S32G How to Develop the QSPI-Nor Lauterbach Script

by John.Li NXA08200

本文说明如何使用和开发 S32G QSPI NOR lauterbach 脚本。

版本	历史	作者
V1	● 创建本文	JohnLi

目录

1 背景和参考资料 2
1.1 背景说明 2
1.2 参考资料 2
2 高速读开发流程 3
2.1 时钟相关修改 5
2.2 Lut配置说明 6
2.3 QSPI NOR控制器配置 12
2.4 QuadSPI_Write32BytesDOPI读函数分析 15
2.5 增加AHB read寄存器配置 17
2.6 测试结果 18
3 高速写开发流程 19
3.1 Erase lut分析及调用 19
3.2 Write lut分析及调用 21
3.3 测试结果 22
3.4 Lauterbach烧写镜像脚本说明 22

1 背景和参考资料

1.1 背景说明

当需要快速验证外部QSPI NOR时，使用lauterbach脚本驱动是一个比较迅速的办法，默认的脚本仅支持低速模式，本文说明如何修改脚本成一般产品比较常见的高速模式。

本文说明了如何验证读与写，至于使用lauterbach来烧写镜像，是除了flash tool外的另一种烧写办法，可以避开UART硬件相关问题及flash tool算法镜像的开发，这样的脚本与算法需要lauterbach发布。

此文做为以下参考资料中关于QSPI NOR开发系列文档的一个补充。

1.2 参考资料

本文基于 S32G2 RDB2 板开发：

分类	名称	类别	说明
文档	S32G2.pdf S32G3.pdf	S32G 芯片手册	从www.nxp.com/s32g下载
Mcal	SW32G_RTD_4.4_3.0.2	BSP 手册	从www.nxp.com个人帐号下载
Boot loader	Integration_Reference _Examples_S32G2_2022_06	BSP 手册	从www.nxp.com个人帐号下载
工具	S32Design Studio	S32DS	从www.nxp.com个人帐号下载 参考其中的 QuadSPI 配置工具。
文档	AN13563: S32G QuadSPI Deep Dive Application note	AppNotes	从www.nxp.com/s32g下载 关于 AutoSar MCAL 驱动的详情，参考此文档。
文档	AN12808: Quad SPI (QSPI) Timing Configuration on the S32G2 Vehicle Network Processor Application Note	AppNotes	从www.nxp.com/s32g下载 高速模式下的寄存器配置请参考此文档。
文档	S32G_QSPINOR_定制_*.pdf	AppNotes	从nxp community 下载 https://community.nxp.com/t5/ NXP-Designs-Knowledge-Base/

			S32G-QSPI-Nor-customization-doc/ta-p/1399906 关于 flash timing header 配置， flash tools SDK 工程定制(用于开发 flash tools 的 qspi nor binary) uboot 定制，内核驱动定制请参考此文档。
文档	MX25UW51245G.pdf	QSPI NOR 数据手册	

2 高速读开发流程

首先需要通用的 M7 启动连接脚本：

假如 M7 已经运行，只需要 attach，则运行脚本：

```
sys.cpu S32G-M7_0
SYStem.config.debugporttype JTAG
SYStem.Option TRST OFF
sys.attach
break
list
ENDDO
```

既可。可以增加运行镜像 symbol 加载语句。

假如 M7 处于 parition/CPU reset 状态，比如说下载模式下处于 HSE M7 运行状态，A53 Linux 启动处于 A53 运行状态，则需要比如说以下脚本：C:\NXP\Integration_Reference_Examples_S32G2_2022_06\code\framework\realtime\swc\bootloader\platforms\S32G2XX\build\cmm\s32gxx_m7.cmm，修改为：

```
;DO QuadSPI_Flash_Init.cmm
DO device.cmm
…
;Data.Load.Elf ../bin_bootloader/Bootloader.elf  /GLOBTYPES /AnySym

;Release the core from the loop from startup
;data.set eaxi:CATCH_CORE_RESET 0x00000000
```

;list

;go main

- GOSUB InitSramEccViaRAMCtrl //初始化 SRAM 及 ECC。
- GOSUB EnableCM7_0 //启动 M7_0，将 M7_0 所在的 parititon 0 及核，启动，时钟打开，退出 reset。
- GOSUB DisableSWTWatchdog //闭关默认打开 watchdog，避免重启。
- DO device.cmm //运行 QSPI NOR 脚本。

然后再开发 QSPI NOR 相关脚本：

所有的 mcal sample 都有一个 device.cmm：C:\NXP\SW32G_RTD_4.4_3.0.2\eclipse\plugins\Fls_TS_T40D11M30I2R0\examples\EBT\Fls_Example_S32G274A_M7\debug\device.cmm。

默认的脚本只支持 QSPI NOR 的低速访问，本文说明如何支持 200Mhz，DDR 模式， 8bit，External DQS，DLL Auto Update Mode。从而模拟出最高读性能的情况。

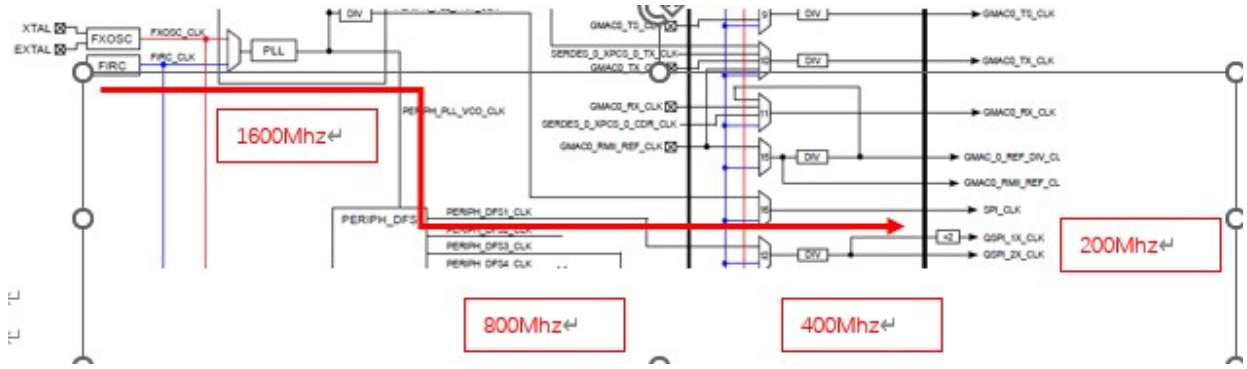
删除掉不相关部分，只保留以下代码及其调用函数：

```
LOCAL &QSPI_BASE
LOCAL &QSPI_Cntl_BASE
&QSPI_BASE=0x00000000
&QSPI_Cntl_BASE=0x40134000
; Clock configurations
; PLL : 1600Mhz
GOSUB PERIPH_PLL_1600MHZ          //考虑时钟源来自于 FIRC=48Mhz，配置外设 PLL 为 1600Mhz，开发此函数
;DFS_800
GOSUB PERIPH_PLL_DFS1_800MHZ // DFS1 源自于 PERIPH_PLL，配置打开，并除 2 得 800Mhz，开发此函数
GOSUB QuadSPI_PinMux_CLKEnable //脚本原有函数，配置管脚 IOMUX 需要增加 MC_CGM0_AC12 来除 2，得到 QSPI_X2 时钟为 400Mhz
; GOSUB QuadSPI_Init //脚本原有函数，用于初始化 LUT，注释掉后将 LUT 初始化修改到以下函数
;GOSUB QuadSPI_InitDOPI_BypassMode_66MHz
GOSUB QuadSPI_InitDOPI_DLL_AutoUpdateMode_100MHz //脚本原有函数，为主要修改函数，用于配置 DDR 高速模式，并将初始化 LUT 拷贝到此处
GOSUB QuadSPI_Read32BytesDOPI
```

