Application Note
ZHCACT7 – 6. 2023

AFE7769 器件软件指南

Jason Ren

摘要

AFE7769 是 TI 的 4T4R2F 高集成度射频收发器芯片，集成了基础的射频通道功能，产品系列中在未来也会集成强大的通信 DPD/CFR 数字处理功能。在 AFE7769 应用在通信系统中时，需要系统软件对器件的底层寄存器进行大量读写操作，使得 AFE7769 和系统中其它信号进行交互。因此需要用户对 AFE7769 和系统软件交互接口进行仔细设计。TI 为了简化客户开发流程，统一维护接口，提供了基于 C 的软件包，实现从器件初始化到器件在线动作控制的功能。TI Transceiver 的产品族内的产品，包括 AFE77xx，AFE79xx 以及 AFE80xx 系列都使用了该软件设计思路，该 App Note 也可对这些系列的产品软件开发有一定的参考作用。本篇 App Note 针对用户在将 TI 软件包封装编译方法进行解释说明，同时解释重点 API 的使用方法，方便用户参照该 App Note 进行系统软件交互设计。

目录

图/表

1 引言

AFE7769 是 TI 的基于零中频架构的射频 Transceiver，其中发射和接收通道为零中频架构，反馈通道为射频直采架构。内部集成的多 PLL 使得器件在各个通信应用场景中可以灵活使用，高集成度的特性使得用户更容易进行板级空间设计。产品族内，也包含带有强大数字处理功能 DPD/CFR 的产品，帮助用户进一步节省整版的成本和功耗。

由于器件功能非常强大且灵活，从器件初始化到器件起机后的在线配置，用户对器件的交互动作比较复杂，为了简化用户软件开发流程，TI 在底层寄存器的操作基础上，开发了基于 C 的软件包。软件帮助用户处理设备的初始化流程以及在初始化后的在线控制功能。同时，在开发软件包时，加入了平台兼容性的设计，用户可以非常灵活地进行平台性升级。

本文针对 AFE7769 产品族的软件包的特性，结构及集成方法进行了介绍，针对软件编译过程中的关键点进行了说明，给出了用户在调试初期建议集成的 CAPI 进行了声明，方便用户更快更容易地从零开始针对 AFE7769 器件搭建软件环境。

2 CAPI 软件库简介

2.1 关键特性

为了便于用户开发和维护，TI AFE7769 的软件库包含了以下特性。

1. Error handling: 每个函数都会返回执行成功/失败信息，便于系统应用中调试接口，返回值说明如下：
    - TI_AFE_RET_EXEC_PASS: 函数执行成功的情况下出现
    - TI_AFE_RET_EXEC_FAIL: 系统参数配置有效，但是函数执行失败。一般会在器件实际工作状态异常，软件底层接口调用失败，函数传参非法的情况下出现
2. Multiple AFE Device Support: 在同一系统软件 host 控制不同 AFE7769 器件时使用。软件定义 afeInst 为不同 AFE7769 的器件 ID，传参给所有软件 Lib 的 function。
3. Channel Remapping: 由于不同客户，不同应用项目中的 channel number 定义不同，TI 软件 Lib 提供了可配置的 channel number mapping 定义方法。
4. Simulation Mode: 一般用户开发流程会在前期硬件还未准备好的情况下针对软件进行调试，需要 TI 软件可脱离实际硬件来进行调试。但是由于很多函数需要器件的寄存器回读值满足一定条件才能正常执行，而脱离硬件的调试条件下，软件无法读取到所需的寄存器值。因此 TI Lib 在仿真模式下屏蔽了寄存器回读和校验的操作，可通过配置 AFE77D_LIBS_RUN_MODE 进入仿真模式。

5. Function Renaming: 由于用户的函数命名习惯不同，在不同的用户软件系统中需要针对用户定义接口函数进行特定函数前缀进行命名。用户可通过修改 MakeFile 中的 AFE77D_FUNC_NAME_PREFIX 进行修改。

