AN12758

使用 i.MX RT1010 FlexIO 模拟 I²S

Rev. 0 — May 2020

Application Note

目录

1 介绍

本应用笔记主要介绍如何使用 FlexIO 模块模拟 I2S 接口进行音频数据的发送和接收。使用 FlexIO 模块模拟的 I2S 接口可以代替传统的 I2S / SAI 外设来传输音频数据，节省 CPU 资源。

i.MX RT1010 处理器基于 ARM Cortex-M7 平台，具有很高的 CPU 性能和最佳的实时响应，且拥有丰富的外设资源。本篇应用笔记通过一个简单的例程来演示 FlexIO 模块模拟 I2S 总线接口的过程，并得到了验证。

本应用笔记中的例程使用模拟的 I2S 接口将音频数据接收到 sram 缓冲区中进行处理，再将音频数据传输到音频播放设备，实现音频数据的数字回环，音频数据的处理都基于 i.MX RT1010 EVK 板上的 Codec 芯片实现。

2 硬件平台

FlexIO 模拟 I²S 接口的例程基于 i.MX RT1010 EVK 板，实际硬件平台如 图 1 所示。

图 1. 硬件平台

为了例程能够成功演示，需要注意以下几点：

- 音频扬声器的数字接口插入 RT1010 EVK 板的 J11 端口。
- 将 ISP 拨码开关 SW8 更改为 0b0010。
- 将 USB 插到板上的 J9 口进行供电。
- 将 J1-3 和 J1-4 引脚用短路帽连接。

对 i.MX RT1010 EVK 板需要进行修改：

- 去掉板上的 R85，R87，R88，R20 这几个电阻。
- 将 FLEXIO 用到的引脚按如下关系进行硬件连接：

表 1. FLEXIO 引脚连接

引脚名称	板子位置	连接	板子位置
RX_DATA	J54-2	<------>	U10-16
TX_DATA	J26-8	<------>	U10-14
SYNC	J26-6	<------>	U10-13
BCLK	J26-4	<------>	U10-12

3 应用

本节介绍了使用 FlexIO 模拟 I²S 接口的一些设计要点，重点是 FlexIO 模块配置介绍。

3.1 系统架构

在本应用笔记中，例程的整个架构和 I²S 接口的连接关系如图 2 所示。i.MX RT1010 作为 I²S 的从机设备，利用 FlexIO 模拟了四根引脚，它们分别是 I²S 接口的 SDA_RX、SDA_TX、FSYNC（WS）以及 BLCK。WM8960 编解码器作为 I²S 的主机设备，从麦克风接收音频信号，然后通过 SDA_RX 引脚将音频数据传输到 RT1010。RT1010 将接收的音频数据处理后，再通过 SDA_TX 引脚发送给 WM8960，WM8960 将 PCM 数据转成模拟信号交由扬声器播放。此外，RT1010 通过 I²C 接口对 WM8960 进行初始化配置操作。

