NXP Semiconductors
Application Note

Document Number: AN12202
Rev. 0, 11/2018

在S32K148上使用同步音频接口(SAI)

作者：NXP半导体公司

URL：https://www.nxp.com/docs/en/application-note/AN12202.pdf

目录

1 产品介绍

S32K148的同步音频接口(SAI)模块适用于各种音频处理的应用需求。该模块高度可配置，且支持不同的音频格式，如I2S、Codec/DSP和TDM。

本应用笔记主要描述了SAI模块的不同音频格式的概述、模块介绍和用例实现。

2 音频总线的拓扑结构

音频总线（可用于I2S、Codec/DSP模式、TDM等）被设计为尽量减少所需的针脚数量，由一个串行数据线、一个通道选择和一个时钟线组成的三线串行总线。由于发送端和接收端具有相同的时钟信号用于数据传输，因此涉及两个不同的角色：主机和从机。

负责提供时钟和通道选择线的发送端设备或接收端设备被定义为主机，无论是否连接多个发送端和接收端，总线中必须只有一个接收端，其余连接到总线的设备被命名为从机设备。下图描述了通用音频总线连接上的主从角色。

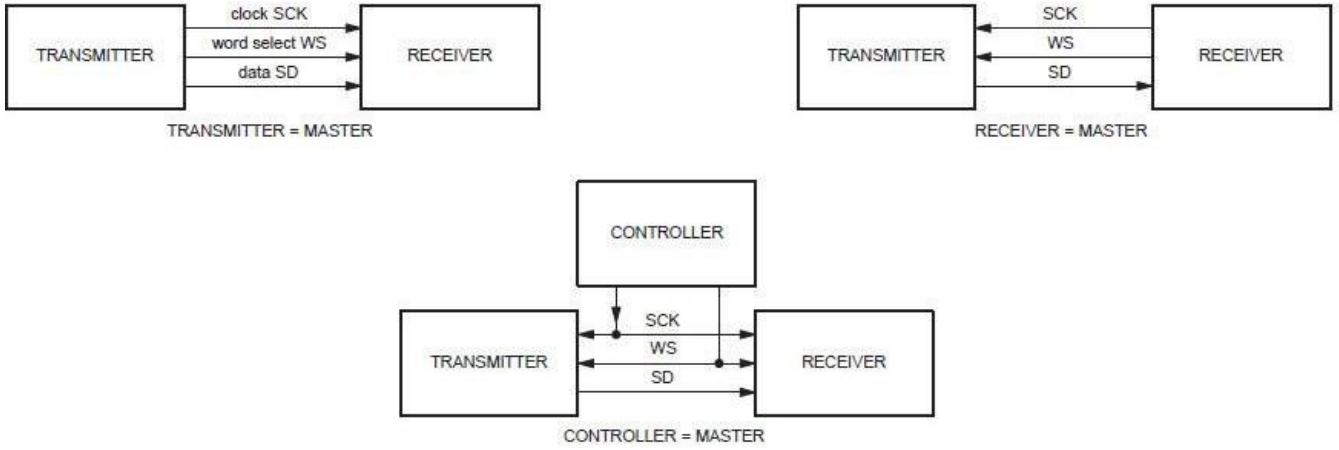


图1.音频总线连接中的主角色和从角色

3. 音频格式

在介绍SAI模块及其功能之前，了解与数字音频信号相关的不同音频格式和概念是很重要的。虽然某些音频集成电路/解编码器可能对本节中介绍的某些音频格式有不同的名称，但根据其特征而不是其名称来识别每种格式是很重要的。

3.1. I2S

串行协议(I2S)是一种专门针对数字音频应用的用于IC制造商和音频处理单元之间的标准化通信。I2S总线有三条线路：

- 连续串行时钟(SCK)，比特时钟(BCLK)
- Word选择(WS)、帧同步(FS)、单词时钟(WCLK)、左右时钟（LRCLK)
- 串行数据(SD)、串行数据输出/输入（SDOUT、SDIN）

串行时钟(SCK)，也被称为位时钟(BCLK)，是用于为每个音频比特提供时钟参考。Word选择(WS)、帧同步(FS)或单词时钟(WCLK)表示正在传输的通道：当此线路设置为“0”时，正在传输通道1（左），当设置为“1”时，则传输通道2（右）。WS行在传输MSB之前提前一个时钟周期发生改变。

该信号的频率对应于音频采样速率的频率。串行数据(SD)总是首先传输MSB数据（因为发送端和接收端可能有不同的字长度）。数据的范围从8位到32位。I2S格式示意图如下图所示。

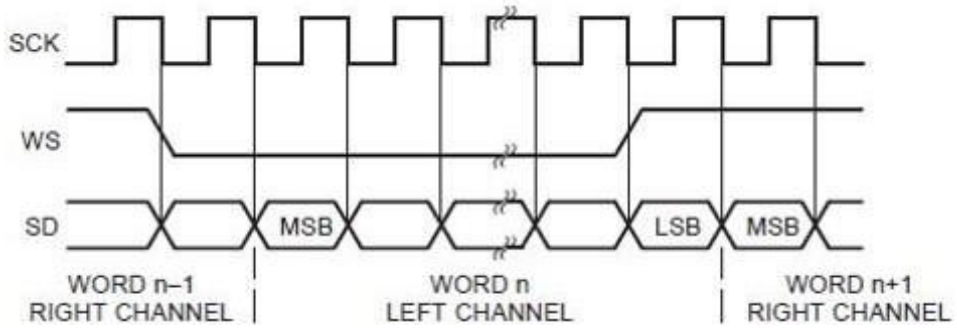


图2.I2S原理图

3.2. Codec模式（左/右对齐）

Codec模式不同于I2S协议，WS信号和SD信号同步，即在同一时刻发生改变(在I2S中SD信号延迟WS信号一个时钟周期)。此外，与I2S相比，WS信号则相反，即：当WS设置为“0”时，发送右通道数据，当设置为“1”时，发送左通道数据。

下面介绍Codec模式的两种模式。

3.2.1. 左对齐（MSB对齐）

对于左对齐，也称为MSB对齐，当有数据帧需要发送时，WS将发生更改。串行数据在左边是有效的，这意味着如果WS的半周期是32位长数据，前24位将用于音频，其余8位必须设置为零。