2.2 软件包结构

TI Lib 软件包主要由四部分组成: TI Lib 自定义的 API 函数，用户 host 维护的驱动模块，示例 main 函数，makeFile 示例。

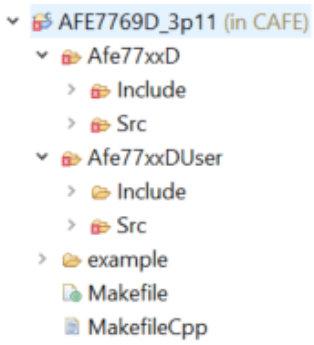

Figure 1. 软件包结构

下文针对这四块进行简要介绍。

2.2.1 TI Lib 自定义 CAPI

该部分包含在 Figure 1 中的 AFE77xxD 文件夹内。主要包括用户调试中所需的 C Function 以及 Function 的函数声明及宏定义，用户直接编译后可以使用。由 TI 侧进行升级和维护，不建议用户直接进行修改，否则在 TI 升级维护后，用户修改的部分不能迭代到升级后的版本中。

该部分的 Function 非常丰富，在器件应用层面包括各个射频链路的射频控制，AGC/PAP 器件功能配置，JESD 接口/SERDES 接口的调试功能，DPD/CFR 配置（在 AFE7769 支持 DPD/CFR 的版本中）。TI 为了完善 TRX 的配置，增加了部分可以在器件初始化后对系统参数进行更新的 CAPI 接口，比如 AGC、PAP 参数配置 CAPI。使得用户的平台项目可以使用同一静态配置脚本，而子项目的差异化配置可以通过 CAPI 实现。在第 4 章中，会给出在用户前期调试过程中建议集成到用户软件平台中的 CAPI。

2.2.2 用户 host 维护的驱动模块

该部分主要包括在 Figure 1 中的 AFE77xxDUser 的文件夹内，需要由用户进行维护。在 TI 维护产品包时，只会针对 AFE77xxD 的内容进行维护，AFE77xxDUser 中由用户维护的内容不会被覆盖。该部分由用户维护的内容较为简单，主要分为以下两大块：

- tiAfe77D_afeParameters.c: 如前文提到，用户需要针对包括多 AFE 系统中的 AFE ID 进行约束，器件的配置信息进行声明（如采样率，频点，数据速率，JESD 接口配置等）。其中，init_tiAfe77DeviceInfo_t()会初始化 tiAfe77D_deviceInfo 成员以及初始化器件默认参数到软件中。

- tiAfe77D_basicFunc.c: 主要包括 host 软件对 AFE 的驱动函数以及部分由 AFE 对 host 软件发起的延时请求，外部 sysref 信号请求，log 打印等级等功能。举个例子，在下面的代码中，/* TBD: User domain */部分，需要由用户将 host 软件中对 AFE 的 SPI 驱动替换进去。

```
TI_AFE_API_COMP uint8_t AFE77DFNP(afeSpiRawWrite)(AFE77D_INST_TYPE afeInst, uint16_t addr, uint8_t data)
{
   afeLogDbg("WRITE: Addrress: 0x%X, data: 0x%X", addr, data);
   /* TBD: User domain */
   return TI_AFE_RET_EXEC_PASS;
}
```

2.2.3 示例 main 函数

仅作为给用户的 main 函数的示例，包括部分 log 等级控制，参数初始化等动作。

2.2.4 makeFile 示例

TI 提供的 makeFile 示例，需要由用户针对 host 环境的编译环境进行维护。

3 CAPI 在用户软件中的应用

本节介绍如何将 CAPI 正确集成到 host 软件中。

3.1 实例化库

实例化过程定义和约束了器件的配置信息，分为：(1) 实际器件工作配置，如采样率，DDC/DUC 系数，detector 门限等信息；(2)软件配置，如器件 ID，通道映射信息。

