSE Setting - DQS (ohm)	48
System ODT Setting (ohm)	60

改为 4Gb
因为 Gb 是
位，所以实
际内存大小
要除以8

改为 IMX6S

因为 IMX6S
只支持 32

因为 IMX6S 只
支持 32 位

修改后就是这样

Device Information	
Manufacturer:	Micron
Memory part number:	MT41K128M16JT-125
Memory type:	DDR3-1600
DRAM density (Gb)	4
DRAM Bus Width	16
Number of Banks	8
Number of ROW Addresses	14
Number of COLUMN Addresses	10
Page Size (K)	2
Self-Refresh Temperature (SRT)	Normal
tRCD=tRP=CL (ns)	13.75
tRC Min (ns)	48.75
tRAS Min (ns)	35
System Information	
i.Mx Part	i.Mx6S
Bus Width	32
Density per chip select (Gb)	4
Number of Chip Selects used	1
Total DRAM Density (Gb)	4
DRAM Clock Freq (MHz)	400
DRAM Clock Cycle Time (ns)	2.5
Address Mirror (for CS1)	Disable
SI Configuration	
DRAM DSE Setting - DQ/DQM (ohm)	48
DRAM DSE Setting - ADDR/CMD/CTL (ohm)	48
DRAM DSE Setting - CK (ohm)	48
DRAM DSE Setting - DQS (ohm)	48
System ODT Setting (ohm)	60

这样就修改完毕了

3.然后选择 RealView

```
1 //=====
2 //init script for i.Mx6S DDR3
3 //=====
4 // Revision History
5 // v01
6 //=====
7
8 wait = on
9 //=====
10 // Disable          WDOG
11 //=====
12 setmem /16          0x020bc000 = 0x30
13
14 //=====
15 // Enable all clocks (they are disabled by ROM code)
16 //=====
17 setmem /32          0x020c4068 = 0xffffffff
18 setmem /32          0x020c406c = 0xffffffff
19 setmem /32          0x020c4070 = 0xffffffff
20 setmem /32          0x020c4074 = 0xffffffff
21 setmem /32          0x020c4078 = 0xffffffff
22 setmem /32          0x020c407c = 0xffffffff
23 setmem /32          0x020c4080 = 0xffffffff
24 setmem /32          0x020c4084 = 0xffffffff
25
26 //=====
27 // IOMUX
28 //=====
29 //DDR IO TYPE:
30 setmem /32          0x020e0774 = 0x000C0000 // IOMUXC_SW_PAD_CTL_GRP_DDR_TYPE
31 setmem /32          0x020e0754 = 0x00000000 // IOMUXC_SW_PAD_CTL_GRP_DDRPKE
32
33 //CLOCK:
34 setmem /32          0x020e04ac = 0x00000028 // IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK_0
35 setmem /32          0x020e04b0 = 0x00000028 // IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK_1
36
37 //ADDRESS:
38 setmem /32          0x020e0464 = 0x00000028 // IOMUXC_SW_PAD_CTL_PAD_DRAM_CAS
39 setmem /32          0x020e0490 = 0x00000028 // IOMUXC_SW_PAD_CTL_PAD_DRAM_RAS
40 setmem /32          0x020e074c = 0x00000028 // IOMUXC_SW_PAD_CTL_GRP_ADDBS
41
42 //Control:
43 setmem /32          0x020e0494 = 0x00000028 // IOMUXC_SW_PAD_CTL_PAD_DRAM_RESET
44 setmem /32          0x020e04a0 = 0x00000000 // IOMUXC_SW_PAD_CTL_PAD_DRAM_SDRAS2 = DSE con
```

记住一定要屏蔽 dcd

选择 excel 脚本选项

// setmem /16 0x020bc000 = 0x30 //注掉此项，DCD 不可以访问此地址

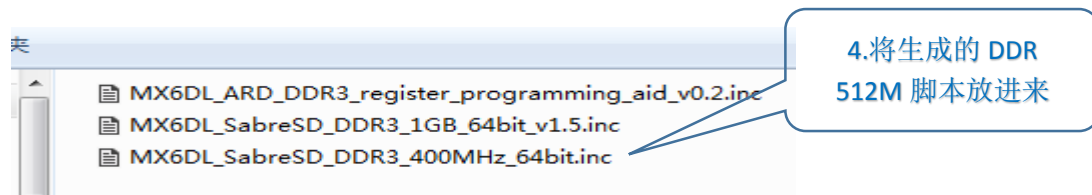
这个文件就是根据前面 register Configuration 选项自动设置的脚本文件。