下图描述了左对齐的格式。

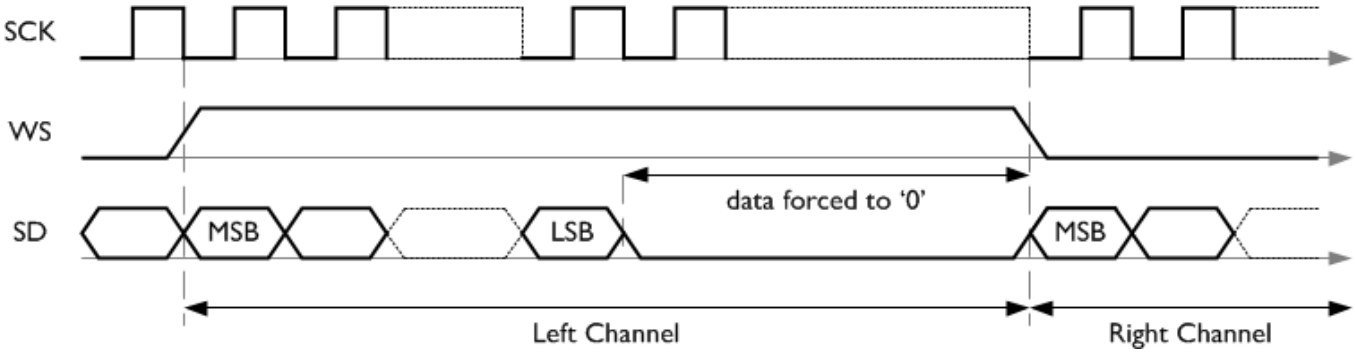


图3.左对格式

3.2.2. 右对齐（LSB对齐）

对于右对齐，也称为LSB对齐，当有数据帧需要发送时，WS将发生更改。串行数据在右边是有效的，这意味着如果WS的半周期是32位长数据，只有24位用于音频数据，前8位则必须设置为零。

由于串行数据以MSB格式传输，LSB位在WS改变其状态之前与最后一个位时钟周期相匹配。

下图描述了右对齐的格式。

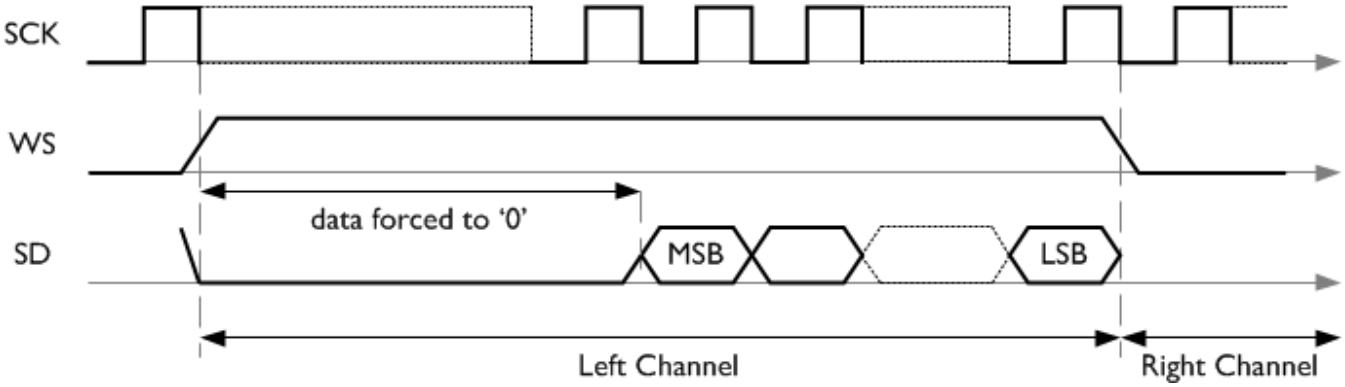


图4.正确正确的格式

3.3. DSP模式

DSP模式类似于左对齐Codec格式，但WS的宽度可能取决于IC架构（最小允许值为1位时钟）。由于WS不是50%占空比信号，WS的上升沿信号表示音频数据的开始，首先是左通道数据，接着是右通道数据。WS的频率仍然定义了音频采样率。

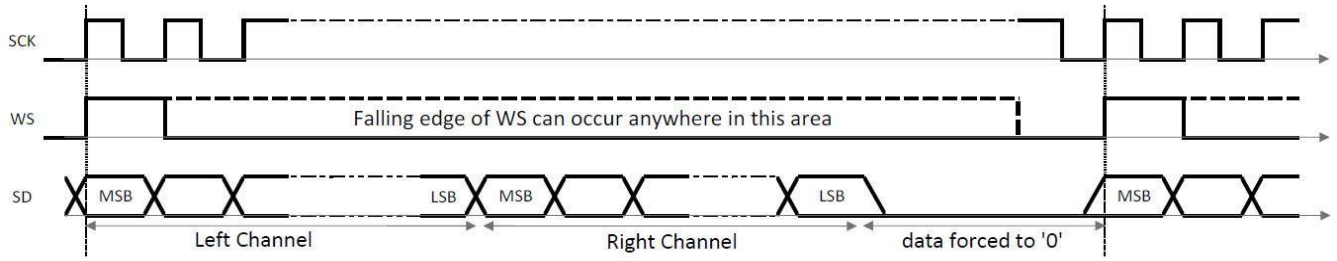


图5.DSP的格式

在一些DSP模式中，数据延迟WS一个时钟。下图说明了这种DSP模式，其中WS宽度只有1个时钟周期长度。

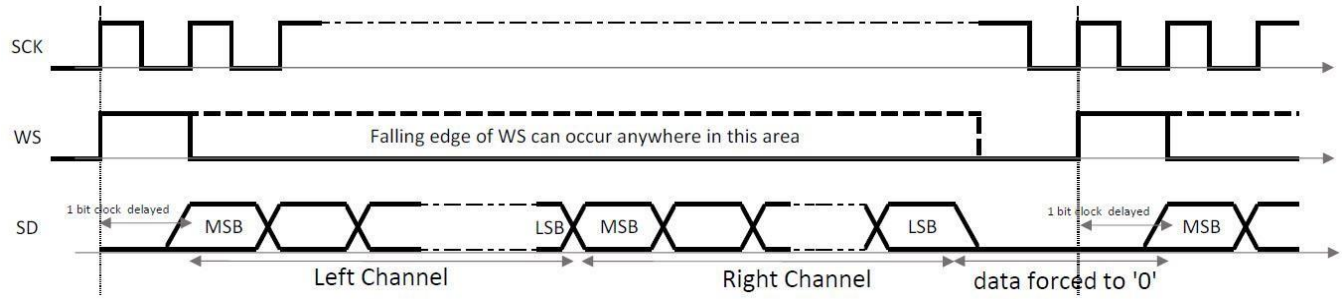


图6.串行数据延迟一个时钟的DSP模式

附注

有些IC可能需要在每个左右通道之间放置0强制数据，而不是将其放置在最后。在从机设备的数据表中了解集成电路的需求是很重要的。

3.4. 时分多路复用(TDM)