ENDDO

2.1 时钟相关修改

配置 QSPI_NOR 时钟源为下图：
FIRC(48MHZ)->PERIPH_PLL(1600Mhz)->PERIPH_DFS1(800Mhz)->MC_CGM0_DIV12=QSPI_2X_CLK(400Mhz)->QSPI_1X_CLK(200Mhz)。



- GOSUB PERIPH_PLL_1600MHZ

PERIPH_PLL_1600MHZ:

Data.Set 0x4003C080 %Long 0x00000000 ; DIV0 disabled

…

Data.Set 0x4003C09C %Long 0x00000000 ; DIV7 disable

Data.Set 0x4003C000 %Long 0x80000000 ; disable PLL

Data.Set 0x4003C020 %Long 0x00000000 ; Select FIRC of 48MHz as source

Data.Set 0x4003C008 %Long 0x0C3F1021 ; PLLDV - ref = 48MHz FIRC, RDIV = 1, MFI = 33 VCO = 1600

Data.Set 0x4003C010 %Long 0x40001800 ; PLLFD - SDMEN = 1, MFN = 6144 fractional mode

— When PLLDV[RDIV] is not 0:

$$f_{pll\_VCO} = \frac{f_{pll\_ref}}{PLLDV[RDIV]} \times \left(PLLDV[MFI] + \frac{PLLFD[MFN]}{18432}\right)$$

Equation 4. PLL VCO frequency in Fractional mode when PLLDV[RDIV] is not 0

PERIPH_PLL=(48/1)X(33+(6144/18432))=1600Mhz

WAIT 100.ms ; Wait for the PLL reference clock to be stable

Data.Set 0x4003C000 %Long 0x00000000 ; enable PLL

WAIT 100.ms ; Wait for the PLL to lock

&temp=Data.Long(A:0x4003C004) ;Read status to make sure that PLL is locked. Expected output = 0x00000004

RETURN

- GOSUB PERIPH_PLL_DFS1_800MHZ

```
PERIPH_PLL_DFS1_800MHZ:
  Data.Set 0x40058014 %Long 0x0000003F ; DFS outputs are in reset
  WAIT 100.ms ; Wait for the DFS port to become 0
  &temp=Data.Long(A:0x4005800C) ;Read status to make sure that DFSs are reset Expected output = 0x00000000
  Data.Set 0x40058018 %Long 0x00000002 ; DFS module is disabled
  Data.Set 0x4005801C %Long 0x00000100 ; for DFS1 DVPORT0 = 1 i.e.
PERIPH_DFS1_CLK=PERIPH_DLL(1600)/(2XMFI=1)=800Mhz
  Data.Set 0x40058018 %Long 0x00000000 ; DFS module is enabled
  Data.Set 0x40058014 %Long 0x0000003E ; DFS1 output is enabled
  WAIT 100.ms ; Wait for the DFS port to become 0
  &temp=Data.Long(A:0x4005800C) ;Read status to make sure that DFS1 is enabled Expected output = 0x00000001
  ;Switch clock from FIRC to PLL(PERIPH_DFS1_CLK) in MC_CGM for QSPI_1X_CLK
  ;See QuadSPI_PinMux_CLKEnable function
RETURN
```

- GOSUB QuadSPI_PinMux_CLKEnable

  增加如下：

```
  QuadSPI_PinMux_CLKEnable:
    ;clk enable for the QuadSPI
    Data.Set SD:40030608 %LE %Long  0x00000000  ;MC_CGM_0_AC12_DC_0 Divider disabled
    Data.Set SD:40030600 %LE %Long  0x1A000004  ;MC_CGM_0_AC12_CSC switch to
PERIPH_DFS1_CLK=800MHz)
    WAIT 100.ms ;Wait for clock switch to complete
    &temp=Data.Long(A:0x40030604) ;Read status to make sure the successful switch Expected output =
0x1A020000
    Data.Set SD:40030608 %LE %Long  0x80010000  ;MC_CGM_0_AC12_DC_0 divide by 2 (i.e. for PLL
QSPI_2X_CLK = 80MHz, QSPI_1X_CLK = 40MHz) if DFS1=160Mhz;*/
    ;QSPI_2X_CLK = 400MHz, QSPI_1X_CLK = 200MHz  if  DFS1=800) */
  …
```

2.2 Lut 配置说明

Lut 示意图如下：

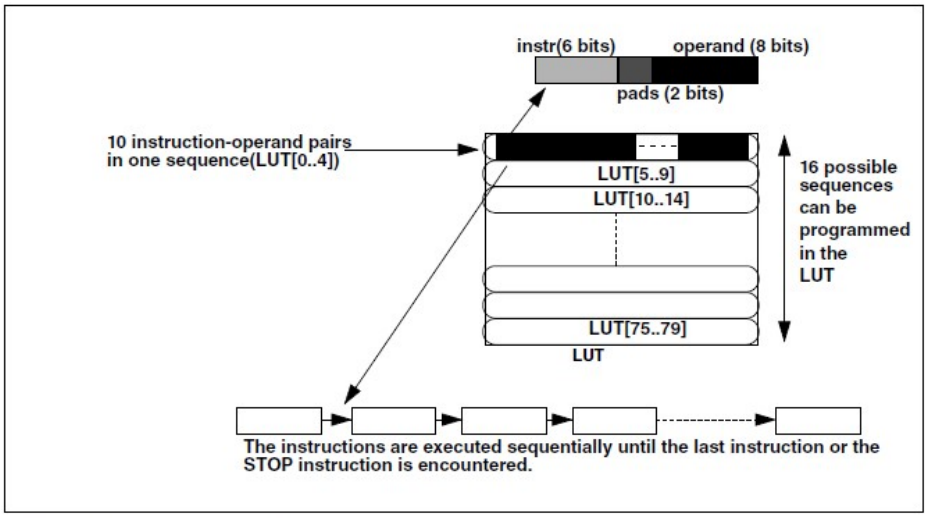


Figure 215. LUT and sequence structure

每个 LUT 寄存器，由一对指令操作对组成：

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
Reset	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INSTR0						PAD0		OPRND0							
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1