然后我们就可以从 Realview.inc 页中获得 DDR3 的初始化设置，拷贝出来保存在一下 txt 文件中，然后改名为：“MX6QD_SabreSD_DDR3_528MHz_64bit.inc”。

这个名字随便改，只要保证后缀是.inc 就可以了。

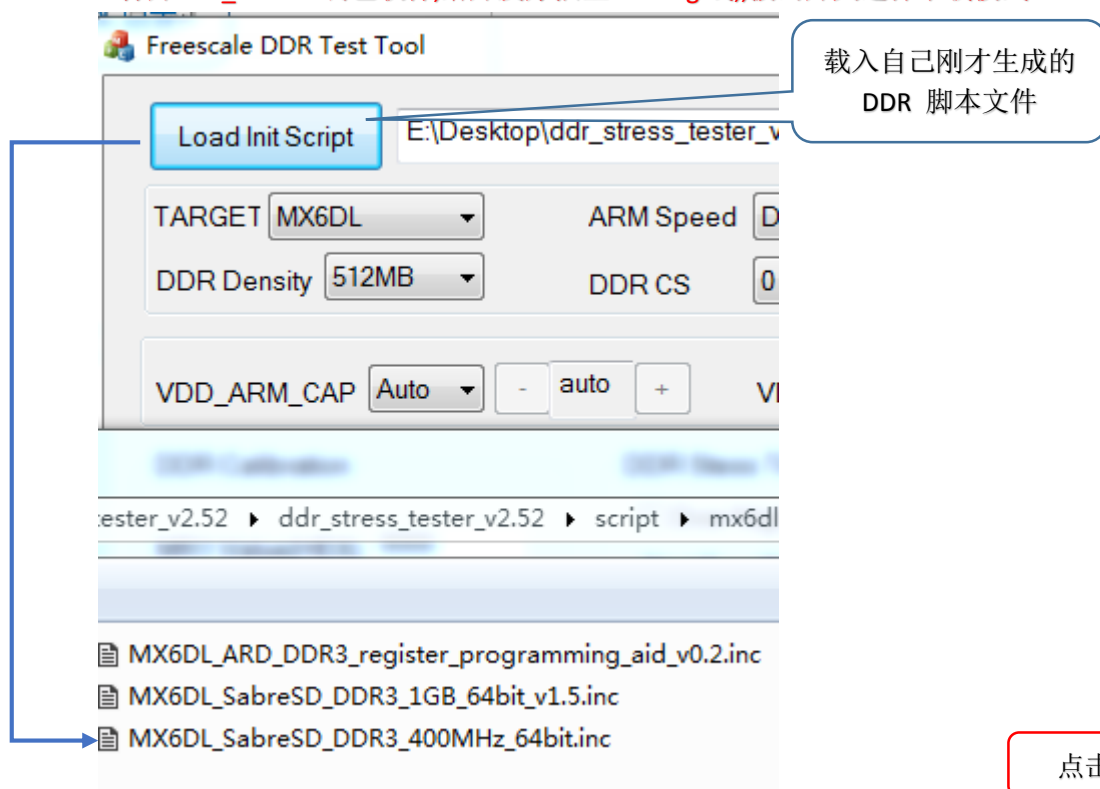
4.然后将这个文件放在 DDR_stream 文件的指定路径下



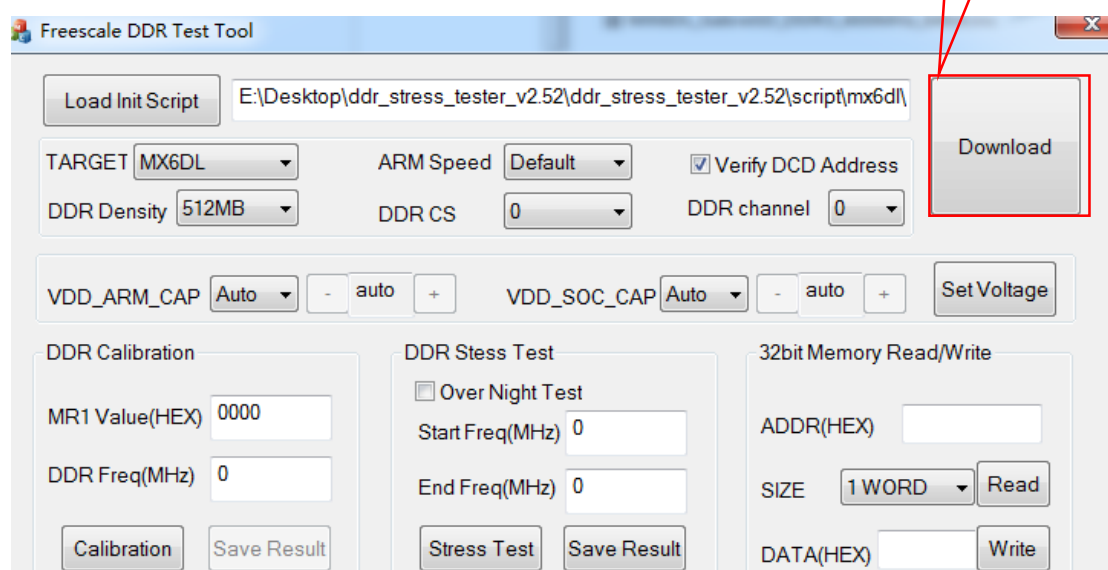


我把保存脚本的 txt 文档改成了 MX6DL_SabreSD_DDR3_400MHz_64bit.inc