TI 推荐两种实例化方法，下面进行介绍：

方法 1：afe77DInstDeviceInfo 的指针传递给软件内所有函数。如下面示例代码所示：

```
afe77DInstDeviceInfo tiAfe77D_deviceInfo[2];

void init_tiAfe77DeviceInfo_t()
{
   extern afe77DInstDeviceInfo tiAfe77D_deviceInfo[2];
   for (uint8_t i = 0; i < 2; i++)
   {
      tiAfe77D_deviceInfo[i].afeId = i;
      AFE77DFNP(setDefaultParams)
      (&tiAfe77D_deviceInfo[i]);
      tiAfe77D_deviceInfo[i].halConfig = NULL;
   }

         }
```

简单介绍一下注意点：

- 定义结构体 ti_Afe77DDeviceInfo_t，并进行参数初始化

- 如 2.2.2 节内所述，在 baseFunc 内集成用户所需的驱动函数
- 在用户需要批量修改函数前缀的情况下，可以在 makeFile 中添加-D 命令。举个例子：

  如需添加“ti_Afe77D_“为所有函数的前缀，应修改为：

-D'TI_AFE80xx_FUNC_NAME_PREFIX(funcName)=(ti_Afe77D_## funcName)'

- 如果不需要针对 channel mapping 进行修改，在编译前需要注释掉 makeFile 中的-DENABLE_RX_CH_REMAP，-DENABLE_TX_CH_REMAP 和-DENABLE_FB_CH_REMAP，反之，不需要注释

方法 2：针对同一 host 软件控制不同 AFE 的应用场景，afe77DInsteviceInfo 定义为 tiAfe77DDeviceInfo_t 数组，对应的 AFE 索引对应的 tiAfe77D_afeParameters。下面为示例代码，注意点和方法 1 类似。

```
afe77DInstDeviceInfo tiAfe77DDeviceInfo_inst[2]; // Example when the host is controlling 2 AFEs. For different number of AFEs, the size of the structure need to be changed.
afe77DInstDeviceInfo *tiAfe77D_deviceInfo;

void init_tiAfe77DeviceInfo_t()
{
   extern afe77DInstDeviceInfo tiAfe77DDeviceInfo_inst[2];
   extern afe77DInstDeviceInfo *tiAfe77D_deviceInfo;
   tiAfe77D_deviceInfo = tiAfe77DDeviceInfo_inst;
   for (uint8_t i = 0; i < 2; i++)
   {
      tiAfe77D_deviceInfo[i].afeId = i;
      AFE77DFNP(setDefaultParams)
      (&tiAfe77D_deviceInfo[i]);
      tiAfe77D_deviceInfo[i].halConfig = NULL;
   }

}
```

3.2 器件初始化配置

针对实际器件的操作分为两部分：器件初始化配置和器件初始化完成后的在线修改配置功能。首先介绍一下初始化配置相关的内容。

软件层面上，器件初始化的过程是在外部硬件环境如供电，参考钟等准备好的前提下，进行寄存器的读写，校验以及请求外部 rst, sysref 信号配合的过程。TI 会提供初始化流程的寄存器表单，由 host 进行解析并按照顺序进行寄存器操作，系统级别的流程说明如下：