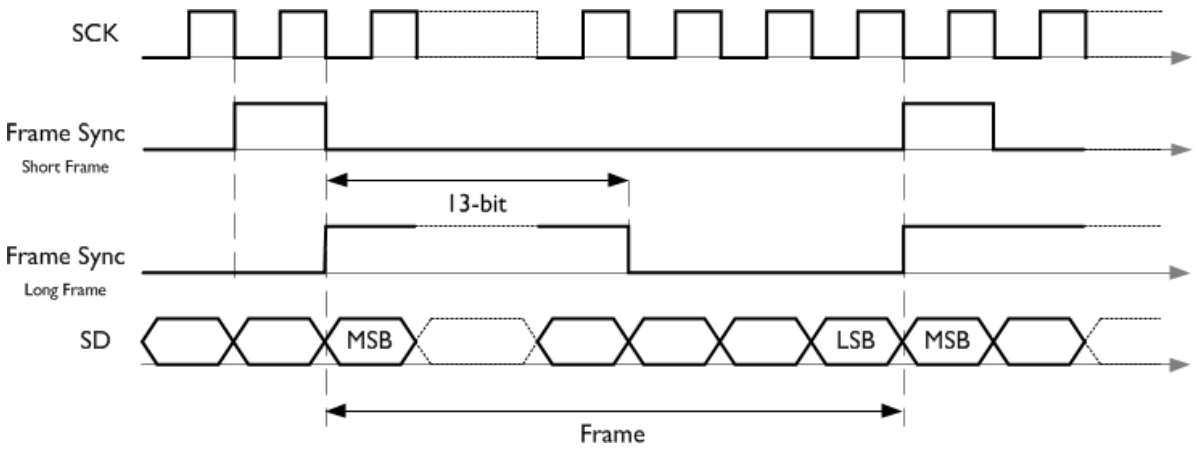
对于上述的音频格式，只能在单个WS周期内发送2个频道，但是，对于TDM格式，可以在一个WS周期间内发送2个以上的频道。在TDM格式中，WS的宽度只有1个时钟周期长度。

当多个从机设备连接到总线时，通常使用TDM模式，其中，主机使用相同的同步信号（WS）向从机发送数据。每个从机设备可以配置偏移参数来获取自己相对应的通道数据。

3.5. PCM

在PCM模式中，一个同步周期内只传输一个通道。同步模式有两种类型：短帧和长帧。对于短帧同步模式，“WS/帧同步”的下降沿表示串行数据的开始。WS/帧同步信号宽度总是为一个时钟周期长度。对于长帧同步模式，“WS/帧同步”的上升沿表示串行数据的开始。WS/帧同步信号宽度保持13个时钟周期。下图显示了两种同步模式的PCM格式。

图7.短同步模式的PCM格式



4. SAI模块概述

由于同步音频接口(SAI)支持全双工串行接口和多个音频协议，如I2S、AC97、TDM和Codec/DSP接口，本节重点介绍SAI特性/组件，以及如何配置不同的音频格式。

时钟信号、WS/帧同步信号和数据信号分别标识为BCLK、SYNC和Dn。

有关SAI模块体系结构的详细信息，请参阅设备参考手册中的SAI章节。

4.1. SAI结构

S32K148中的SAI模块包括独立的发送端和接收端。每个逻辑包括8个Word（每个32位）的FIFO支持，具有DMA支持，以便有效地提高音频处理性能。S32K148中有两个SAI模块：SAI0和SAI1，SAI0支持4个数据行，而SAI1只支持一条数据行。

图中显示了SAI的功能框图。

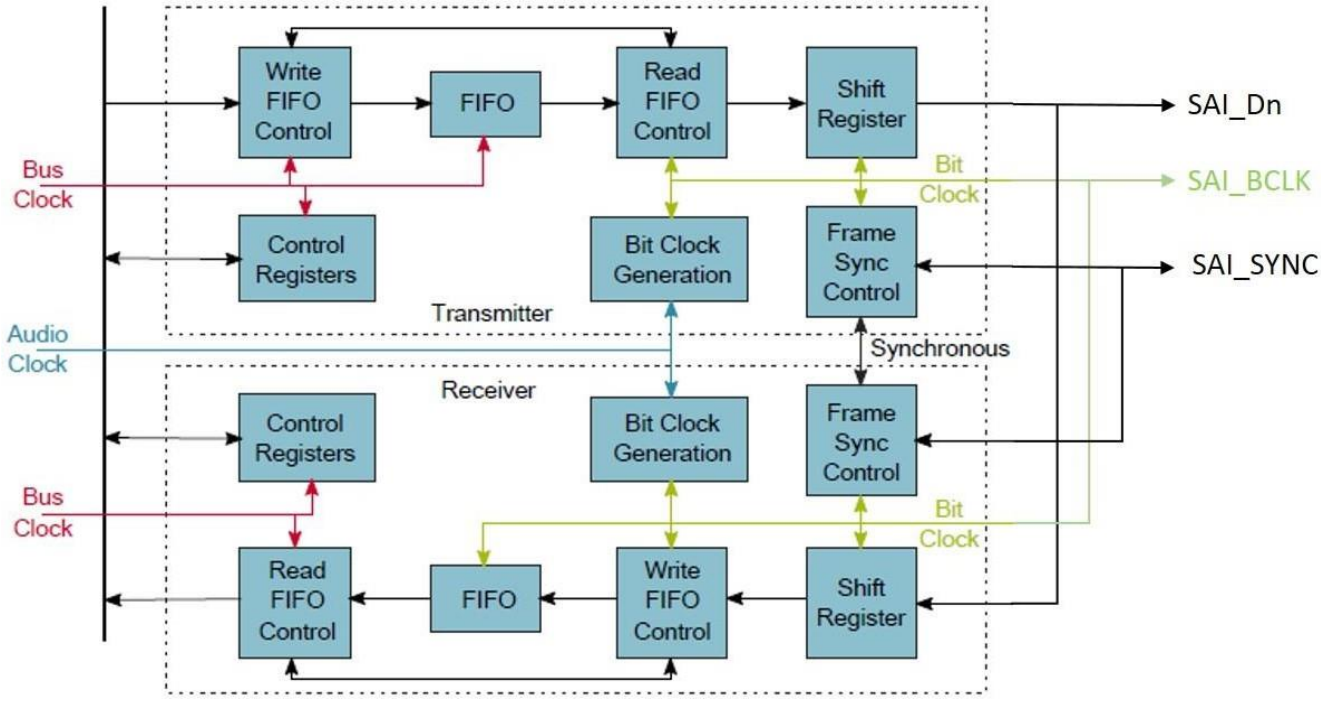


图8.S32K148上的SAI方框图

由于发送端和接收端的通信线连接在一起，一次只能使能一条。请注意，发送端和接收端的寄存器之间是相互独立的，因此一个寄存器将被称为SAI_xCRn，分别为SAI_TCRn或SAI_RCRn。

4.2. SAI时钟

在访问任何SAI寄存器之前，必须在主机配置和从机配置的PCC_SAIx寄存器中使能模块时钟。

4.2.1. 主机的时钟配置

当SAI配置为Master(SAI_xCR4[FSD]=1和SAI_xCR2[BCD]=1)时，将基于时钟源参考和分频器配置字段在内部生成BCLK和SYNC信号。在S2K148中，SAI最多可以使用四种不同的时钟源：模块时钟（总线时钟）、外部MCLK时钟、系统振荡器时钟分频器1(SOSCDIV1)和外部MCLK时钟。下图显示了这些时钟选项。