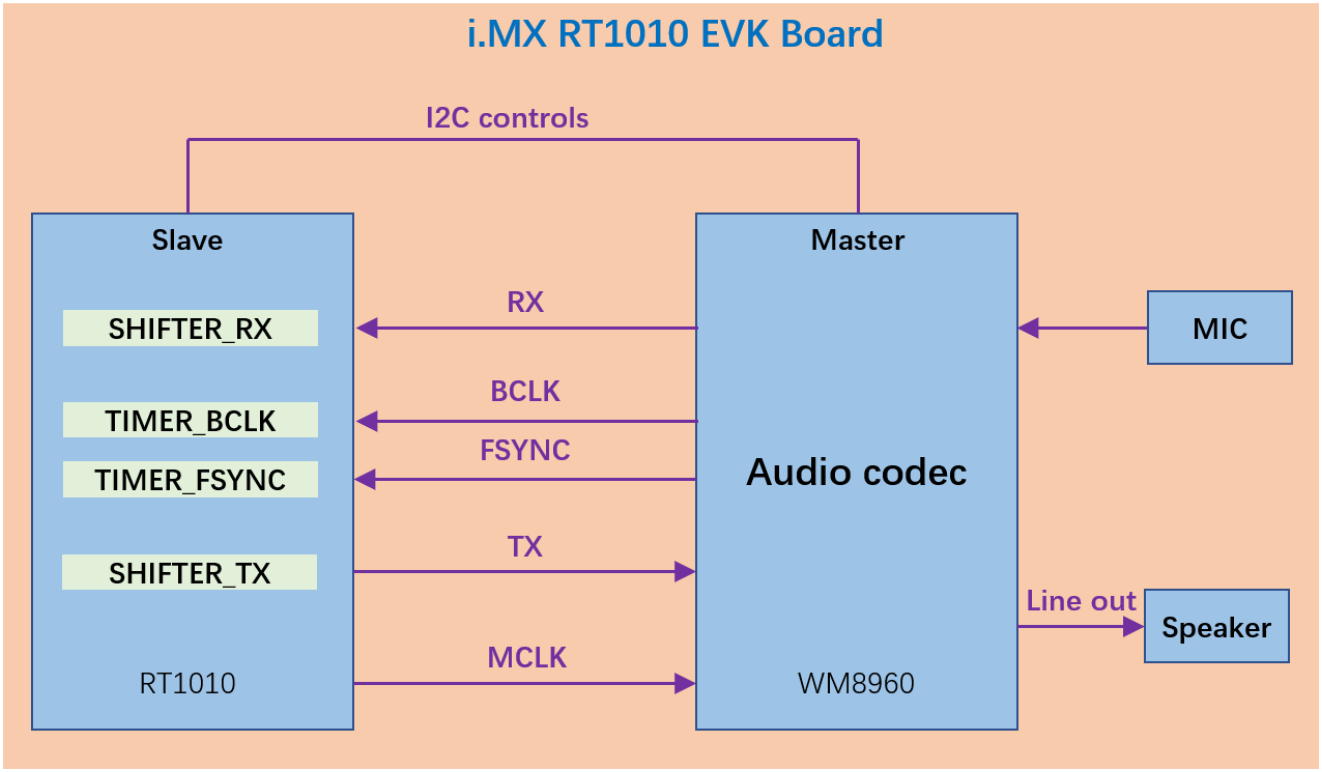

图 2. 系统架构

3.2 时钟及采样频率配置

与 MCU 做为 I²S 主机不同的是，当 MCU 作为 I²S 从机时，FSYNC 和 BLCK 信号是由 WM8960 产生的，因此第一步要把 WM8960 配置成 master 模式。WM8960 的 MCLK 时钟可以由 RT1010 提供，也可以由 WM8960 提供。本应用中，我们利用 RT1010 产生 MCLK 信号比较方便，并把 MCLK 配置为 6.144 MHz。

常见的 I²S 采样频率通常有两组，如下所示。本应用中，I²S 采样频率配置为 16 kHz，同样 WM8960 的采样频率也需要被配置为 16 kHz。一旦 I²S 的采样频率确定了，那 FSYNC 信号的频率也就确定了，也为 16 kHz。

表 2. I²S 采样频率

典型采样频率 (Hz)	典型采样频率(Hz)
11025	8000
22050	16000

Table continues on the next page...

表 2. I²S 采样频率 (续)

典型采样频率 (Hz)	典型采样频率(Hz)
44100	24000
—	32000
—	48000

为了能够模拟 I²S 接口作为 slave 模式，FlexIO 的时钟配置是有一定要求的，它必须是采样频率的某个倍数，这样 FlexIO 的定时器才能匹配这个频率。在本应用中，FlexIO 时钟被配置为 6.144 MHz，FlexIO 模拟的 I²S 接口基于这个时钟来实现。

下一步，需要对 WM8960 进行一系列配置，使其从板载麦克风采集的音频信号在传输给 RT1010 时，是 16 KHz，32 bit 的立体声（左右声道）音频数据。