五个 LUT 寄存器配置，组成一组 LUT 序列 Seq，所以 Seq ID 从：

314	LUT Register (LUT1)	32	RW	2400_1C08
318 - 44C	LUT Register (LUT2 - LUT79)	32	RW	0000_0000

Align 为 5X4。

其中指令表如下：

Instruction	Instruction encoding	Pins	Operand	Action on serial flash memories
CMD	1d	N={0,1,2,3}d 0d - One pad 1d - Two pads 2d - Four pads	8-bit command value	Provides the serial flash memory with the SFM command operand (Encoded) on the number of pads specified in STR mode.
ADDR	2d		Number of address bits	Provide the serial flash memory with address cycles according to the operand on the number of pads specified

WRITE	8		Write data size in bytes	Write data on the number of pads specified The data size can be overwritten by writing to the IDATSZ field of IP Configuration Register (IPCR).
WRITE_DDR	15d		Write data size in bytes	Write data on the number of pads specified at each clock edge of serial flash memory The data size can be overwritten by writing to the IDATSZ field of IP Configuration Register (IPCR).
CMD_DDR	17d	N={0,1,2,3}d 0d - One pad 1d - Two pads	8-bit command value	Provides the serial flash memory with the SFM command operand (Encoded) on the number of pads specified in DTR mode..
ADDR_DDR	10d	N={0,1,2,3}d 0d - One pad 1d - Two pads 2d - Four pads 3d- Eight pads	Number of address bits to be sent (for example, 24d => 24 address bits required)	Provide the serial flash memory with address cycles according to the operand on the number of pads specified at each clock edge of serial flash memory clock. The actual address to be provided is derived from the incoming address in case of AHB-initiated transactions and the value of SFAR in case of IPS-initiated transactions if SFACR[CAS] is set to 0. Otherwise, the actual address takes CAS into consideration.
DUMMY	3d		Number of dummy clock cycles (should be <= 64 and > 2 cycles)	Provide the serial flash memory with dummy cycles according the operand The PAD information defines the number of pads in input mode. For example, one pad implies that pad 1 is not driven, rest all are driven. NOTE If DLL is enabled and N dummy cycles are needed, you must program two back-to-back DUMMY instructions: DUMMY: N-2 followed by DUMMY:2.

READ_DDR	14d	N={0,1,2,3}d 0d - One pad 1d - Two pads 2d - Four pads 3d - Eight pads	Read data size in bytes (for AHB transactions, the application should ensure that data size is in multiple of 8 bytes)	Read data from flash memory on the number of pads specified at each clock edge of serial flash memory. The data size might be overwritten by writing to the ADATSZ field of the BUFxCR registers for AHB-initiated transactions and IDATSZ field of IP Configuration Register (IPCR) for IP initiated transactions.

Instruction	Instruction encoding	Pins	Operand	Action on serial flash memories
STOP[3]	0d	NA	NA	Stop execution; deassert CS

主要用到以下三个 LUT：

- 配置 QSPI NOR 为可写：

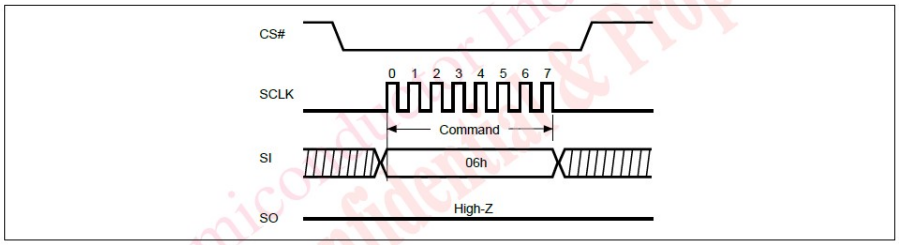
10-1. Write Enable (WREN)

The Write Enable (WREN) instruction is for setting Write Enable Latch (WEL) bit. For those instructions like PP/ PP3B/PP4B, SE/SE3B/SE4B, BE/BE3B/BE4B, CE, WRSR, WRCR2, SBL, WRFBR, ESFBR, WRSCUR, WRLR, WSPB and ESSPB which are intended to change the device content WEL bit should be set every time after the WREN instruction setting the WEL bit. WREN is is also required before initiation of write-to-buffer sequence (WRBI command).

The sequence of issuing WREN instruction is: CS# goes low→sending WREN instruction code→ CS# goes high.

WREN instruction is not allowed while the device is operating in PE mode or Write Register Mode.

Figure 5. Write Enable (WREN) Sequence (SPI Mode)



WREN	06h	•	•

```
;Program LUT10 with WRITE_ENABLE
Data.Set A:&QSPI_Cntl_BASE+0x338 %LE %Long 0x00000406   ;       SEQID 2
Data.Set A:&QSPI_Cntl_BASE+0x33C %LE %Long 0x0
```

	Instr(6 bits)	Pads(8 bits)	Operand(8 bits). 查 QSPI 数据手册命令字一节，或参考 S32G 指令集表参数。
0406	0x01(CMD)	0x0(1 bit)	0x06(WREN)

0000	0x0(STOP)	0x0(1 bit)	0x0(STOP)

调用如下：

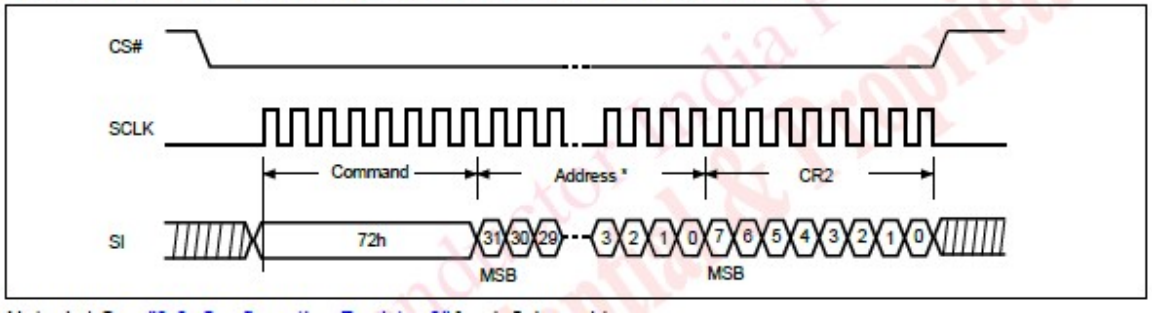
```
; write sequence ID and assert WriteEnable id command
Data.Set A:&QSPI_Cntl_BASE+0x08 %Long  (2.<<24.)  ; sequence
wait 100.ms
```

- 配置 QSPI NOR 为 DOPI 模式：

```
;Program LUT60 Write CONFIG2 REGISTER - SPI mode
Data.Set A:&QSPI_Cntl_BASE+0x400 %LE %Long 0x08200472   ;          SEQID 12
Data.Set A:&QSPI_Cntl_BASE+0x404 %LE %Long 0x00002001
```

默认是在 SPI 模式下，配置 QSPI NOR：

Figure 31. Write Configuration Register 2 (WRCR2) Sequence (SPI Mode)



为：0x2。

The WRCR2 instruction is for changing the values of Configuration Register 2. Before sending WRCR2 instruction,the Write Enable (WREN) instruction must be decoded and executed to set the Write Enable Latch (WEL) bit in advance.