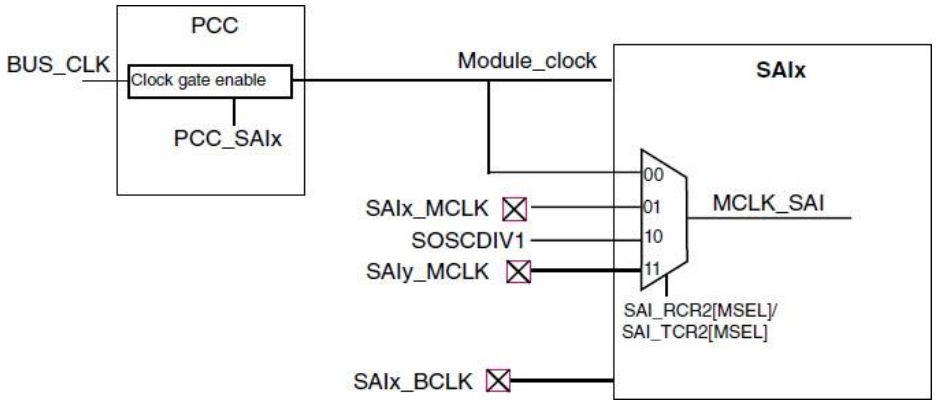


图9.SAI时钟源选项

附注

当SAIx_MCLK引用该MCLKpin，SAIy_MCLK引用其他MCLKpin。

一旦选择了MCLK_SAI，此时钟将用于计算比特时钟的频率，如下所示：

$$BCLK = \frac{MCLK\_SAI}{(DIV + 1) \cdot 2}$$

其中，DIV是来自SAIx_TCR2或SAIx_RCR2的8位字段，这取决于是否配置了发送端或接收端。

由于普通音频频率基于帧同步（采样率）频率，可以根据采样速率计算比特时钟：

$$BCLK = bits\ per\ channel \cdot number\ of\ channels \cdot sample\ rate\ freq$$

因此，DIV可以从采样速率频率中计算出来，如下图所示：

$$bits\ per\ channel \cdot number\ of\ channels \cdot sample\ rate = \frac{MCLK\_SAI}{(DIV + 1) \cdot 2}$$

$$DIV + 1 = \frac{MCLK\_SAI}{bits\ per\ channel \cdot number\ of\ channels \cdot sample\ rate \cdot 2}$$

$$DIV = \frac{MCLK\_SAI}{bits\ per\ channel \cdot number\ of\ channels \cdot sample\ rate \cdot 2} - 1$$

例如，对于I2S格式，通道数为2（左右），每个通道位为32，采样率可以为：8kHz、11.025kHz、16kHz、22.05kHz、32kHz、44.1kHz、48kHz或96kHz。假设MCLK_SAI时钟来自于总线时钟(SAI_xCR2[MSEL]=0)，且总线时钟已配置为40MHz，则DIV值为：

$$DIV = \frac{40\ MHz}{32 \cdot 2 \cdot sample\ rate \cdot 2} - 1$$

下表显示了一些主要样本速率值的不同DIV值。

表1.采样率和DIV值

Sample Rate (kHz)	Exact DIV value	Closet DIV value	Resulting BCLK frequency (MHz) (SAI clock as 40Mhz)	Real Sample Rate (BCLK / 64) in KHz
8	38.06	38	0.512	8.012
11.025	27.34	27	0.714	11.16
16	18.53	18	1.052	16.447
22.05	13.17	13	1.428	22.321
32	8.76	9	2.222	34.722
44.1	6.08	6	2.857	44.642
48	5.51	5	3.333	52.08
96	2.25	2	6.666	104.166

为了获得更准确的采样率频率，建议使用具有标准主时钟值的MCLK时钟。例如，如果MCLK设置为24.576MHz，则对于8、16、32、48和96kHz，将获得更准确的DIV值，如下所示：

表2.当MCLK为24.576MHz时，采样速率和DIV值

Sample Rate (kHz)	DIV value	Resulting BCLK frequency (MHz)
8	23	0.512

Sample Rate (kHz)	DIV value	Resulting BCLK frequency (MHz)
11.025	16.41497	0.7056
16	11	1.024
22.05	7.707483	1.4112
32	5	2.048
44.1	3.353741	2.8224
48	3	3.072
96	1	6.144

4.2.2. 从机的时钟配置

当将SAI配置为Slave(SAI_xCR4[FSD]=0和SAI_xCR2[BCD]=0)时，由于BCLK和SYNC将由主机生成，则无需配置SAI_xCR2[MSEL]和SAI_xCR2[DIV]值。

4.3. 同步模式

SAI模块可以配置为使用两种不同的同步模式进行操作：

- 同步模式：SAI发送端和接收端可配置为使用同步比特时钟和帧同步进行操作，并由启同一模块进行控制。（发送端和接收端均使用相同的SAI线路）
- 异步模式：SAI发送端或接收端可配置为相互异步操作。

虽然S32K148中的SAI模块实现不允许同时操作同一实例的发送端和接收端（由于发送端和接收端之间已经共享大多数线路），但建议在发送端或接收端寄存器（TCR2[SYNC]或RCR2[SYNC]）中设置同步模式，以便在配置为主机时正确生成SAI的内部比特时钟。