其实这个脚本里面就是 DDR3 大小 512M 速度 400M ， 16 位宽总线的参数

5.打开 ddr_stream 调色软件,给开发办插上 USBotg 线,,拨码开关选择下载模式



选择 IMX6DL DDR 选择 512M 然后点击 Download



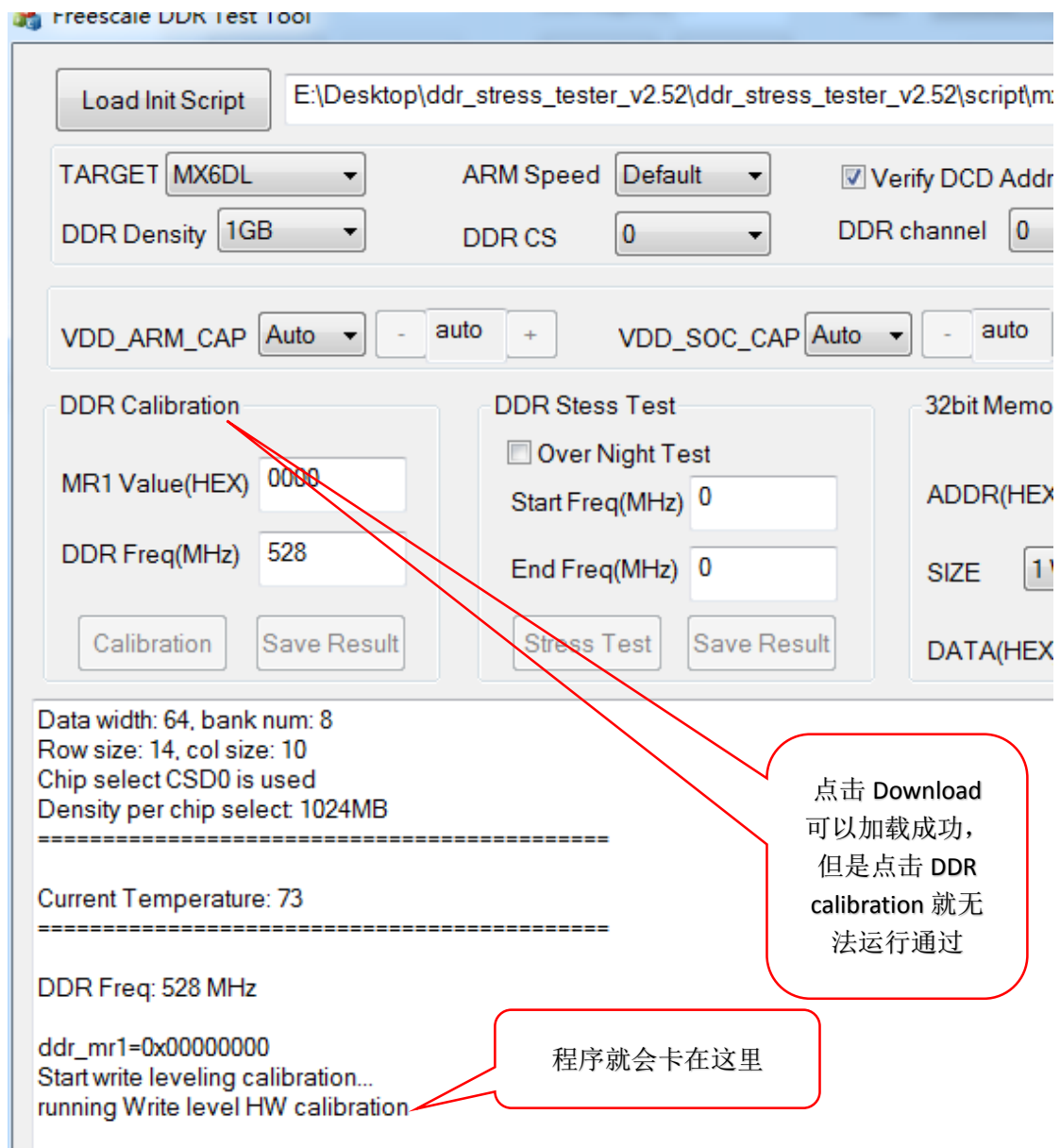
成功载入脚本就是下面这样

```
Boot Configuration
SRC_SBMR1(0x020d8004) = 0x00010030
SRC_SBMR2(0x020d801c) = 0x32000001
=====
```