9-3. Configuration Register 2

Address	Bit	Name	Description	Default
000h	Bit 0	SOPI (STR OPI Enable)	0= STR OPI disable 1= STR OPI enable	0
	Bit 1	DOPI(DTR OPI Enable)	0= DTR OPI disable 1= DTR OPI enable	0

	Instr(6 bits)	Pads(8 bits)	Operand(8 bits). 查 QSPI 数据手册命令字一节，或参考 S32G 指令集表参数。
0472	0x01(CMD)	0x0(1 bit)	0x72(WRCR2)
0820	0x02(ADDR)	0x0(1 bit)	0x20(32 Addr bits to be sent on 1 pad)
2001	0x8(WRITE)	0x0(1 bit)	0x01 write data size in byte
0000	0x0(STOP)	0x0(1 bit)	0x0(STOP)

然后值为：

QuadSPI_InitDOPI_DLL_AutoUpdateMode_100MHz:

…

; We assume we are after a reset, in SPI mode 1X SDR

Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0x00000002 //TX Buffer Data Regsiter=2

; Program LUT60 Write CONFIG2 REGISTER - SPI mode with value to switch to DOPI mode. From this point on, all LUT seqs should be DDR OPI mode compatible

Data.Set A:&QSPI_Cntl_BASE+0x08 %Long (12.<<24.) ; sequence

- 使用 DOPI 模式 READ_8DTRD

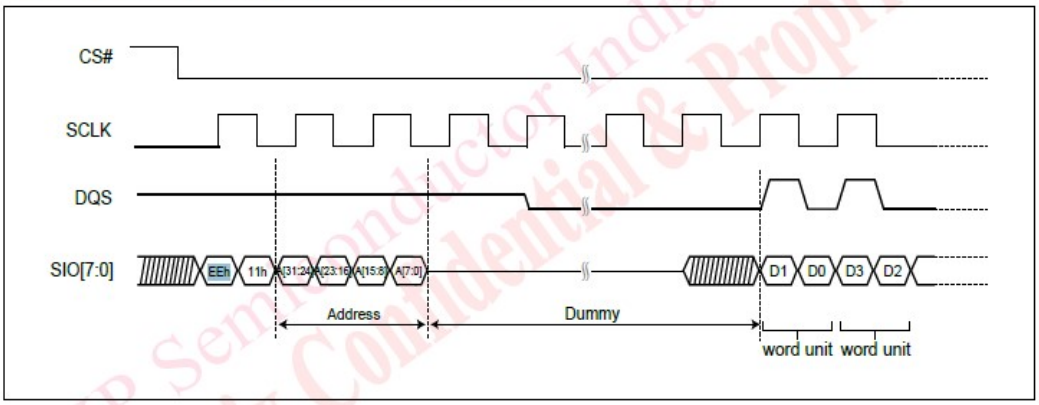
;Program LUT35 with 8DTRD - READ DOPI mode

Data.Set A:&QSPI_Cntl_BASE+0x39C %LE %Long 0x471147EE ; SEQID 7

Data.Set A:&QSPI_Cntl_BASE+0x3A0 %LE %Long 0x0C142B20

Data.Set A:&QSPI_Cntl_BASE+0x3A4 %LE %Long 0x00003B01

Figure 43. OCTA Read Mode Sequence (DTR-OPI Mode)



	Instr(6 bits)	Pads(8 bits)	Operand(8 bits). 查 QSPI 数据手册命令字一节，或参考 S32G 指令集表参数。
47ee	0x17(CMD_DDR)	0x3(8 bit)	0xee 0x11(Octa I/O DTR read)
4711	0x17(CMD_DDR)	0x3(8 bit)	
2b20	0x10(ADDR_DDR)	0x3(8 bit)	0x20(32 Addr bits to be sent on 4 pad)
0f14	0x3(DUMMY)	0x3(8 bit)	0x14(20 dummy cycles)
3b10	0x14(READ_DDR)	0x3(8 bit)	0x10(Read 16 Bytes on 4 pad)
0000	0x0(STOP)	0x0(1 bit)	0x0(STOP)

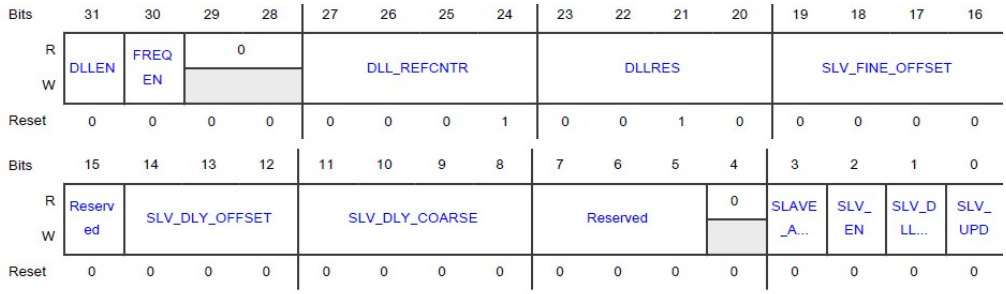
调用如下：

; write sequence ID and assert Read command

```
Data.Set A:&QSPI_Cntl_BASE+0x08 %Long  ((7.<<24.)+32.) ; sequence 7 + 32 bytes to be read
```

2.3 QSPI NOR 控制器配置

原函数 QuadSPI_InitDOPI_DLL_AutoUpdateMode_100MHz，为 100Mhz 低速，DDR 模式，需要修改为 200Mhz，DDR，auto update 模式，关键在于 DLLCRA 寄存器的配置：



Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLLEN	FREQEN	0		DLL_REFCNTR				DLLRES				SLV_FINE_OFFSET			
W																
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	SLV_DLY_OFFSET			SLV_DLY_COARSE				Reserved			0	SLAVE_A...	SLV_EN	SLV_DLL...	SLV_UPD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 DLLEN	DLL enable 0 - DLL reference logic remains in reset and should be 0 for at least three flash memory clock cycles for reset. 1 - Enables DLL logic. Set it to 1 after all the configuration for DLLCR reference settings is complete.
30 FREQEN	Frequency enable 0 - Selects delay chain for low frequency of operation 1 - Selects delay chain for high frequency of operation
29-28 —	Reserved
27-24 DLL_REFCNTR	DLL reference counter Select the "n+1" interval of DLL phase detection and reference delay updating interval (minimum recommended value = 1).
23-20 DLLRES	DLL resolution