接着，我们需要计算并配置 FlexIO 的各个引脚时钟，BCLK 时钟信号是由 WM8960 产生的，MCLK 作为 WM8960 的参考时钟，经过分频得到 BCLK 时钟，过程如下：

- FSYNC = 16 KHz
- MCLK = 6.144 MHz
- BCLK = FSYNC × 声道数 × 声道位宽 = 16 KHz × 2 × 32 = 1.024 MHz
- BCLK_DIV （BLCK 分频系数） = MCLK/BCLK = 6.144 MHz/1.024 MHz = 6

例 1 给出了 WM8960 编解码芯片的部分配置代码：

例 1

```
wm8960_config_t wm8960Config = {
    .i2cConfig         = {.codecI2CInstance =
BOARD_CODEC_I2C_INSTANCE,.codecI2CSourceClock                          =
BOARD_CODEC_I2C_CLOCK_FREQ},
    .route             = kWM8960_RoutePlaybackandRecord,
    .rightInputSource  = kWM8960_InputDifferentialMicInput2,
    .playSource        = kWM8960_PlaySourceDAC,
    .slaveAddress      = WM8960_I2C_ADDR,
    .bus               = kWM8960_BusI2S,
    .format            = {.mclk_HZ = 6144000U,
    .sampleRate        = kWM8960_AudioSampleRate16KHz,
    .bitWidth          = kWM8960_AudioBitWidth32bit},
    .master_slave      = true,
};
codec_config_t boardCodecConfig = {
        .codecDevType      = kCODEC_WM8960,
        .codecDevConfig    = &wm8960Config
};
/* Init codec */
CODEC_Init(&codecHandle, &boardCodecConfig);
```

3.3 FlexIO 模拟

FlexIO 是一个具有多种功能，高度可配置的模块，具有以下特点：

- 支持多种串行/并行通信协议的仿真，例如 UART，I²C，SPI，I²S 等。
- 灵活的 16 位定时器，支持多种触发，复位，使能和禁止条件。
- 可编程逻辑块，允许在片上实现数字逻辑功能以及内部和外部模块的可配置交互。
- 可编程状态机，用于不依赖 CPU 而实现基本的系统控制功能。

在 i.MX RT1010 上，FLEXIO 共有 27 个引脚。本应用一共使用了 4 个 FlexIO 引脚（FlexIO03 引脚，FlexIO21 引脚，FlexIO22 引脚，FlexIO26 引脚）分别模拟 I^2S 接口的 SDA_RX 引脚，BCLK 引脚，FSYNC 引脚和 SDA_TX 引脚。图 3 显示了 FlexIO模拟 I^2S 接口的内部连接。 Timer0 和定时 Timer2 连接的 FlexIO 引脚分别用于产生 BCLK 信号和 FSYNC 信号，而 SHIFTER0和 SHIFTER2 的 FlexIO 引脚分别用于产生 SDA_TX 信号和 SDA_RX 信号。下面分别介绍 SHIFTER 和 Timer 使用方法和配置。