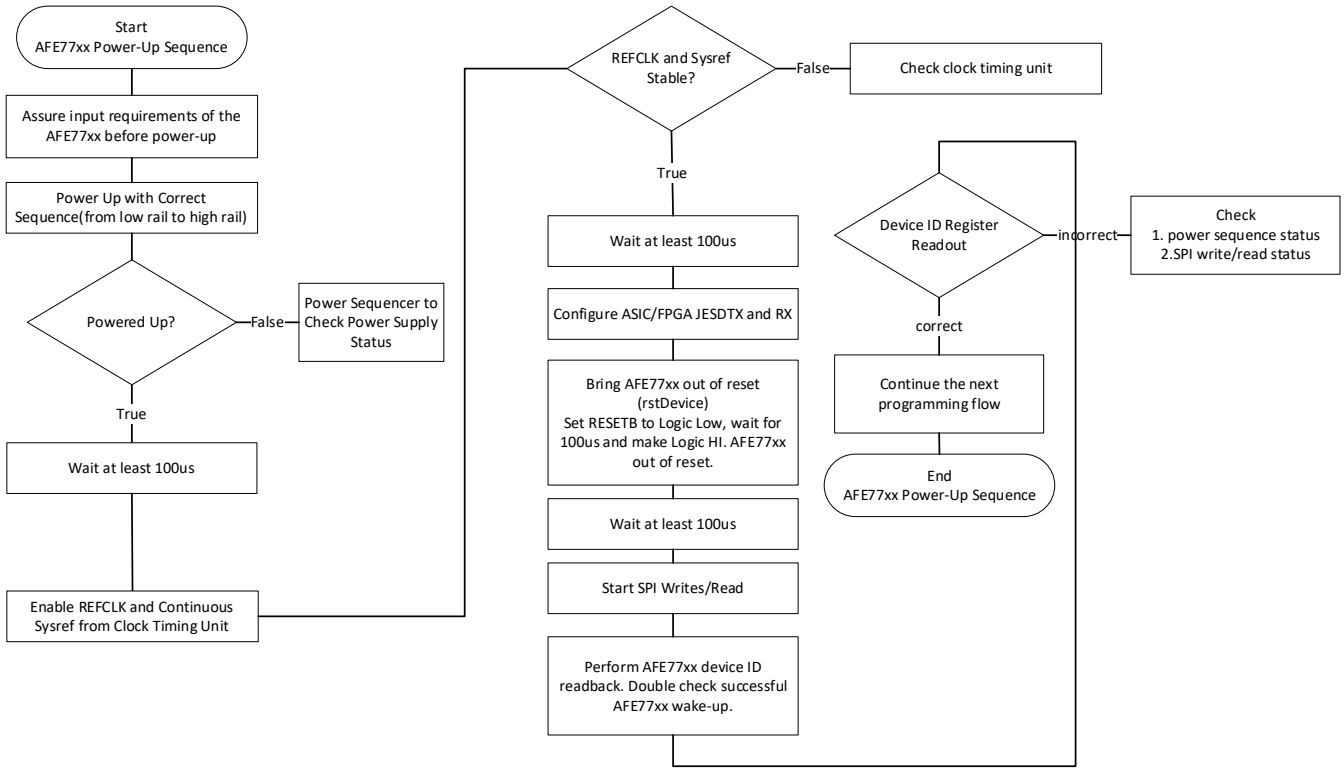

Figure 2. 系统级别初始化流程

针对 Figure2 中硬复位 AFE 后的 Start SPI Writes/Read 操作，进行进一步的说明如图 3 所示，方便用户对器件初始化过程更加了解。其中，省略了更底层的器件动作，只描述了器件模块级别的动作。在下图中，External action 是需要 host 控制其它器件如时钟芯片(LMK04828, LMK5C33216, etc)给出 sysref 信号，FPGA 芯片给出 GPIO 控制信号。这几个动作是无法单独由 AFE 器件完成的，需要用户 host 软件解析 external action，并通知其它受控芯片做出相应动作。需要注意的是，外部动作的 ASIC/FPGA 的 SERDES 发送初始化过程需要保证在图 3 的 Config and bring up SERDES 步骤前进行（甚至可以提前到 Transceiver 初始化前），否则 AFE 侧的 SERDES 模块无法正常自适应 SERDES 链路参数。