ARM Clock set to 1GHz

```
=====
DDR configuration
BOOT_CFG3[5-4]: 0x00, Single DDR channel.
DDR type is DDR3
Data width: 32, bank num: 8
Row size: 14, col size: 10
Chip select CSD0 is used
Density per chip select: 512MB
```

如果你选择的主板和 DDR 型号或者数量和设置的脚本不一样



如果你选择的脚本和主板 CPU 的信号和 DDR 是匹配的

```
ddr_mr1=0x00000000
```

Start write leveling calibration...

running Write level HW calibration

Write leveling calibration completed, update the following registers in your initialization script

```
MMDC_MPWLDECTRL0 ch0 (0x021b080c) = 0x0043004A
```

```
MMDC_MPWLDECTRL1 ch0 (0x021b0810) = 0x00350037
```

Write DQS delay result

```
Write DQS0 delay: 74/256 CK
```

```
Write DQS1 delay: 67/256 CK
```

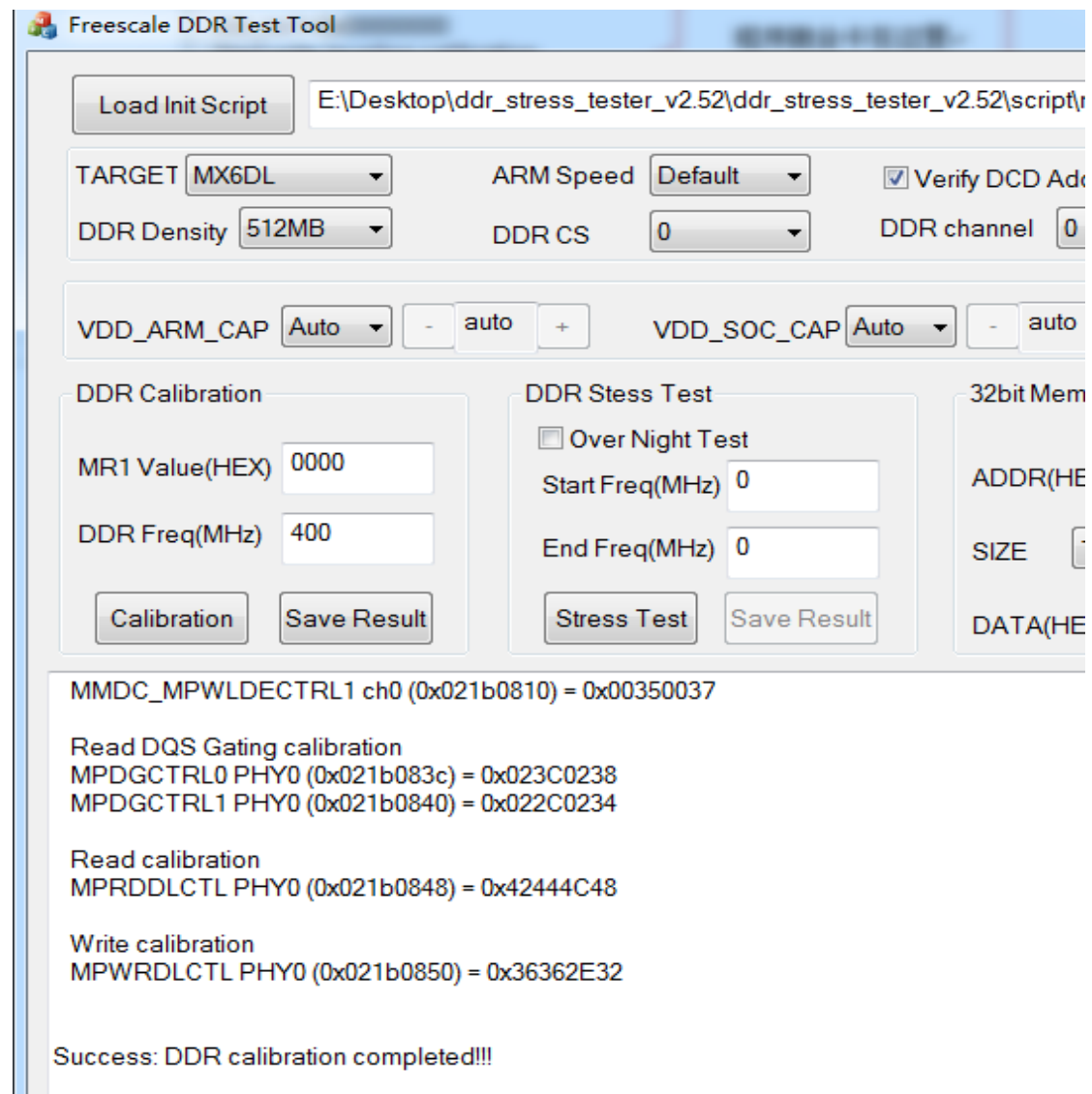
```
Write DQS2 delay: 55/256 CK
```

```
Write DQS3 delay: 53/256 CK
```