4.4. SAI配置

SAI模块允许配置一些音频设置，如下图所示。

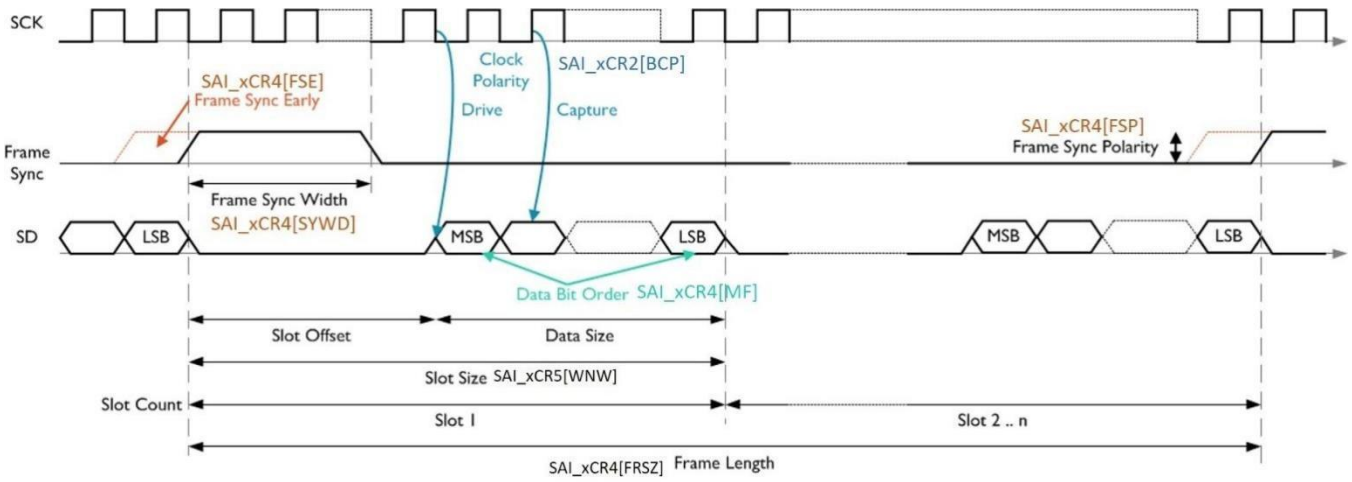


图10.音频设置和SAI字段

- 时钟极性(SAI_xCR2[BCP])：时钟线可以配置成两种不同的方式，具体取决于哪个边缘用于准备数据，哪个边缘用于采样/捕获数据。
- 早期帧同步(SAI_xCR4[FSE])：让同步信号领先数据信号一个位时钟周期称为早期帧同步。如果使能了早期帧同步，那么帧同步信号将提前数据信号一个位时钟周期。如果此特性被禁用，则两者同步。
- 帧同步极性（SAI_xCR4[FSP]）：此功能配置帧同步信号的有效极性。
- 帧同步宽度(SAI_xCR4[SYWD])：以“位时钟”为单位，此参数确定帧同步信号维持有效电平多少个位时钟。写入的值必须比比特时钟数少一个。
- 数据位顺序(SAI_xCR4[MF])：尽管大多数音频格式的数据位顺序为MSB，但如果需要，可以选择将此特性更改为LSB。
- 插槽大小(SAI_xCR5[WNW])：以“bit” 为单位。此参数配置每个通道数据的位数。所写入的值必须比每个通道位数少1。
- 插槽计数（SAI_xCR4[FRSZ]）：此功能可配置每个音频帧中的通道数。写入的值必须比帧中的通道数少1。
- 插槽偏移量：通过写入SAI_TDRn时来实现数据偏移。通常，插槽大小会受到左右移位数据“n”位以对齐到特定值的影响。

4.5. SAI FIFO和DMA/中断

SAI发送和接收通道都有8 word（32bit）的FIFO，可以通过读写SAI发送/接收data寄存器来对FIFO进行读写操作。

SAI_RCR1寄存器可以配置FIFO水印（小于或等于8）。此FIFO水印帮助用户设置一个特定的值，该值可以保证在DMA请求音频数据之前FIFO不会为空。

有两个不同的标志用于触发DMA请求：

1. FIFO请求标志：当前FIFO中数据深度分别低于传输FIFO和接收FIFO的水印配置。如果条目数高于/低于水印值，则会自动清除。下图显示了在某些条件下，发送端和接收端的FIFO请求标志值。

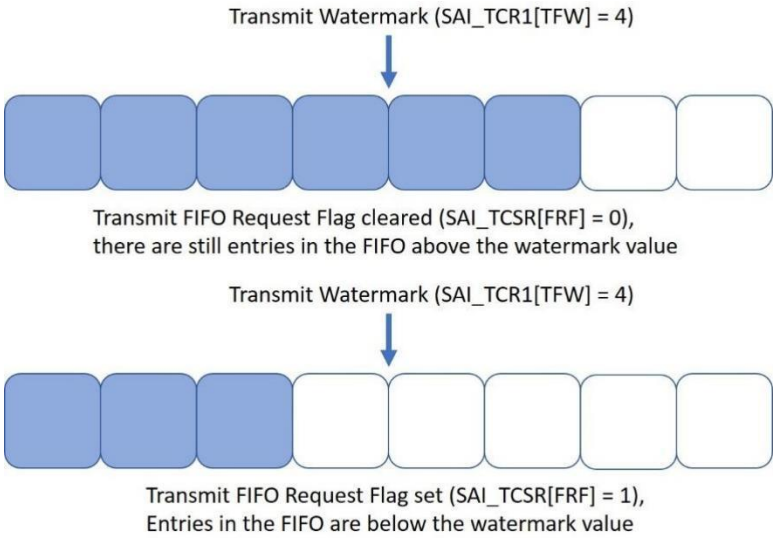


图11.发送端的FIFO请求标志

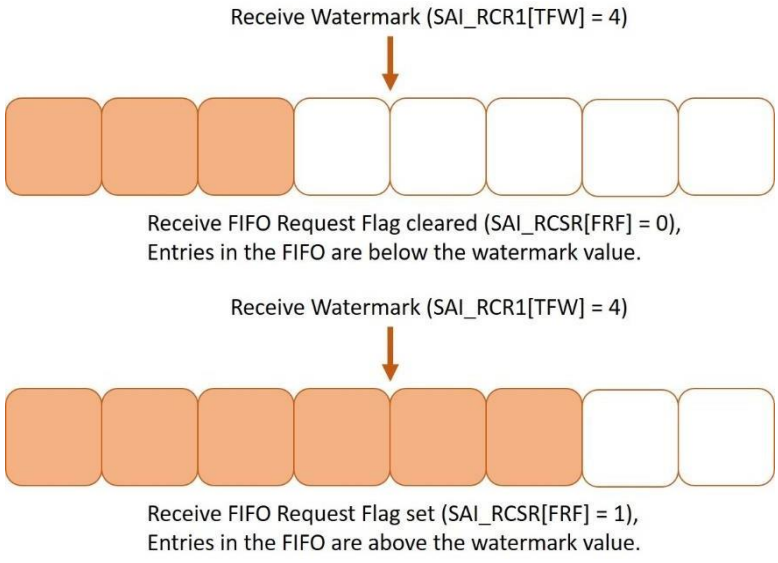


图12.接收端的FIFO请求标志

2. FIFO警告标志：如果FIFO为空（发送端）或满（接收端），如果上述情况消失则自动清除信号。

SAI发送端和接收端产生单独的中断和DMA请求，但支持相同的状态标志。参考手册中的中断和DMA请求部分详细描述了这些标志含义。

4.5.1. FIFO错误处理

每当检测到FIFO错误时，SAI模块将停止传输数据，除非设置了SAI_xTCR4[FCONT]位。为了继续发送/接收数据，必须清除SAI_xCSR[FEF]标志。

4.6. 屏蔽SAI通道

SAI模块允许用户通过设置传输掩码寄存器（SAIx_TMR）或接收掩码寄存器（SAIx_RMR）中的位来进行掩码/解除掩码通道。这个16位字段中的每个位表示帧中的一个通道。通过掩蔽（写0）通道位，可以从SAI的FIFO中发送/接收相应的通道。如果打开（写1）通道位，则不认为相应的通道数据将从FIFO中被读取/写入。

例如，如果需要以I2S格式传输音频数据，则需要两个通道，但如果只使用其中一个通道（例如左通道），则应清除位0（通道1-左），同时可以设置位1（通道2-右）。这样做之后，发送端FIFO只包含通道1的（左）的数据。如果通道2没有被屏蔽，那么FIFO将需要存储左右两个通道的数据，因此用户应该对正确通道对应的所有单词写零，因此如果没有正确设置通道掩蔽，则需要用户填充。

