AN13112

如何使用.mac 文件初始化 FlexSPI 设备

版本 0 — 2021 年 8 月 30 日

应用笔记

目录

1 引言

i.MX RT 系列单片机是恩智浦的跨界产品。i.MX RT 包含了一个 FlexSPI 控制器，该控制器支持多种设备，如 Nor Flash，HyperBus 设备等。

为了调试方便，一般将程序直接下载到片内 RAM 或片外 RAM 中运行。当在片外 RAM 中调试代码时，片外 RAM 需要在代码下载进 RAM 之前进行初始化，IAR 是通过.mac 文件实现的。

本应用笔记以 IAR 为调试环境，描述了如何使用.mac 文件来初始化连接到 i.MX RT FlexSPI 控制器的设备，如 HyperRAM，Flash 等。

硬件是基于 RT1060-EVK，软件基于 SDK 2.7.0。

2 概述

在嵌入式开发中，由于对 Flash 芯片的烧录，读取速度较慢，而调试的时候需要频繁修改程序，对程序的读取、写入速度对开发速度影响很大。因此对调试来说，一般将程序直接烧入片内 RAM 或者是片外 RAM 中运行，这样可以提升调试速度。

使用片内 RAM 调试时，一般将 IAR 的.icf 文件中 ROM 和 RAM 地址都设成片内 RAM 的地址即可。但有时片内 RAM 空间并不够使用，此时就需要使用片外 RAM，当在外部 RAM 中调试代码时，片外 RAM 需要在代码下载进 RAM 之前进行初始化，在 IAR 中是通过.mac 文件进行片外 RAM 的初始化的。

本应用笔记主要描述如何用.mac 文件初始化连接至 FlexSPI 的片外 RAM，其主要包括以下步骤：

1. PIN 脚初始化
2. 时钟初始化
3. FlexSPI 初始化和设备初始化

完成以上三个步骤，连接到 FlexSPI 的设备就被成功初始化了。

3 例程

本节以 hello_world 例程为例，阐述了如何修改 SDK 的.mac 文件来初始化 HyperRAM 设备。对于初始化其他设备，如 Flash 等，仿照初始化 HyperRAM 的代码，根据不同设备参数，修改相关寄存器数值即可。

1. 进入 SDK_2.7.0_EVK-MIMXRT1060\boards\evkmimxrt1060\demo_apps\hello_world\iar 路径。
2. 复制 evkmimxrt1060.mac 并重新命名为 evkmimxrt1060_hyperram_init.mac。
3. 打开 evkmimxrt1060_hyperram_init.mac 文件，加入相应初始化函数分别用于初始化 PIN 脚，时钟以及对应设备，如 图 1 所示：

```
execUserPreload()
{
  _load_dcdc_trim();
  _pin_init();
  _clock_init();
  _hyperram_Init();
}

execUserReset()
{
  _load_dcdc_trim();
  _pin_init();
  _clock_init();
  _hyperram_Init();
}
```

图 1. 添加初始化函数

3.1 PIN 脚初始化

初始化 FlexSPI 与设备硬件连接的相应 PIN 脚，一般有数据，时钟，DQS 等 PIN 脚，根据具体连接的 PIN 脚进行初始化，如图 2 所示：

```
_pin_init()
{
  // Set Pin Mux for HyperRAM
  __writeMemory32(0x00000011, 0x401F81D4, "Memory"); // FLEXSPIB_DATA03
  __writeMemory32(0x00000011, 0x401F81D8, "Memory"); // FLEXSPIB_DATA02
  __writeMemory32(0x00000011, 0x401F81DC, "Memory"); // FLEXSPIB_DATA01
  __writeMemory32(0x00000011, 0x401F81E0, "Memory"); // FLEXSPIB_DATA00
  __writeMemory32(0x00000011, 0x401F81E4, "Memory"); // FLEXSPIB_SCLK
  __writeMemory32(0x00000011, 0x401F81E8, "Memory"); // FLEXSPIA_DQS
  __writeMemory32(0x00000011, 0x401F81EC, "Memory"); // FLEXSPIA_SS0_B
  __writeMemory32(0x00000011, 0x401F81F0, "Memory"); // FLEXSPIA_SCLK
  __writeMemory32(0x00000011, 0x401F81F4, "Memory"); // FLEXSPIA_DATA00
  __writeMemory32(0x00000011, 0x401F81F8, "Memory"); // FLEXSPIA_DATA01
  __writeMemory32(0x00000011, 0x401F81FC, "Memory"); // FLEXSPIA_DATA02
  __writeMemory32(0x00000011, 0x401F8200, "Memory"); // FLEXSPIA_DATA03

  // Set Pin Config for HyperRAM
  __writeMemory32(0x000110F9, 0x401F83C4, "Memory"); // FLEXSPIB_DATA03
  __writeMemory32(0x000110F9, 0x401F83C8, "Memory"); // FLEXSPIB_DATA02
  __writeMemory32(0x000110F9, 0x401F83CC, "Memory"); // FLEXSPIB_DATA01
  __writeMemory32(0x000110F9, 0x401F83D0, "Memory"); // FLEXSPIB_DATA00
  __writeMemory32(0x000110F9, 0x401F83D4, "Memory"); // FLEXSPIB_SCLK
  __writeMemory32(0x000110F9, 0x401F83D8, "Memory"); // FLEXSPIA_DQS
  __writeMemory32(0x000110F9, 0x401F83DC, "Memory"); // FLEXSPIA_SS0_B
  __writeMemory32(0x000110F9, 0x401F83E0, "Memory"); // FLEXSPIA_SCLK
  __writeMemory32(0x000110F9, 0x401F83E4, "Memory"); // FLEXSPIA_DATA00
  __writeMemory32(0x000110F9, 0x401F83E8, "Memory"); // FLEXSPIA_DATA01
  __writeMemory32(0x000110F9, 0x401F83EC, "Memory"); // FLEXSPIA_DATA02
  __writeMemory32(0x000110F9, 0x401F83F0, "Memory"); // FLEXSPIA_DATA03
}
```