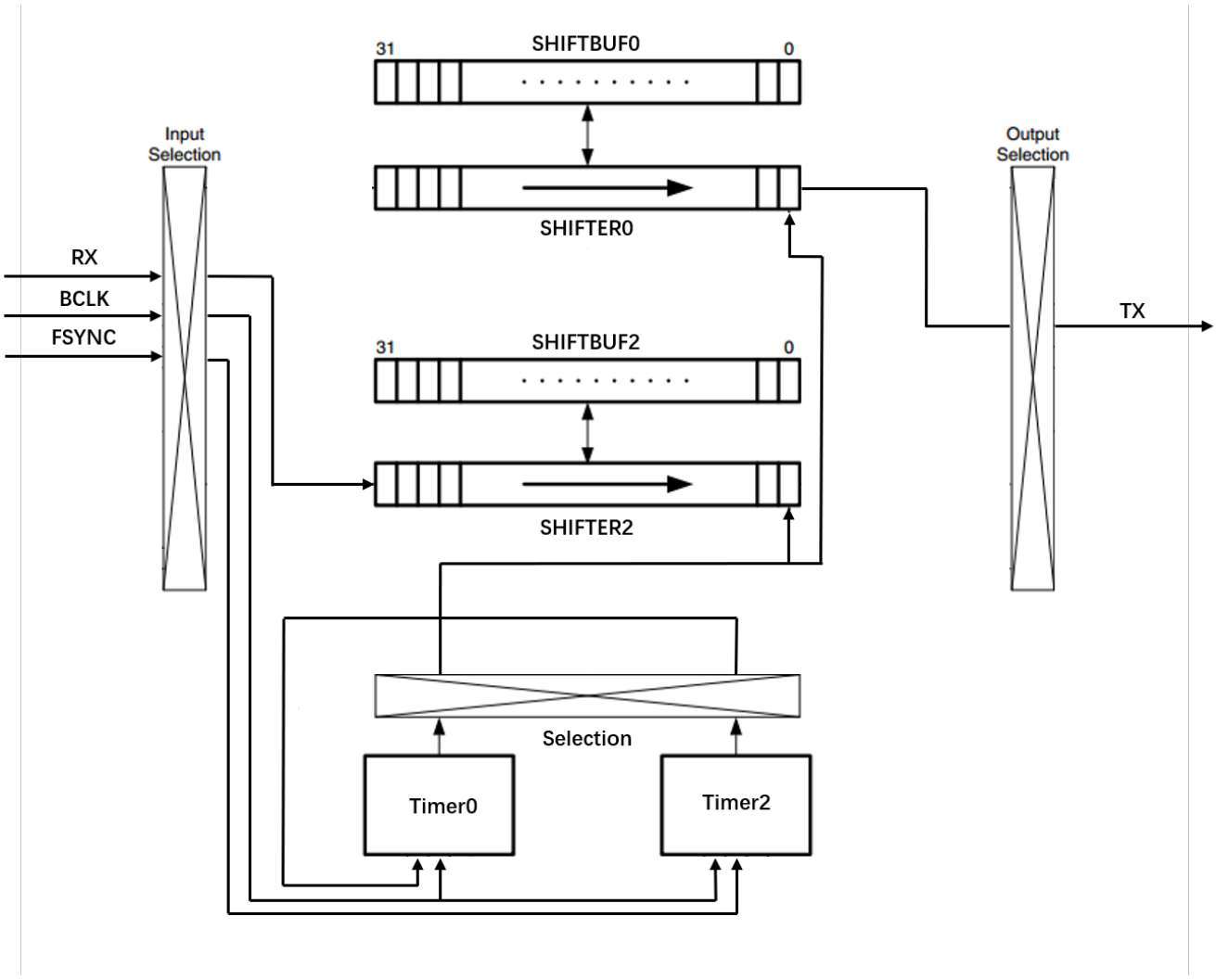

图 3. FlexIO 内部连接框图

通过配置 SHIFTCTL 寄存器，可以将 SHIFTER 配置成 6 种模式，本应用中只需要关注 Transmit 模式和 Receive 模式就行。

先对 SHIFTER0 进行配置。将 SHIFTER0 配置为 Transmit 模式，SHIFTER0 在移位时钟的上降沿将 SHIFTBUF0 中的数据从 TX 引脚输出。SHIFTER2 被配置为 Receive 模式，同样 SHIFTER2 在移位时钟的下降沿从 RX 引脚上获取数据并放入 SHIFTBUF2。当数据从 SHIFTER 加载到 SHIFTBUF 寄存器中或数据从 SHIFTBUF 寄存器加载到 SHIFTER 中时，如果已经将 SHIFTER 状态标志位（SHIFTSDEN SSDE）置 1，就可以产生一个 DMA 请求。整个 SHIFTER 的微体系结构如 图 4 所示，它充分展示了 SHIFTER 中各个模块之间的关系以及 IO 引脚输入输出的关系。