4.7. SAI初始化过程

下一节列出了配置SAI模块时所需的所有步骤。

1. 通过设置PCC_SAIx[CGC]位来打开SAI模块的时钟。
2. 在PORTx_PCR寄存器中配置SAI_BCLK、SAI_Dn和SAI_SYNC引脚功能。
3. 禁用发射器/接收端。
4. 重置发送端和接收端的FIFO。
5. 通过设置SAI_xCR2[SYNC]字段，配置SAI的发送端和接收端的同步模式。
6. 配置SAI_xCR2、SAI_xCR4、SAI_xCR5寄存器，以满足所需的音频格式的特性。
7. 通过设置/清除SAI_xCR4[FSD]和SAI_xCR2[BCD]位来配置主模式/从属模式。
8. 如果需要主模式，请按照SAI时钟节中的所述，调整时钟设置。
9. 在SAI_xMR寄存器中设置掩码/未屏蔽的所需通道。

10. 通过清除SAI_xCR3[TCE]中的相应位来禁用SAI数据行。
11. 如果使用DMA/中断，请在SAI_xCR1[TWF]中配置FIFO水印。
12. 使能发射器/接收端。使能发送端/接收端后，将产生位时钟和帧同步信号（如果配置为主信号）。

4.7.1. 触发SAI传输/接收操作

如果用户想要启动SAI传输/接收，在初始化SAI模块后，用户需要执行下一步操作。：

- 使用DMA
    1. 请确保DMAMUX模块已经配置为使用SAI的发射器/接收端请求。
    2. 如果SAI配置为发送端，请写入SAI发送端FIFO，以填充FIFO，让发送端缓冲区在发送开始时已经有数据，否则，可能会发生underrun的错误。
    3. 请确保DMA的TCD已配置为将数据从RAM位置移动到SAI发送器/接收端。根据SAI配置（每个通道的位数、水印值），TCD配置可能会有所不同。考虑到对于发送端，有8个深度的数据已经从RAM移动到SAI的FIFO缓冲区。
    4. 在SAI的寄存器上使能DMA请求：SAI_xCSR[FWDE]和/或SAI_xCSR[FRDE]。
    5. 在SAI_xCR3[TCE]中使能所使用的通道。
- 使用中断。
    1. 确保在NVIC中已经使能了SAI中断。
    2. 如果SAI配置为发送端，请写入SAI发送端FIFO，以填充8个深度FIFO，让发送端缓冲区在发射开始时已经有数据，否则，可能会发生underrun的错误。
    3. 在SAI的寄存器上使能中断请求：SAI_xCSR[FWIE]和/或SAI_xCSR[FRIE]。
    4. 在SAI_xCR3[TCE]中使能所使用的通道。

4.7.2. 暂停/停止SAI传输

如果用户通过DMA/中断传输数据，并且打算暂停/停止传输，用户应该执行下面列出的步骤，以避免发生FIFO错误事件。

1. 通过分别清除SAI_xCSR[FWDE]和/或SAI_xCSR[FRDE]或SAI_xCSR[FWIE]和/或SAI_xCSR[FRIE]来禁用DMA请求/中断生成。
2. 等待SAI_xCSR[FWF]置位，即FIFO为空。
3. 通过清除SAI_xCR3[TCE]中的相应位来禁用所有已使用的通道。
4. 如果要停止传输，请通过设置SAI_xCSR[FR]来重置FIFO。

5. 针对不同音频格式的SAI配置

下表显示了根据某些音频格式进行的不同配置字段。下面未列出的其他一些配置字段可能会根据Codec体系结构的不同而有所不同。其中Data bits表示多少位代表一个音频帧，而Slot Size表示每个通道中包含多少位。例如考虑SAI时钟取自总线时钟（40MHz），期望的采样率为48kHz。

表3.针对不同音频格式的SAI配置字段

Audio Format	Data bits	Frame Sync Early *SAI_xCR4[FSE]*	Frame Sync Width *SAI_xCR4[SYWD]*	Slot Offset *(SAI_xDRn)]*	Slot Size *SAI_xCR5[WNW]*	Slot count *SAI_xCR4[FRSZ]*	Bit Clock divider *(SAI_xCR2[DIV])*	Channel Mask *SAI_xMR[xWM]*	Diagram
I2S	32	1	31	Data[31:0]	31	1	6	0xFFFC	Figure A. I2S 32bit format
	24	1	31	(Data[23:0] << 8) & 0xFFFFFF00	31	1	6	0xFFFC	Figure B. I2S 24bit format
	16	1	31	(Data[15:0] << 16) & 0xFFFF0000	31	1	6	0xFFFC	Figure C. I2S 16bit format
Right-Justified	32	0	31	Data[31:0]	31	1	6	0xFFFC	Figure D. Left-Justified 32bit / Right-Justified 32bit formats
	24	0	31	Data[23:0] & 0x00FFFFFF	31	1	6	0xFFFC	Figure H. Right-Justified 24bit format
	16	0	31	Data[15:0] & 0x0000FFFF	31	1	6	0xFFFC	Figure I. Right-Justified 16bit format
Left-Justified	32	0	31	Data[31:0]	31	1	6	0xFFFC	Figure D. Left-Justified 32bit / Right-Justified 32bit formats
	24	0	31	(Data[23:0] << 8) & 0xFFFFFF00	31	1	6	0xFFFC	Figure E. Left-Justified 24bit format
	16	0	31	(Data[15:0] << 16) & 0xFFFF0000	31	1	6	0xFFFC	Figure F. Left-Justified 16bit format
	16	0	15	(Data[15:0] << 16) & 0xFFFF0000	15	1	12	0xFFFC	Figure G. Left-Justified 16bit format (Slot size is 16)
TDM	32	0	0	(Data[15:0] << 16) & 0xFFFF0000	31	7	1	0xFF00	Figure J. TDM, 32bit format, 8 channels

6. 使用示例

当在嵌入式系统中实现音频处理时，正确管理音频缓冲区是很重要的，这样就不会出现任何故障。有几种技巧可以做到这一点，最常见的是实现Ping-Pong缓冲区，连同DMA的使用，减少CPU开销，并促进音频处理。

下一节使用S32K148的基本示例代码来实现此音频处理方法。I2S模式下48kHz的采样率和16bit数据宽度。

附注

尽管指定在I2S模式下每个通道宽度位32位，但是在40MHz采样率下16位宽度能提供较为准确的值。

6.1. Ping-Pong缓冲通道处理（左右通道在相同的缓冲区）

通过在DMA中使用scatter gather功能，系统能够将音频数据存储在两个独立的存储区中。当DMA从缓冲区2传输数据时，CPU可以处理缓冲区1的信息。当传输完成后，DMA将更改其源地址，以开始从缓冲区1发送数据，而CPU将开始处理来自缓冲区2的数据。下图显示了此实现的方框图。