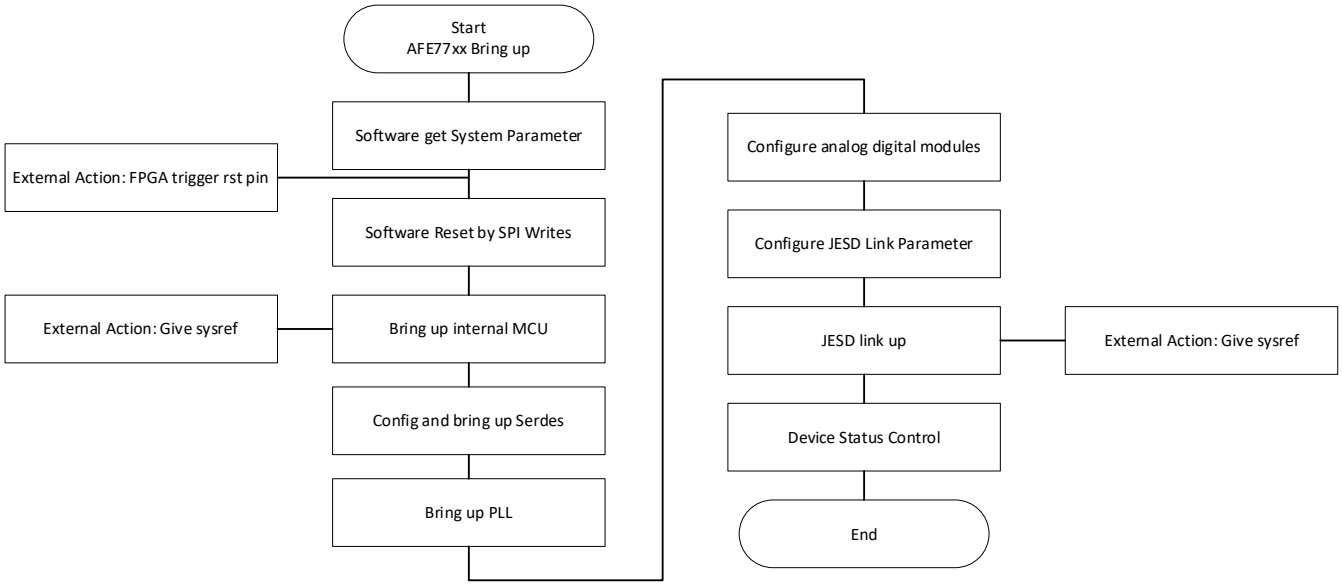

Figure 3. 器件内部模块初始化流程

3.3 配置文件格式

如 3.2 节所述，初始化 AFE 时需要 host 软件根据寄存器表单的寄存器顺序读写寄存器值，本节针对寄存器表单的应用进行描述。

寄存器表单可以由 TI EVM 软件 Latte 生成，或者由技术支持给出。用户拿到寄存器表单后，由 host 软件进行解析并映射到底层驱动中，与 AFE 器件进行交互。TI 的寄存器表单有两种格式可供用户选择，下面分别进行介绍。

3.3.1 原始日志格式(Raw log format)

原始日志格式的结构较为简单明晰，在寄存器表单中包含相应的器件操作动作，如 SPIWrite/ SPIRead/ SPIPoll/SPI Readcheck 等，用户需使用解析代码，将相应的操作动作映射到底层 SPI 驱动中，下面一一进行举例说明：

		函数说明
1	SPIWrite address, Value, LSB, MSB	SPI 写操作。Address: 地址，Value：写入值，LSB：最低有效位，MSB：最高有效位
2	SPIRead address, LSB, MSB	SPI 写操作。Address: 地址，LSB：最低有效位，MSB：最高有效位
3	SPI ReadCheck address, LSB, MSB, expected_Value	读取相应地址寄存器值并与预设值比较，返回比较结果。Address: 地址，LSB：最低有效位，MSB：最高有效位，expected_Value：预设值

4	SPIPoll address, LSB, MSB, expected_Value	以非常短的时间间隔，不停读取相应地址寄存器值并与预设值比较，在与预设值相同后跳出循环；如果比较次数超出了预设门限，则返回error。Address: 地址，LSB：最低有效位，MSB：最高有效位，expected_Value：预设值

3.3.2 十六进制格式（Hex format）

Hex format 是 16 进制操作存储格式，存储在用户 host 软件中，由 API 进行解析并映射到底层驱动中。相比于 raw log format，具有更小文件大小，升级寄存器时不需重新编译版本等优势。下面对文件格式进行描述。

每个操作的操作动作(Opcode)，操作入参(Operand)如寄存器地址，寄存器值，LSB，MSB 以 8-byte 的形式进行存储。每个操作都是 host 软件解析 Opcode，找到对应的底层驱动，然后解析 Operand，将正确入参传入底层驱动中。简单来说，Opcdoe 告知 host 软件下一步该进行哪种软件交互操作，而 Operand 是在 host 软件知晓交操作种类后的交互操作入参。注意，本文中讲解的是在 host 软件层面的 Opcode 和 Operand 关系，是需要软件设计者关注的。而在 AFE 内部的软件方面，有部分的 Macro 宏指令是集成到片上 MCU 的，host 软件通过底层寄存器下发 Macro 的 Opcode 和 Operand，由 MCU 自行访问相应指令集。