	Minimum resolution for DLL phase detector to remain locked/unlocked based on flash memory clock jitter. The minimum value is 2, and should be programmed to a more suitable value, such as 6.
19-16 SLV_FINE_OFFSET	Fine offset delay elements in incoming DQS This field sets the number of fine offset delay elements up to 16 in incoming DQS, and the default must be 1 element.
15 —	Reserved
14-12 SLV_DLY_OFFSET	T/16 offset delay elements in incoming DQS This field sets the number of T/16 offset delay elements in incoming DQS; default is 0.
11-8 SLV_DLY_COARSE	Delay elements in each delay tap This field sets the number of delay elements in each delay tap. The field is used to overwrite DLL-generated delay values and works when the value of SLV_DLL_BYPASS is 1. Note : Please refer to the QuadSPI datasheet for more details.
7-5 —	Reserved
4 —	Reserved
3 SLAVE_AUTO_UPDT	Slave chain update This field automatically updates the slave chain as soon as DLL is locked. 0 - Auto-update feature is disabled. 1 - Auto-update feature is enabled.
2 SLV_EN	Slave enable 0 - DLL slave logic remains in reset, and its value should be 0 for at least three flash memory clock cycles for reset. 1 - Enables DQS slave delay chain, and should be 1 before any slave configuration settings take place.
1 SLV_DLL_BYPASS	Slave DLL bypass This field enables selection of the number of delays in each slave delay tap. 0 - Disables manual selection of coarse delays in the slave delay chain. 1 - Enables selection of number of delays in each slave delay tap, based on DLLCRA[SLV_DLY_COARSE].
0 SLV_UPD	Slave update You must program this field only after slave delay chain configuration takes place.

Field	Description
	0 - Disables any further update on DQS slave delay chain. 1 - Updates the DQS slave delay chain with either ref-delay or bypass slave delay value, and should be set in the absence of the DQS clock.

参考 MCAL 代码的配置方法，设计以下对高速模式，auto update 的配置流程:

	Qspi_ip_controller.c: Qspi_Ip_ConfigureDLLA:	Device.cmm : QuadSPI_InitDOPI_DLL _AutoUpdateMode_100MHz :
1	/* Ensure DLL and slave chain update are off */	Data.Set A:&QSPI_Cntl_BASE+0x60 %LE %Long

	Qspi_Ip_DLLSlaveUpdateA(baseAddr, FALSE); Qspi_Ip_DLLEnableA(baseAddr, FALSE);	0x02800000 ;SLV_UPD=0, DDLEN=0
2	/* Enable DQS slave delay chain before any settings take place */ Qspi_Ip_DLLSlaveEnA(baseAddr, TRUE);	Data.Set A:&QSPI_Cntl_BASE+0x60 %LE %Long 0x02800004 ;+SLV_EN=1, SLV_DLL_BYPASS=0
3	Qspi_Ip_ConfigureDLLAUpdate |-> /* Set DLL in auto update mode and configure coarse and fine delays */ Qspi_Ip_DLLSlaveBypassA(baseAddr, FALSE);	Data.Set A:&QSPI_Cntl_BASE+0x60 %LE %Long 0x02800004 ;+SLV_EN=1, SLV_DLL_BYPASS=0
4	Qspi_Ip_ConfigureDLLAUpdate |-> Qspi_Ip_DLLSlaveAutoUpdateA(baseAddr, (QSPI_IP_DLL_AUTO_UPDATE == userConfigPtr->dllSettingsA.dllMode));	Data.Set A:&QSPI_Cntl_BASE+0x60 %LE %Long 0x0280000c ;+SLAVE_AUTO_UPDT=1
5	Qspi_Ip_ConfigureDLLAUpdate |-> Qspi_Ip_DLLFreqEnA(baseAddr, userConfigPtr->dllSettingsA.freqEnable);	Data.Set A:&QSPI_Cntl_BASE+0x60 %LE %Long 0x4280000c ;+FREQEN=1
6	Qspi_Ip_ConfigureDLLAUpdate |-> Qspi_Ip_DLLSlaveUpdateA(baseAddr, TRUE);	Data.Set A:&QSPI_Cntl_BASE+0x60 %LE %Long 0x4280000d ;+SLV_UPD=1.
7	Qspi_Ip_ConfigureDLLAUpdate |-> /* Enable DLL */ Qspi_Ip_DLLEnableA(baseAddr, TRUE);	Data.Set A:&QSPI_Cntl_BASE+0x60 %LE %Long 0xc280000d ;+DDLEN==1. wait 100.ms
8	Qspi_Ip_ConfigureDLLAUpdate |-> /* Disable slave chain update */ Qspi_Ip_DLLSlaveUpdateA(baseAddr, FALSE);	Data.Set A:&QSPI_Cntl_BASE+0x60 %LE %Long 0xc280000c ;+SLV_UPD=0.

之后，按照 S32DS 的 flash timing header 的设置，补齐以下寄存器配置：

```
Data.Set A:&QSPI_Cntl_BASE+0x130 %LE %Long 0x40ff40ff ; QuadSPI0->DLCR;
```

Data.Set A:&QSPI_Cntl_BASE+0x190 %LE %Long 0xaa553443 ; QuadSPI0->DLPR;

; mx25_sim200ddr.bin

QuadSPI Registers

Register Name	Value	Description
MCR	0x30f00cc	Module Configuration Register
FLSHCR	0x10303	Flash Memory Configuration Register
BFGENCR	0x0	Buffer Generic Configuration Register
DLLCRA	0xc280000c	DLL Flash Memory A Configuration Register
PARITYCR	0x0	Parity Configuration Register
SFACR	0x20000	Serial Flash Memory Address Configuratio...
SMPR	0x44000000	Sampling Register
DLCR	0x40ff40ff	Data Learning Configuration Register
SFA1AD	0x20000000	Serial Flash Memory A1 Top Address Regis..
SFA2AD	0x20000000	Serial Flash Memory A2 Top Address Regis..
DLPR	0xaa553443	Data Learn Pattern Register
SFAR	0x0	Serial Flash Memory Address Register
TBDR	0x0	TX Buffer Data Register

修改

Data.Set A:&QSPI_Cntl_BASE+0x104 %LE %Long 0x00000000 ; QuadSPI0->SFACR; PPWB = 0

为

Data.Set A:&QSPI_Cntl_BASE+0x104 %LE %Long 0x00020000 ; QuadSPI0->SFACR; PPWB = 0

17 BYTE_SWAP	Byte swapping In case of Octal DDR mode, this field controls whether a word unit composed of 2 bytes from posedge and negedge of a single DQS cycle needs to be swapped. 0 - One word of two bytes at [nth, n+1th] address 1 - One word of two bytes at [n+1th, nth] address