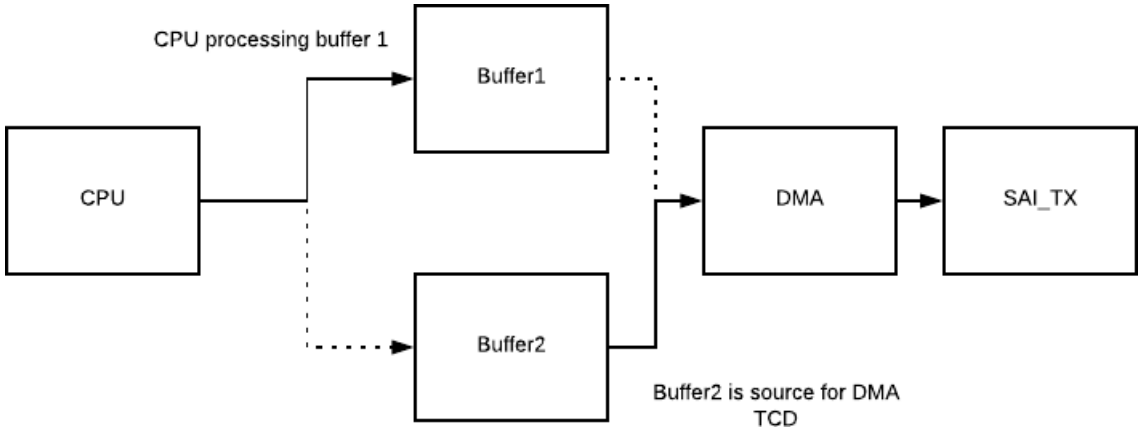


图13.Ping-Pong缓冲区的实现

CPU必须能够在DMA完成传输之前处理所有数据。对于这个特定的用例，每个DMA传输都会发送8 word，因此DMA完成传输所需的时间由以下公式给出：

$$t = \frac{Total\ words}{sample\ rate \cdot words\ per\ frame} = \frac{8\ words}{48\ kHz \cdot 2\ words} = 83.33\mu s$$

为了确定哪个缓冲区是空闲的，在程序中可以将当前DMA的TCD的源地址(DMA_TCD[n]。SADDR)与存储在SRAM中的第一个TCD阵列上保存的地址相比较。

应用笔记中的示例代码已经满足了大部分使用场景需求。

6.2. SAI接收端将左右通道数据放置在不同缓存中

在一些音频应用中，音频处理应该分别应用于每个通道，因此，在这种情况下，接收端和发送端都有分开的左右缓冲区，DMA则需要从适当的缓冲区发送/接收数据。

为了做到这一点，TCD在每个数据传输和每个请求结束时都配置了特别的偏移量。DMA配置起着正确获取有序数据的作用。下图显示了配置偏移量的方式。

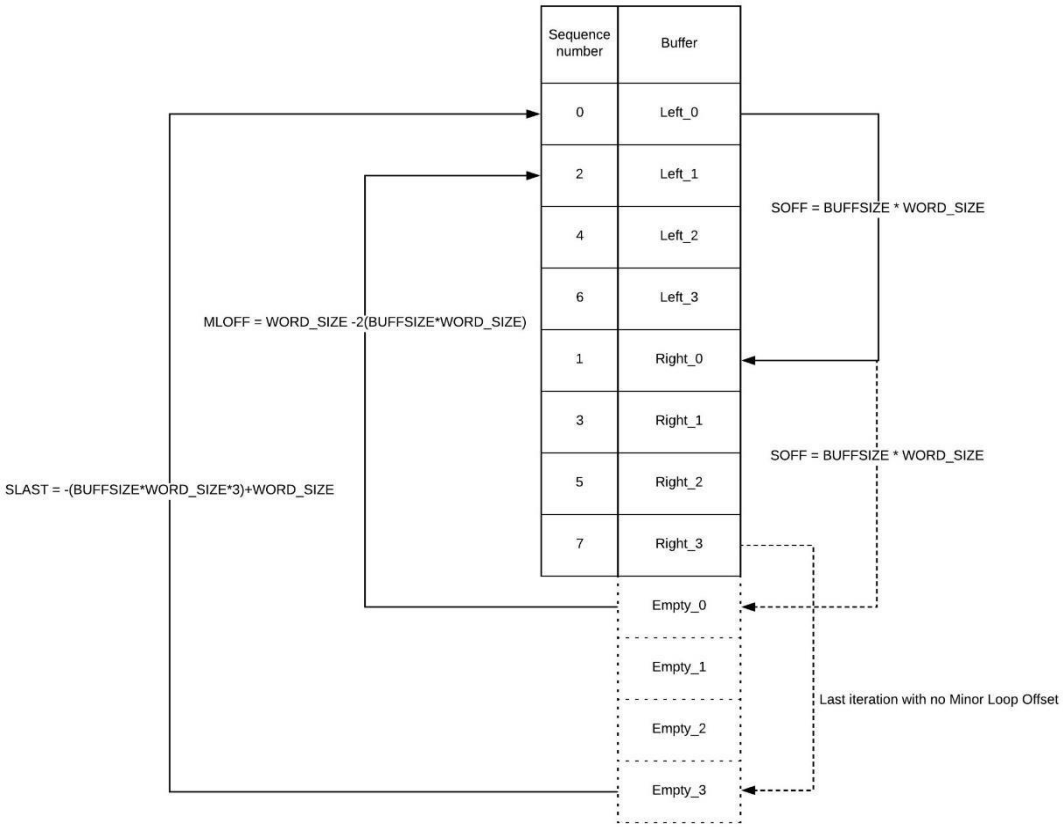


图14。DMATCD偏移量配置

随着TCD配置为每次请求传输两个SAI通道数据（根据SAI配置的1、2或4字节每个通道），DOFF则可以被使用，而DOFF偏移则就是缓冲区1和缓冲区2之间的地址差（或缓冲区大小乘以通道数据宽度（1，2，4））。一旦请求完成（传输了这两个通道数据后），将应用小循环偏移量，因此它将返回到上一个缓冲区的下一个地址。

这个过程将重复“citer”次数，以至于两个缓冲区充满的数据。一旦主循环完成，小循环偏移则不再使用，因此SADDR/DADDR指向“辅助缓冲区”上的最后一个元素。因此，需要调整（-3*缓冲区大小*+通道数据宽度）。

对于这个示例代码，SAI0将向SAI1发送数据，SAI1负责接收并分离数据的。

对于SAI1，每个接收缓冲器需要多加4个深度的大小，因为发送端一开始需要发送额外的8个深度的数据（4个作为右通道和4个作为左通道），这多出来的8个深度的数据就是在初始化过程中为了避免FIFO出现underrun错误一开始填入FIFO的值。

发送端（SAI0）和接收端（SAI1）都使用相同的DMA逻辑将数据移动到不同的缓冲区。下图显示了在SAI上看到的数据，以及当接收到这些数据时，如何将它们分割为两个缓冲区。

7. SDK中的SAI驱动。

SDK（0.8.6版本）中包含SAI驱动。有一个类似的示例代码，在6.2.