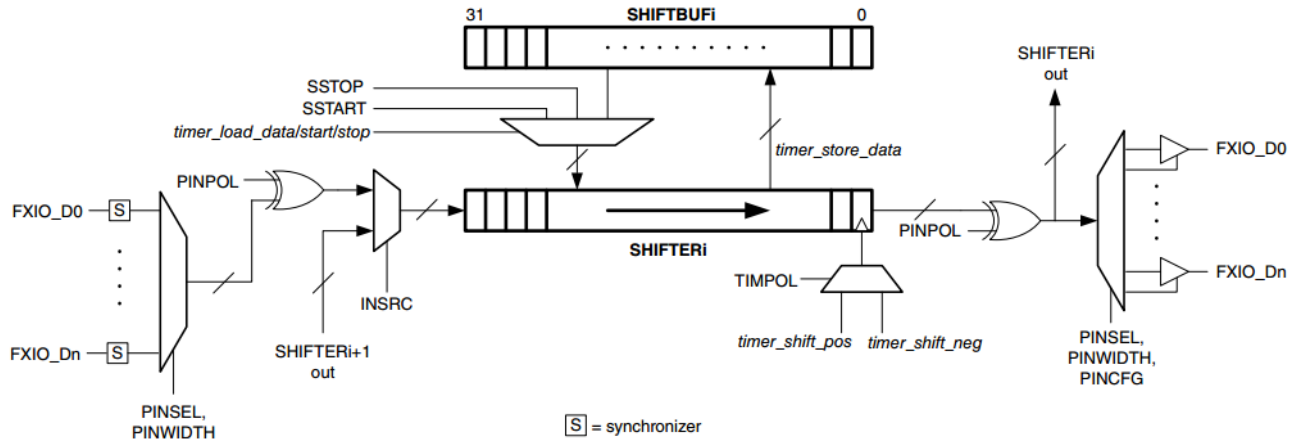

图 4. SHIFTER 模块架构图

下面需要对 Timer0 和 Timer2 进行配置。

当 Timer2 检测到 FSYNC 的上升沿时使能，当其检测到 trigger 事件的下降沿时禁用。Timer0 的使能发生在 BLCK 的上升沿以及 Timer2 的 trigger 事件检测为高电平时，Timer0 的关闭发生在其自身的比较事件产生时。此外，Timer0 和 Timer2 的时钟状态需要被初始化为逻辑 1。Timer2 的时钟模式需要被配置为 16 位计数模式，其比较事件产生的值设置为 0，并使用 FSYNC 引脚的输入作为递减量，使用 BCLK 引脚的输入作为 trigger 事件。Timer0 的时钟模式需要被配置为 16 位计数器，并使用 BCLK 引脚的输入作为递减量。 在此应用中，Timer2 的比较事件产生的值设置为 127（32 * 4-1），这是根据 64bit 一帧的音频数据位宽长度计算得到的。

下面给出了详细的寄存器配置：

- FIEXIO01.SHIFTCTL[0] = 0x00031A02
- FIEXIO01.SHIFTCTL[2] = 0x00800301
- FIEXIO01.TIMCTL[0] = 0x0B401583
- FIEXIO01.TIMCTL[2] = 0x2A401683
- FIEXIO01.TIMCFG[0] = 0x00202500
- FIEXIO01.TIMCFG[2] = 0x00206400
- FIEXIO01.TIMCMP[0] = 0x0000007F
- FIEXIO01.TIMCMP[2] = 0x00000000

3.4 音频数据流处理

MCU 在同时接收和发送 PCM 数据并进行播放的应用场景中，容易出现播放音乐卡顿的情况，为了避免出现这种卡顿，一个好的传输机制是必不可少的，图 5 给出了一种处理 PCM 数据的方法。每当 FlexIO 中的 SHIFTER 有 DMA 请求产生时，应立即从 SHIFTBUF 读取或写入音频数据。图 5 中两个缓冲区用于 PCM 数据的发送和接收，这两个缓冲区构成了 ping-pong buffer。PCM 数据帧的位宽为 64 位（左通道和右通道），每个缓冲区被设置为 256bit，即存放 4 帧 PCM 数据， 每当一个缓冲区中的 PCM 数据接收满或发送时，下一个缓冲区将立即开始接收或发送。