这里先介绍 Opcode 和 Operand 的解析方法，再进行举例说明。

Opcode 存储在高 16bit，低 16bit 为 0。下图为 Opcode 对应的操作动作：

0	1	2	3	4	5	6	7	8	9	10	11	0b10xx
Write	Read	Read Check	Poll	Delay	Start New Step	Macro Execute	Give Pin Sysref	Pll Spi Access	Load DSA Packet	Reserved	Burst Write	System Parameter Update

Opcode 一般占用 4 个 byte（实际占不满的情况下高位补 0），Operand 分成很多种，最低占 2byte，最高不设上限(如 burst writes)。下表说明了 8 byte 和 4 byte 各个 Operand 的格式：

Opcode	Function	Word0 Bit[31:16]	Word0 Bit[15:8]	Word0 Bit[7:0]	Word1 Bit[31:28]	Word1 Bit[27:24]	Word1 Bit[23:16]	Word1 Bit[15:8]	Word1 Bit[7:0]
0	Write	0	xx(0 by default)		MSB	LSB	Data	Address	
1	Read	0x1	Address		NA				
2	Readcheck	0x2	xx(0 by default)		MSB	LSB	Expected Value	Address	
3	Poll	0x3	xx(0 by default)		MSB	LSB	Expected Value	Address	
4	Delay	0x4	Delay in Ms		NA				
5	New Step	0x5	Step Code						
6	Execute Macro	0x6	Num of Operand Bytes	Macro Opcode	Operand Byte 4	Operand Byte 3	Operand Byte 2	Operand Byte 1	…

7	Leak Sysref	0x7	Sysref Mode	NA
8	Pll Access	0x8	Request Type	NA
9	Load DSA Packet	0x9	RX/TX DSA	NA

下表介绍了 Burst Writes 的 Operand 格式：

Word0[31:16]	Word0[15:0]	Word1[15:0]	Word2[27:24]	Word2[23:16]	Word2[15:8]	Word2[7:0]	
Opcode:11	Num of bytes to write	Start address	Byte 4	Byte 3	Byte 2	Byte 1	…

下面针对两个例子进行解释说明：

```
14960  #START: Doing RX IMD vs DSA improvement Analog Writes
14961  #
14962  0x00000000; #Write
14963  0x700F0011; #PAGE: rx_ec_i=0xf;     Address(0x11[3:0])
14964  0x00000000; #Write
14965  0x700F0011; #
14966  0x00000000; #Write
14967  0x70000B33; #
14968  0x00000000; #Write
14969  0x70000B32; #
14970  0x00000000; #Write
14971  0x70080B33; #
14972  0x00000000; #Write
14973  0x70200C91; #
14974  0x00000000; #Write
14975  0x70000011; #
14976  0x00000000; #Write
14977  0x70000011; #PAGE: rx_ec_i=0x0;     Address(0x11[3:0])
```

Figure 4. Opcode & Operand 解析例 1

如图 4 的 14962-14963 行所示，host 软件首先解析到 14962 行的 bit[31-16]=0，代表需要进行寄存器写入，则 14962 行的剩余的 4 byte 不需解析。第二部解析到 14963 行，bit[31-28]=7 代表 MSB 为 7，bit[27-24]=0 代表 LSB 为 0，bit[23:16]=0x0f 代表写入值为 0x0f，bit[15:0]=0x11 代表写入地址为 0x11。

```
3020  #Read   chip_id=0x77d;  Address(0x4[7:0],0x5[7:0],0x5[7:0],0x6[7:0])
3021  #
3022  0x00010006
3023  0x00020000; #ReadCheck
3024  0x70100006; #
```