然后就是 LUT 的初始化，以及之前的配置 QSPI NOR 可写，再配置为 DOPI 模式的配置调用。

2.4 QuadSPI_Write32BytesDOPI 读函数分析

QuadSPI_Read32BytesDOPI:

Data.Set A:&QSPI_Cntl_BASE+0x100 %Long &QSPI_BASE ; SFAR , FLASH BASE ADDRESS

Field	Description
31-0 SFADR	Serial flash memory address

; clear reception flags prior to triggering IP read command

&temp=Data.Long(A:&QSPI_Cntl_BASE)

Data.Set A:&QSPI_Cntl_BASE %Long (&temp|0x0c00) ;Clear Tx/Rx buffer

11 CLR_TXF	Clear TX FIFO/buffer This is a self-clearing field that invalidates the TX buffer content. NOTE Software must wait for at least five system cycles and three flash cycles after writing '1' to this field. 0 - No action 1 - Read and write pointers of the TX buffer are reset to 0 and TBSR[TRCTR] is reset to 0.
10	Clear RX FIFO

Table continues on the next page...

Table continued from the previous page...

Field	Description
CLR_RXF	This is a self-clearing field that invalidates the RX buffer content. 0 - No action 1 - Read and write pointers of the RX buffer are reset to 0 and RBSR[RDBFL] is reset to 0.

; clear the error flags

Data.Set A:&QSPI_Cntl_BASE+0x160 %LE %Long 0xFFFFFFFF

Register	Offset
FR	160h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLPFF	0	Reserved	DLLABRT	TBFF	TBUF	0	DLLULNLCK	ILLINE	0		0	0		RBOF	RBDF
W	W1C			W1C	W1C	W1C		W1C	W1C						W1C	W1C
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AAEF	AITEF	AIBSEF	ABOF	IUEF	CRCAEF	CRCBEF	PPWF	IPAEF	IPIEF	0	0		0		TFF
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C						W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

;set WMRK level

; write sequence ID and assert Read command

Data.Set A:&QSPI_Cntl_BASE+0x08 %Long ((7.<<24.)+32.) ; sequence 7 + 32 bytes to be read

27-24 SEQID	Points to a sequence in the LUT This field contains the sequence index of the LUT. See LUT for details. Each sequence index can accommodate up to 10 instructions (2 instructions per register). A write to this field triggers a transaction on the serial flash memory interface.

15-0 IDATSZ	IP data transfer size This field defines the data transfer size, in bytes, of the IP command.

```
PRINT "1st 0x" Data.Long(A:&QSPI_Cntl_BASE+0x200)
PRINT "2nd 0x" Data.Long(A:&QSPI_Cntl_BASE+0x204)
PRINT "3rd 0x" Data.Long(A:&QSPI_Cntl_BASE+0x208)
PRINT "4th 0x" Data.Long(A:&QSPI_Cntl_BASE+0x20C)
PRINT "5th 0x" Data.Long(A:&QSPI_Cntl_BASE+0x210)
```

200 - 2FC	RX Buffer Data Register (RBDR0 - RBDR63)	32	RO	0000_0000

```
; clear the RX Buffer and reception flags, counters.
&temp=Data.Long(A:&QSPI_Cntl_BASE)
Data.Set A:&QSPI_Cntl_BASE %Long (&temp|0x0c00)   ;Clear Tx/Rx buffer
RETURN
```

2.5 增加 AHB read 寄存器配置

根据文档《S32G QuadSPI Deep Dive Application notes》说明，AHB read 的配置方法是：

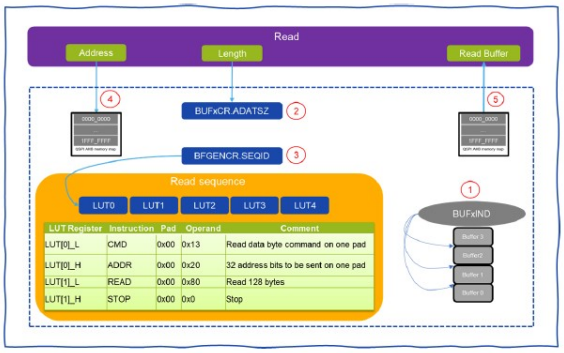


Figure 17. **Read – AHB command**

1. Configure flexible read AHB buffer size in BUFxIND
2. Typically, BUF0IND=BUF1IND=BUF2IND = 0, which means the size of buffer0, buffer1, and buffer2 is 0. The buffer3 is 1024 bytes.
3. Configure for any read access routed to buffer0 ~ buffer3 in BUFxCR
4. Optionally, buffer3 may be configured as an "all master" buffer by writing 1 to BUF3CR[ALLMST]
5. Set the amount of data to be fetched from the flash memory on every missed access in BUFxCR[ADATSZ] field
6. Configure the correct sequence ID in the BFGEBCR[SEQID] field
7. Choose a start address for reading in the memory mapped area
8. Read data from the memory mapped area directly

所以如下设置 Buffer3：

```
;Data.Set A:&QSPI_Cntl_BASE+0x10 %LE %Long 0x0000040B       ; QuadSPI0->BUF0CR = 32 bytes prefetch size, HSE master ID
;Data.Set A:&QSPI_Cntl_BASE+0x30 %LE %Long 0x00000400       ; QuadSPI0->BUF0IND = 1024 bytes buffer size
```

Data.Set A:&QSPI_Cntl_BASE+0x1c %LE %Long 0x80000400 ; QuadSPI0->BUF3CR = 32 bytes prefetch size, all master enable

Field	Description
31 ALLMST	All master enable When set, buffer3 acts as an all-master buffer. Any AHB access with a master ID not matching with the master ID of buffer0, buffer1, or buffer2 is routed to buffer3. When set, the MSTRID field of this register is ignored.

15-8 ADATSZ	AHB data transfer size Defines the read data transfer size in 8 bytes of an AHB triggered read access to serial flash memory. When ADATSZ = 0, the data size mentioned in the sequence pointed to by the SEQID field overrides this value. If OTFAD is enabled, the QuadSPI module prefetches only the required data from the flash memory so that the prefetch never crosses the 1 KB boundary.