Starting DQS gating calibration

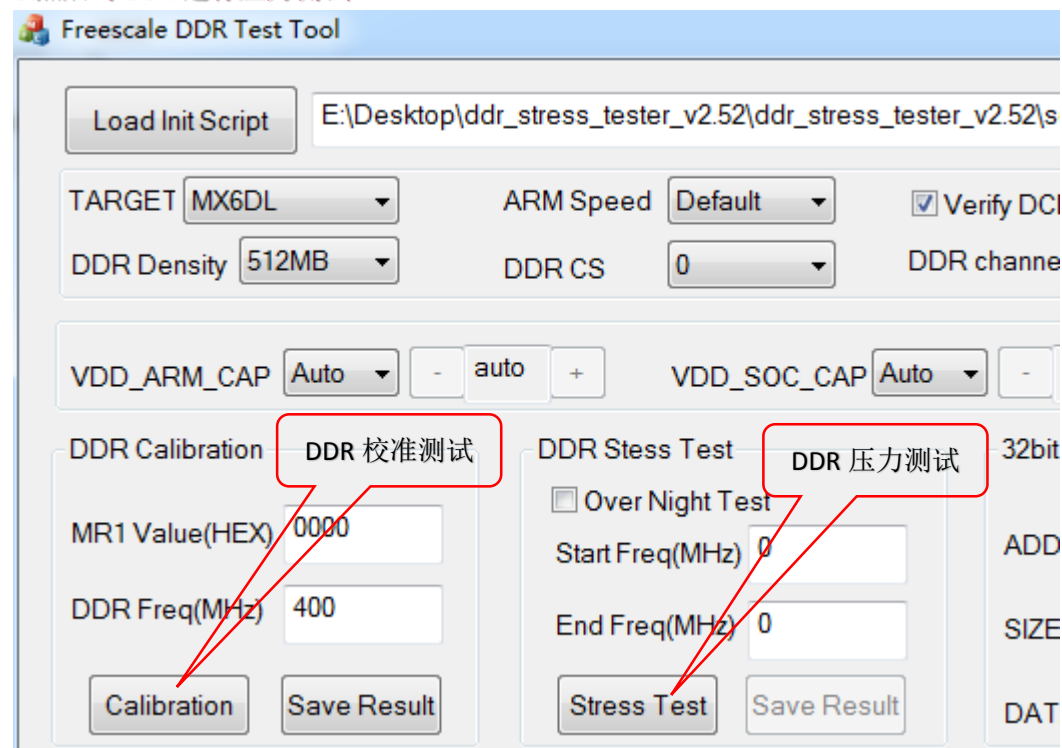
.....