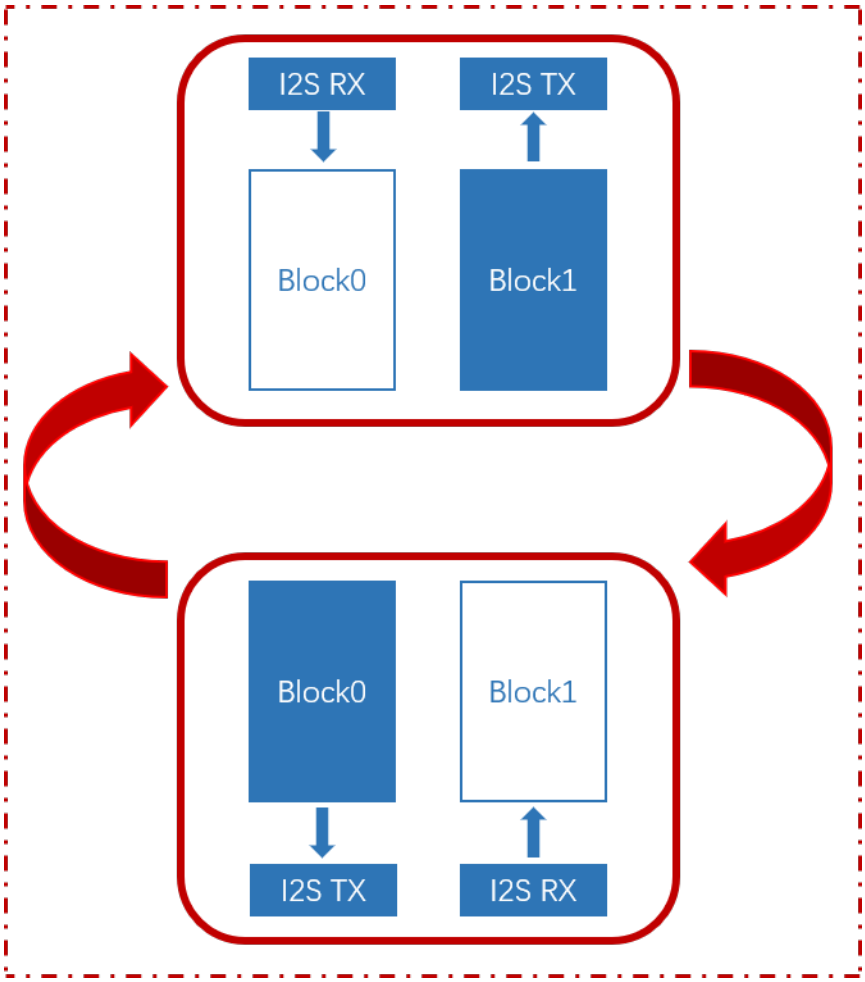

图 5. PCM 数据处理

4 总结

本应用笔记介绍了利用 i.MX RT1010 的 FlexIO 模块来模拟 I²S 接口的方法和例程。当 CPU 硬件资源不足，FlexIO 可以模拟 I²S 接口以很好地传输音频数据。使用 FlexIO 模块模拟 I²S 接口时，需要注意两个注意事项：

- 由于 FlexIO 同步延迟，当 FlexIO 用来模拟 I²S 从设备时，串行数据的输出有效时间是 FlexIO 时钟周期的 2.5 倍。 因此，I²S 的 BCLK 最大时钟频率应当是 FlexIO 时钟频率的六分之一。
- FlexIO 是一个功能强大，非常灵活的模块，除了本文给出的 Timer 和 SHIFTER 的配置外，读者也可以利用其它配置模拟出 I²S 接口。

5 参考资料

- *i.MX RT1010 Processor Reference Manual (Rev. B, 07/2019)* (document IMXRT1010RM)
- *Emulating the I2S Bus Master with the FlexIO Module* (document AN4955)
- WM8960 Data Sheet

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and 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: May 2020

Document identifier: AN12758

arm