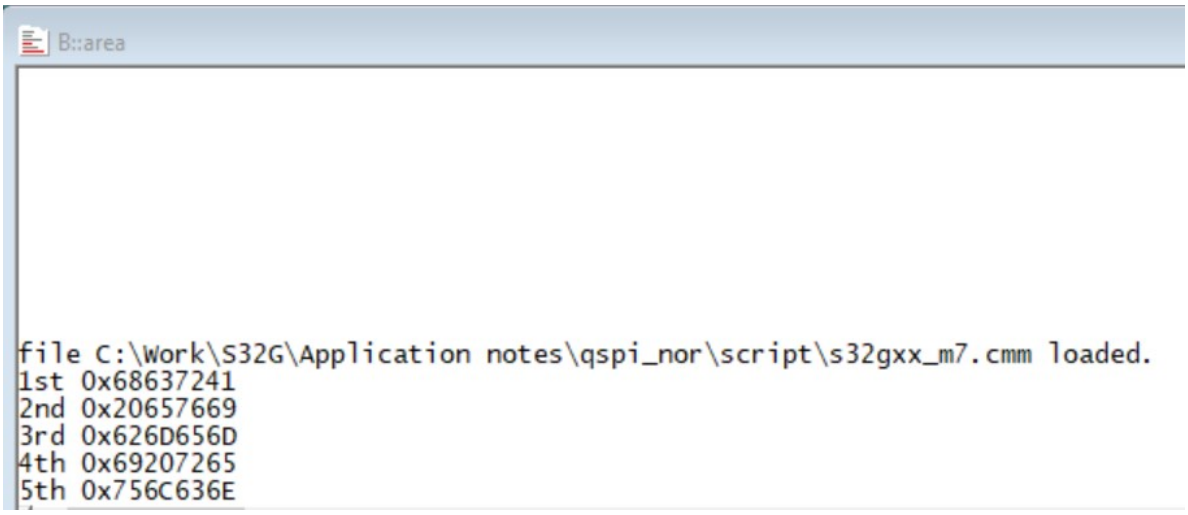
Data.Set A:&QSPI_Cntl_BASE+0x20 %LE %Long 0x00007000 ; QuadSPI0->BFGENCR SEQID= 7 8DTRD - READ DOPI mode

15-12 SEQID	Points to a sequence in the LUT. This field contains the sequence index of the LUT... See LUT. NOTE If the sequence pointer differs in the new and the previous sequences, you should reset it. See sequence pointer clear register for more information.

2.6 测试结果

将 RDB2 板设置为下载模式，或者在 QSPI NOR 烧入带错误 IVT 头的镜像，启动：

1. 运行 lauterbach: t32marm.exe：File->Run script…= s32gxx_m7.cmm。
2. 然后在命令栏里敲 area 命令，就可以看到打印出来的 RX buffer 里的值：

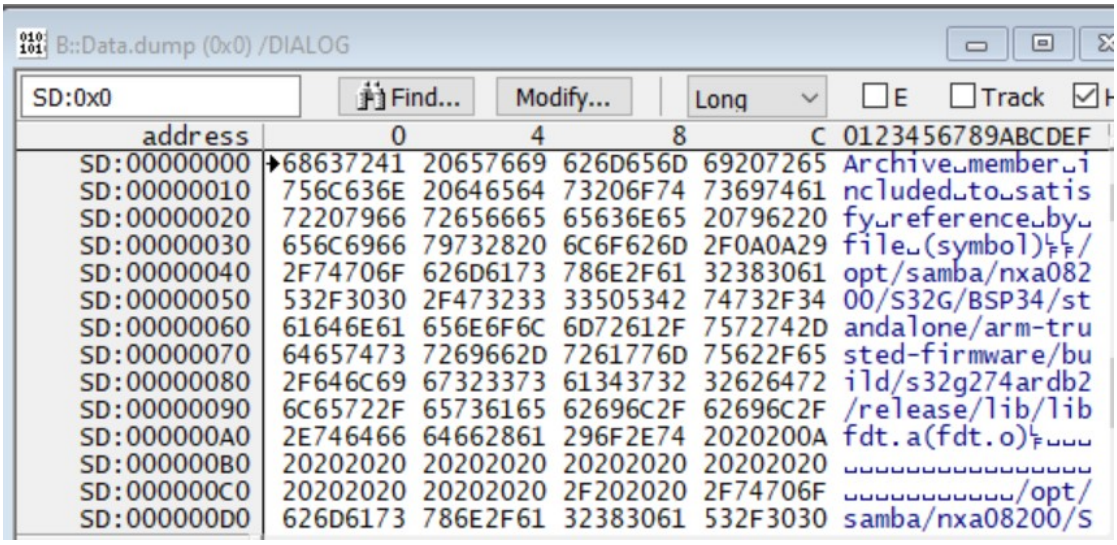


S32G QSPI_NOR Script

3. 然后在菜单 View->dump…里，输入地址 0x0：

Start address (hex)	End address (hex)	Size (KB)	40-bit Master Description 32-bit Master Description (except M7) M7 Description HSE M7 Description	A53 CC FlexNOC Slave port	M7 Default Cache mode	M7 Bus	M7 Memory Space	M7 Memory Type
Code								
0x00_0000_0000	0x00_1FFF_FFFF	524288	QSPI AHB Buffer	s_flash	WT	AXIM	Code	Normal

可以看到 AHB 地址有读出来的 QSPI NOR 内容：



3 高速写开发流程

主函数增加：

```
GOSUB QuadSPI_EraseDOPI ;//erase
GOSUB QuadSPI_Write32BytesDOPI //write
GOSUB QuadSPI_Read32BytesDOPI  //read again
```

3.1 Erase lut 分析及调用

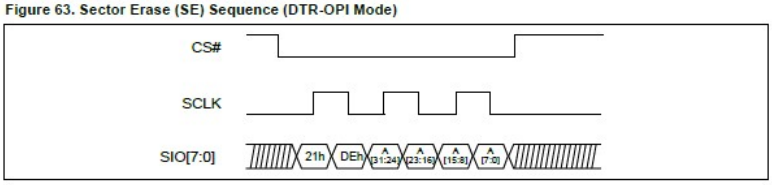
Sector Eraser seq 为：

;Program LUT45 with SE - SECTOR ERASE DOPI mode

Data.Set A:&QSPI_Cntl_BASE+0x3C4 %LE %Long 0x47DE4721 ; SEQID 9

Data.Set A:&QSPI_Cntl_BASE+0x3C8 %LE %Long 0x00002B20

Figure 63. Sector Erase (SE) Sequence (DTR-OPI Mode)



	Instr(6 bits)	Pads(8 bits)	Operand(8 bits). 查 QSPI 数据手册命令字一节，或参考 S32G 指令集表参数。
47de	0x17(CMD_DDR)	0x3(8 bit)	0xde 0x21(Octa I/O DTR sector erase)
4721	0x17(CMD_DDR)	0x3(8 bit)	
2b20	0x10(ADDR_DDR)	0x3(8 bit)	0x20(32 Addr bits to be sent on 4 pad)
0000	0x0(STOP)	0x0(1 bit)	0x0(STOP)

调用如下：

QuadSPI_EraseDOPI:

Data.Set A:&QSPI_Cntl_BASE+0x100 %Long &QSPI_BASE ; SFAR , FLASH BASE ADDRESS

; clear reception flags prior to triggering IP read command

&temp=Data.Long(A:&QSPI_Cntl_BASE)

Data.Set A:&QSPI_Cntl_BASE %Long (&temp|0x0c00) ;Clear Tx/Rx buffer

; clear the error flags

Data.Set A:&QSPI_Cntl_BASE+0x160 %LE %Long 0xFFFFFFFF

;set WMRK level

; write sequence ID and assert WriteEnable id command

Data.Set A:&QSPI_Cntl_BASE+0x08 %Long (8.<<24.) ; sequence

wait 100.ms

; write sequence ID and assert ERASE id command

Data.Set A:&QSPI_Cntl_BASE+0x08 %Long (9.<<24.) ; sequence

wait 200.ms

; clear the RX Buffer and reception flags, counters.