在进行 DDR calibration 也就是 DDR 校准的时候，会成功执行到 Starting DQS 信号测试
等待一个不太长的时间



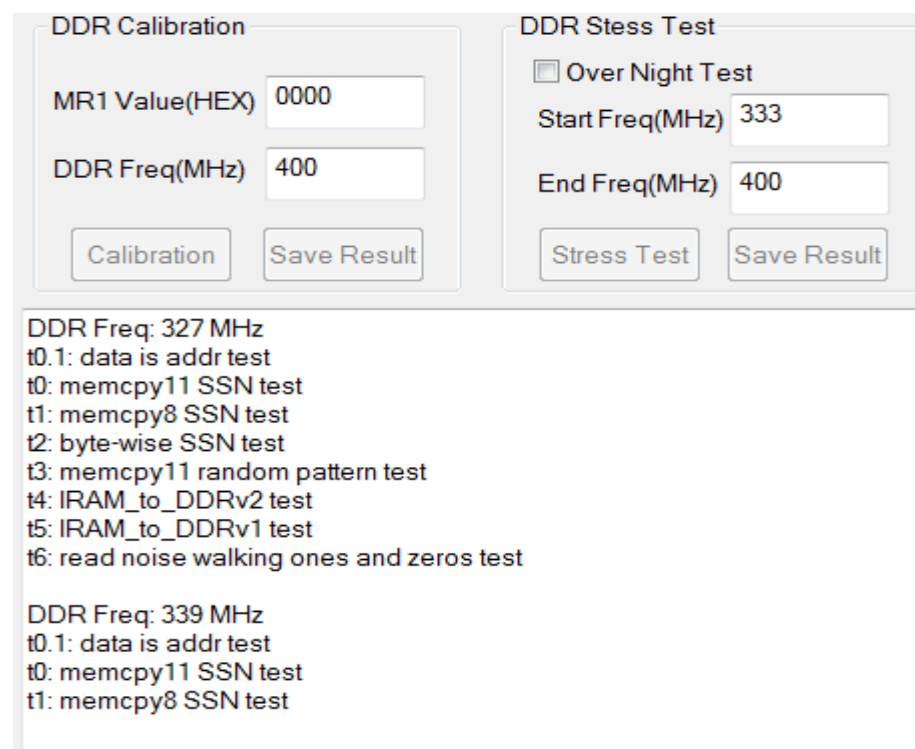
DDR 成功校准

6. 然后对 DDR 进行压力测试



DDR 压力测试频率标准

束频率。此频率必须在 135 MHz 至 672 MHz 的范围内。由于 $t_{CK} (AVG)_{MIN} = 3 \text{ ns}$ 限制了 DLL 何时启用且“Aid”的脚本仅支持 DLL 启用,因此 DDR3 的工作频点设置应高于 333 MHz。如



这就是 DDR 压力测试

DDR Calibration

MR1 Value(HEX)

0000

DDR Freq(MHz)

400

Calibration

Save Result

DDR Stress Test

☐ Over Night Test

Start Freq(MHz)

333

End Freq(MHz)

400

Stress Test

Save Result

t6: read noise walking ones and zeros test

DDR Freq: 396 MHz
t0.1: data is addr test
t0: memcpy11 SSN test
t1: memcpy8 SSN test
t2: byte-wise SSN test
t3: memcpy11 random pattern test
t4: IRAM_to_DDRv2 test
t5: IRAM_to_DDRv1 test
t6: read noise walking ones and zeros test

Success: DDR Stress test completed!!!

DDR 3 内存 333M~400M 速率压力测试成功。

如果我重启主板，先进行压力测试可以吗？

DDR Calibration

MR1 Value(HEX)

0000

DDR Freq(MHz)

400

Calibration

Save Result

DDR Stress Test

☐ Over Night Test

Start Freq(MHz)

333

End Freq(MHz)

400

Stress Test

Save Result

```

=====
DDR Stress Test Iteration 1
Current Temperature: 76
=====

DDR Freq: 327 MHz
t0.1: data is addr test
Address of failure: 0x10000080
Data was: 0x10000000
But pattern should match address
Error: failed to run stress test!!!

```

这就是重启主板后先 Download 脚本，然后不经过 DDR 校准，直接进入 DDR 压力测试，发送错误。所以必须先进行 DDR 校准，然后才能进行 DDR 压力测试，顺序不能乱。