图 2. PIN 脚初始化

3.2 时钟初始化

初始化 FlexSPI 时钟，根据 HyperRAM 设备支持的时钟频率以及 FlexSPI 控制器支持的最大时钟频率进行设置，如 图 3 所示：

```
_clock_init()
{
  __var reg;
  // Enable all clocks
  __writeMemory32(0xffffffff, 0x400FC068, "Memory");
  __writeMemory32(0xffffffff, 0x400FC06C, "Memory");
  __writeMemory32(0xffffffff, 0x400FC070, "Memory");
  __writeMemory32(0xffffffff, 0x400FC074, "Memory");
  __writeMemory32(0xffffffff, 0x400FC078, "Memory");
  __writeMemory32(0xffffffff, 0x400FC07C, "Memory");
  __writeMemory32(0xffffffff, 0x400FC080, "Memory");

  // Config PLL for FlexSPI
  // PERCLK_PODF: 1 divide by 2
  __writeMemory32(0x04900001, 0x400FC01C, "Memory");
  // Enable SYS PLL but keep it bypassed.
  __writeMemory32(0x00012001, 0x400D8030, "Memory");
  do
  {
    reg = __readMemory32(0x400D8030, "Memory");
  }while((reg & 0x80000000) == 0);
  // Disable bypass of SYS PLL
  __writeMemory32(0x00002001, 0x400D8030, "Memory");

  // PFD2_FRAC: 29, PLL2 PFD2=528*18/PFD2_FRAC=327
  // Ungate SYS PLL PFD2
  __writeMemory32(0x001D0000, 0x400D8100, "Memory");

  __writeMemory32(0x44100001, 0x400FC01C, "Memory");//divide by 1

  __message "clock init done\n";
}
```

图 3. 时钟初始化

3.3 FlexSPI 及设备初始化

FlexSPI 控制器初始化：

1. 在配置之前先重置 FlexSPI。
2. 设置 MCR0[MDIS]为 0x1（确保此时 FlexSPI 控制器处于停止模式）。
3. 配置 FlexSPI 模块控制相关寄存器：MCR0，MCR1，MCR2（注意确保 MCR0[MDIS]为 0x1）。
4. 如果用到 AHB 命令，则需要配置 AHB 总线控制寄存器（AHBCR）以及 AHB 接收缓冲区控制寄存器（AHBRXBUFxCR0）。
5. 配置 IP 接收/发送 FIFO 控制寄存器：IPRXFCR，IPTXFCR。

设备初始化：

1. 根据连接的 HyperRAM 设备参数，配置 Flash 控制寄存器：FLSHxCR0，FLSHxCR1，FLSHxCR2。
2. 根据选择的时钟源，配置 DLL 控制寄存器（DLLxCR）。
3. 根据 write mask 使能与否，配置 flash 控制寄存器 FLSHxCR4。
4. 设置 MCR0[MDIS]为 0x0（退出停止模式）。
5. 通过配置 LUTKEY，LUTCR 寄存器解锁 LUT。
6. 根据 HyperRAM 设备参数，更新 LUT 表（用于 AHB 命令或 IP 命令访问）以设置 HyperRAM 的数据读写，寄存器读写等操作的时序。