Figure 5. Opcode & Operand 解析例 2

如图 5 的 3022 行所示，host 软件首先解析到 3022 行的 bit[31:16]=0x1，代表需要进行寄存器读取，3022 行的 bit[15:0]=0x6 代表需要读取的寄存器地址为 0x6，bit[]23:16]=0x10 代表预期回读值为 0x10。如果从地址 0x6 的寄存器回读值为 0x10，则软件返回通过；如果地址为 0x6 的寄存器回读值不是 0x10，则软件返回失败，并返回实际回读值。

3.4 初始化函数

如本文的 3.2 节所讲，AFE7769 约束了初始化流程，包含上电时序要求，外部信号配合要求，JESD 建链流程，由两种方法用软件实现 AFE7769 初始化。

方法一（不推荐）：如 3.3.1 节中描述，TI 提供 Raw Log Format，由用户自行开发针对关键字的解析代码，映射到底层驱动中。

方法二（推荐）：使用 TI 提供的初始化函数(afeDeviceBringup)，该函数包括解析 hex 配置文件内的 opcode 和 oprand 功能，由底层驱动完成初始化的参数和寄存器配置。

Figure 6. Device Bring Up 函数列表

4 动态 API

在器件初始化完成后，用户需要在不复位的情况下针对器件的工作模式进行配置，如修改输出频点，修改 AGC 门限等。TI 在操作底层寄存器的基础上，提供给用户动态 API，用户只需提供 API 的几个入参，由 API 进行运算并进行对应的寄存器写入。TI 提供的动态 API 种类非常丰富，易于用户对器件进行维护，主要分为两大类：

1. 参数获取：用户在软件初始化完成后，读取相应的器件及软件参数
2. 实际器件交互：

- 器件业务控制功能 API：对器件的工作模式进行控制，如打开/关闭通道，修改频点等
- 器件状态监测功能 API：对器件状态进行回读，如读取 PLL 锁定状态，JESD 链路状态
- 器件调试接口 API：业务场景下不涉及，器件出现异常时需要调用，如 Serdes 自环，JESD 发固定数的功能

4.1 调试初期推荐集成的动态 API

如上文所述，TI 提供了非常丰富的动态 API，这些 API 会覆盖极大部分场景下用户所需的控制需求，我们强烈建议用户将所有 API 集成到 host 软件中去。在特殊情况下，软件工程师的集成 API 的时间有限，为了减小软件开发压力，同时也要保证项目正常调试，我们推荐用户在项目前期最低限度要保证以下动态 API(2.2.1 节中叙述的 API)集成到 host 软件中。请注意，此列表只保证特定应用场景的调试需求，如在调试过程中发现列表外额外的 API 需求，请软件进行集成。

	File name	Functions
1	tiAfe77D_agc.c	agcAlcConfiguration bigStepAttackConfig bigStepDecayConfig smallStepAttackConfig smallStepDecayConfig configDgcCoarseFine enableDgc enableInternalAgc extLnaControlConfig freezeAgc getCurrentRxAttenuation setDefAttenuation setMaxMinAttenuation
2	tiAfe77D_calibrations.c	除 loadRxDsaPacket, loadTxDsaPacket, readDsaPacket 的所有 API
3	tiAfe77D_controls.c	除 sleep 相关 API，getOptimalRxPointers，maskPllAlarmsToPin，maskSpiAlarmsToPin，overrideAlarmPin，overrideDigPkDetPin，overrideRelDetPin 的所有 API
4	tiAfe77D_dpdCfr.c	所有 API 需集成
5	tiAfe77D_fb.c	fbDataMemCapture getFbDsa readFbNco readFbPeak readLowIfNcoFb setFbDsa updateFbNco
6	tiAfe77D_jesd.c	所有 API
7	tiAfe77D_pap.c	如果用户不启用 PAP 功能，不需集成； 如果用户启用 PAP 功能，需集成所有 API

8	tiAfe77D_rx.c	所有 API
9	tiAfe77D_serdes.c	所有 API
10	tiAfe77D_serdes.c	所有 API

5 参考文献

1. afe77xxDCLibsDocumentation 7/19/2022

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265