&temp=Data.Long(A:&QSPI_Cntl_BASE)

Data.Set A:&QSPI_Cntl_BASE %Long (&temp|0x0c00) ;Clear Tx/Rx buffer

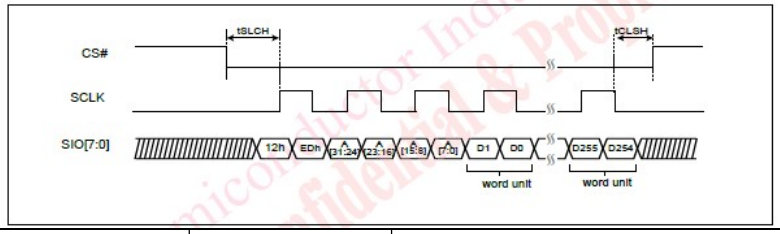
RETURN

3.2 Write lut 分析及调用

Page program seq 设计为：

```
;Program LUT55 with PP - PAGE PROGRAM DOPI mode
Data.Set A:&QSPI_Cntl_BASE+0x3EC %LE %Long 0x47ED4712   ;          SEQID 11
Data.Set A:&QSPI_Cntl_BASE+0x3F0 %LE %Long 0x3F102B20
Data.Set A:&QSPI_Cntl_BASE+0x3F4 %LE %Long 0x00000000
```

Figure 72. Page Program (PP) Sequence (DTR-OPI Mode)



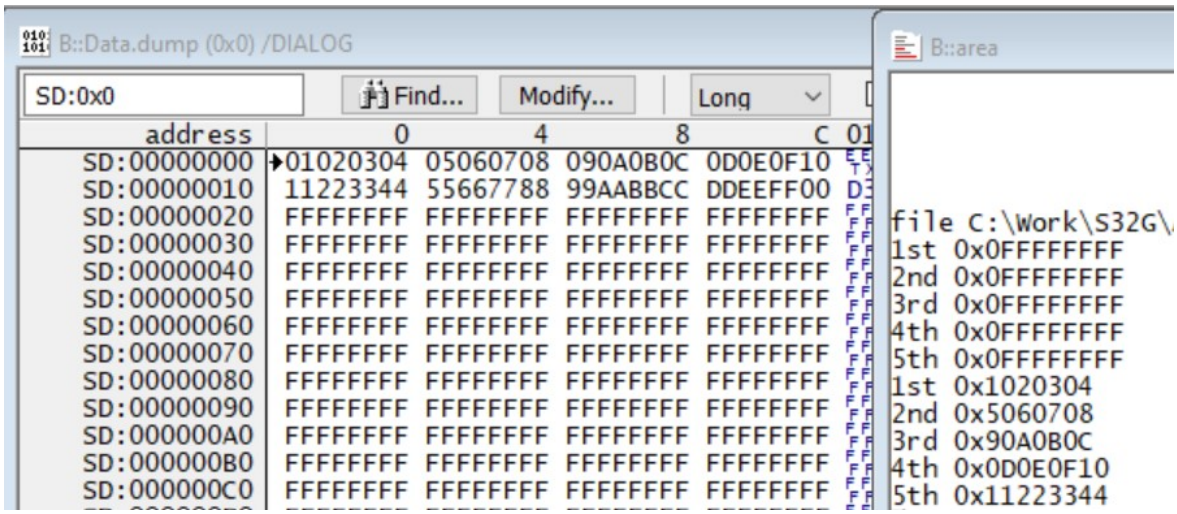
	Instr(6 bits)	Pads(8 bits)	Operand(8 bits). 查 QSPI 数据手册命令字一节，或参考 S32G 指令集表参数。
47ed	0x17(CMD_DDR)	0x3(8 bit)	0xed 0x12(Octa I/O DTR page program)
4712	0x17(CMD_DDR)	0x3(8 bit)	
2b20	0x10(ADDR_DDR)	0x3(8 bit)	0x20(32 Addr bits to be sent on 4 pad)
3f10	0xf(WRITE_DDR)	0x3(8 bit)	0x10(write data size in bytes)
0000	0x0(STOP)	0x0(1 bit)	0x0(STOP)

调用如下：

```
QuadSPI_Write32BytesDOPI:
  Data.Set A:&QSPI_Cntl_BASE+0x100 %Long  &QSPI_BASE      ; SFAR , FLASH BASE ADDRESS
  ; clear the error flags
  Data.Set A:&QSPI_Cntl_BASE+0x160 %LE %Long 0xFFFFFFFF
  ; load TX buffer
  ; clear TX counters
  &temp=Data.Long(A:&QSPI_Cntl_BASE)
  Data.Set A:&QSPI_Cntl_BASE %Long (&temp|0x0c00)   ;Clear Tx/Rx buffer
  Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0x01020304
  Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0x05060708
```

```
  Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0x090A0B0C
  Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0x0D0E0F10
  Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0x11223344
  Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0x55667788
  Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0x99AABBCC
  Data.Set A:&QSPI_Cntl_BASE+0x154 %LE %Long 0xDDEEFF00
  ; write sequence ID and assert WriteEnable id command
  Data.Set A:&QSPI_Cntl_BASE+0x08 %Long  (8.<<24.)  ; sequence
  wait 100.ms
  ; write sequence ID and assert ERASE id command
  Data.Set A:&QSPI_Cntl_BASE+0x08 %Long  ((11.<<24.)+32.)  ; sequence + 32 bytes to be written
  wait 200.ms
  ; clear the RX Buffer and reception flags, counters.
  &temp=Data.Long(A:&QSPI_Cntl_BASE)
  Data.Set A:&QSPI_Cntl_BASE %Long (&temp|0x0c00)   ;Clear Tx/Rx buffer
RETURN
```

3.3 测试结果



3.4 Lauterbach 烧写镜像脚本说明

Lauterbach 烧写 QSPI NOR 的脚本是由 Lauterbach 负责发布，如果安装了支持 S32G 的 Trace32 版本，可以参考：

S32G QSPI_NOR Script

C:\T32\demo\arm\flash\ s32g274-cm7-qspi.cmm，它用到的算法 binary:

```
FLASH.TARGET 0x34000000 E:0x34002000 0x2000 ~~/demo/arm/flash/byte/snor_s32g274.bin /DualPort
```

在：C:\T32\demo\arm\flash\byte\snor_s32g274.bin。相关修改和开发支持可以联系 Lauterbach。