7. 配置 LUTKEY，LUTCR 寄存器锁定 LUT。
8. 通过设置 MCR0[SWRESET]为 0x1，重置 FlexSPI 控制器。

具体实现代码如 图 4 所示：

```
_hyperram_Init()
{
  // Config FlexSPI Registers
  __writeMemory32(0xFFFF80C0, 0x402A8000, "Memory"); // MCR0
  _Flexspi_reset();
  __writeMemory32(0xFFFF3032, 0x402A8000, "Memory"); // MCR0
  __writeMemory32(0xFFFFFFFF, 0x402A8004, "Memory"); // MCR1
  __writeMemory32(0x200801F7, 0x402A8008, "Memory"); // MCR2
  __writeMemory32(0x00000078, 0x402A800C, "Memory"); // AHBCR
  __writeMemory32(0x80000020, 0x402A8020, "Memory"); // AHBRXBUFCR00
  __writeMemory32(0x80000020, 0x402A8024, "Memory"); // AHBRXBUFCR01
  __writeMemory32(0x80000020, 0x402A8028, "Memory"); // AHBRXBUFCR02
  __writeMemory32(0x80000020, 0x402A802C, "Memory"); // AHBRXBUFCR03
  __writeMemory32(0x00000000, 0x402A80B8, "Memory"); // IPRXFCR
  __writeMemory32(0x00000000, 0x402A80BC, "Memory"); // IPTXFCR
  // Config HyperRAM
  __writeMemory32(0x00004000, 0x402A8060, "Memory"); // FLASHA1CR0
  __writeMemory32(0x00021C63, 0x402A8070, "Memory"); // FLASHA1CR1
  __writeMemory32(0x00000100, 0x402A8080, "Memory"); // FLASHA1CR2
  __writeMemory32(0x00001D00, 0x402A80C0, "Memory"); // DLLCR
  __writeMemory32(0x00000000, 0x402A8094, "Memory"); // FLASHA1CR4
  __writeMemory32(0x00000004, 0x402A8094, "Memory"); // FLASHA1CR4
  __writeMemory32(0xFFFF3030, 0x402A8000, "Memory"); // MCR0
  __writeMemory32(0x5AF05AF0, 0x402A8018, "Memory"); // LUTKEY
  __writeMemory32(0x00000002, 0x402A801C, "Memory"); // LUTCR
  __writeMemory32(0x8B1887A0, 0x402A8200, "Memory"); // LUT[0]
  __writeMemory32(0xB7078F10, 0x402A8204, "Memory"); // LUT[1]
  __writeMemory32(0x0000A704, 0x402A8208, "Memory"); // LUT[2]
  __writeMemory32(0x00000000, 0x402A820C, "Memory"); // LUT[3]
  __writeMemory32(0x8B188720, 0x402A8210, "Memory"); // LUT[4]
  __writeMemory32(0xB7078F10, 0x402A8214, "Memory"); // LUT[5]
  __writeMemory32(0x0000A304, 0x402A8218, "Memory"); // LUT[6]
  __writeMemory32(0x00000000, 0x402A821C, "Memory"); // LUT[7]
  __writeMemory32(0x8B1887E0, 0x402A8220, "Memory"); // LUT[8]
  __writeMemory32(0xB7078F10, 0x402A8224, "Memory"); // LUT[9]
  __writeMemory32(0x0000A704, 0x402A8228, "Memory"); // LUT[10]
  __writeMemory32(0x00000000, 0x402A822C, "Memory"); // LUT[11]
  __writeMemory32(0x8B188760, 0x402A8230, "Memory"); // LUT[12]
  __writeMemory32(0xB7078F10, 0x402A8234, "Memory"); // LUT[13]
  __writeMemory32(0x0000A304, 0x402A8238, "Memory"); // LUT[14]
  __writeMemory32(0x00000000, 0x402A823C, "Memory"); // LUT[15]
  __writeMemory32(0x5AF05AF0, 0x402A8018, "Memory"); // LUTKEY
  __writeMemory32(0x00000001, 0x402A801C, "Memory"); // LUTCR
  _Flexspi_reset();
  __message "HyperRAM init done\n";
}
      _Flexspi_reset()
      {
        __var reg;
         reg = __readMemory32(0x402A8000, "Memory");
        __writeMemory32((reg | 0x1), 0x402A8000, "Memory");
        do
        {
          reg = __readMemory32(0x402A8000, "Memory");
        }while((reg & 0x1) != 0);
      }
```

图 4. FlexSPI 及设备初始化

4 参考资料

1. *i.MX RT1060 Processor Reference Manual*（文档 IMXRT1060RM）

5 修订历史

版本号	日期	说明
0	2021 年 8 月 30 日	初始版本

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2021 年 8 月 30 日

Document identifier: AN13112

arm