您可以查阅SDK文档，以了解有关用于配置SAI模块的API的更多信息。

8. 参考文献

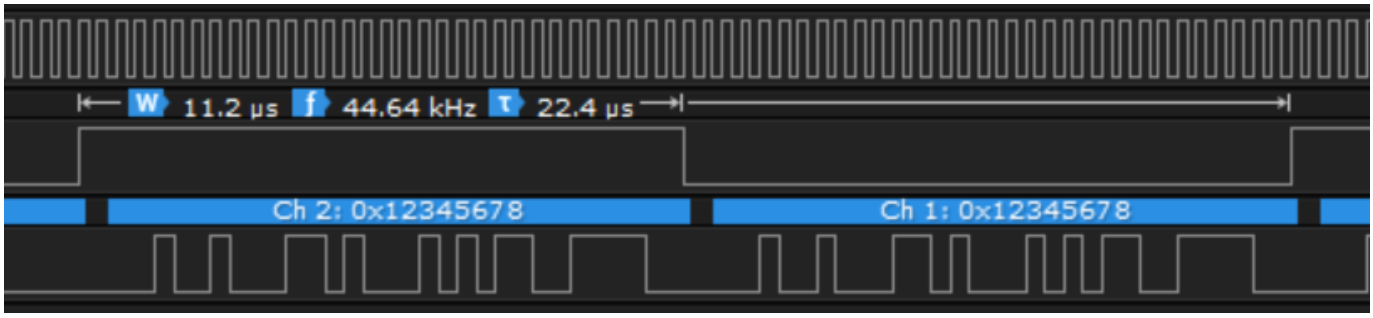
https://www.keil.com/pack/doc/CMSIS/Driver/html/group_sai_interface_gr.html

9. 修订版历史记录

版本号	修订日期	变更的说明
修订版0	2018年11月	初始发布版本

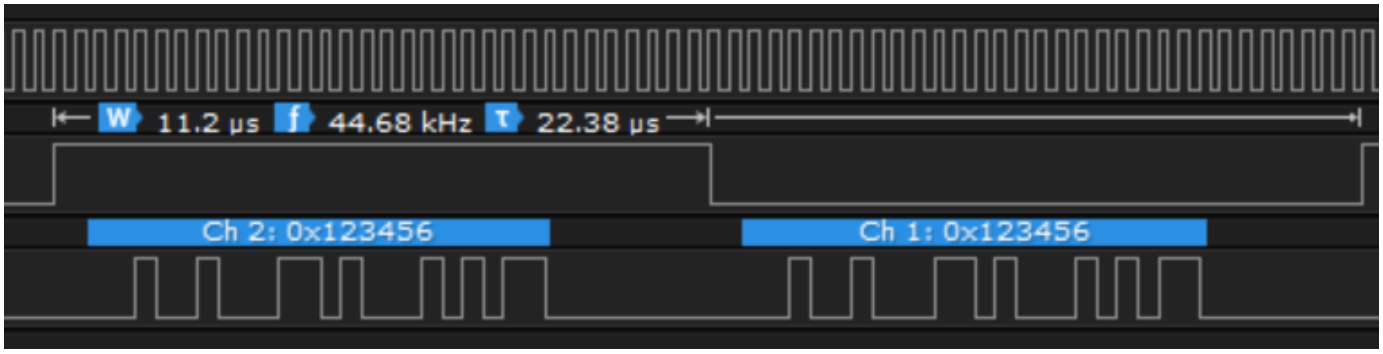
附录A.附图

图A.I2S 32位格式



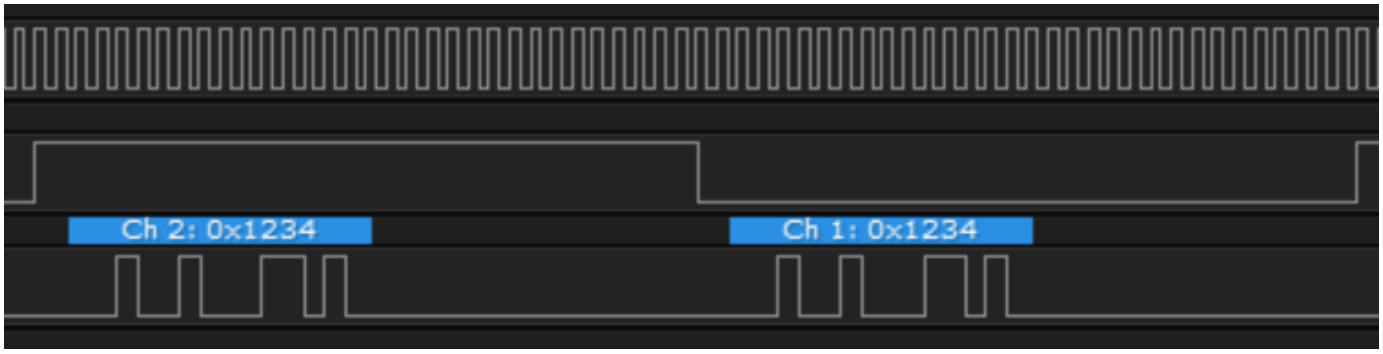

```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = txBuffer[fifoIndex];
}
```

图B.I2S 24位格式



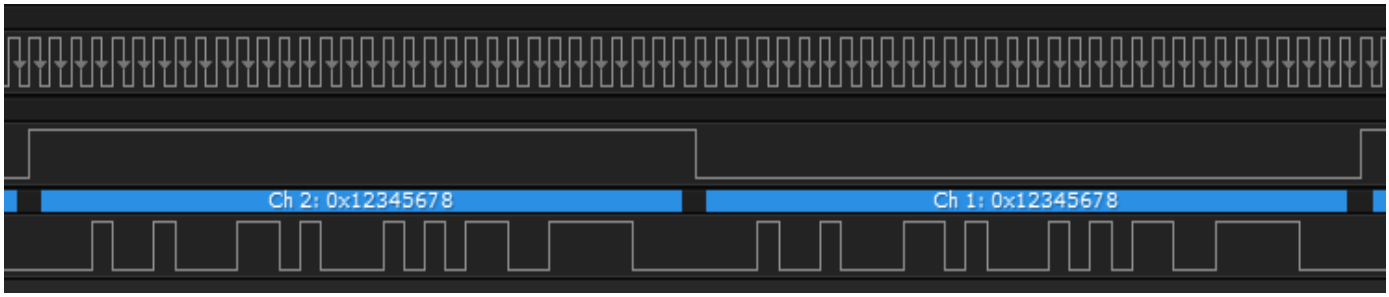

```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = txBuffer[fifoIndex] & 0xFFFFFF00;
}
```

图C.I2S 16位格式



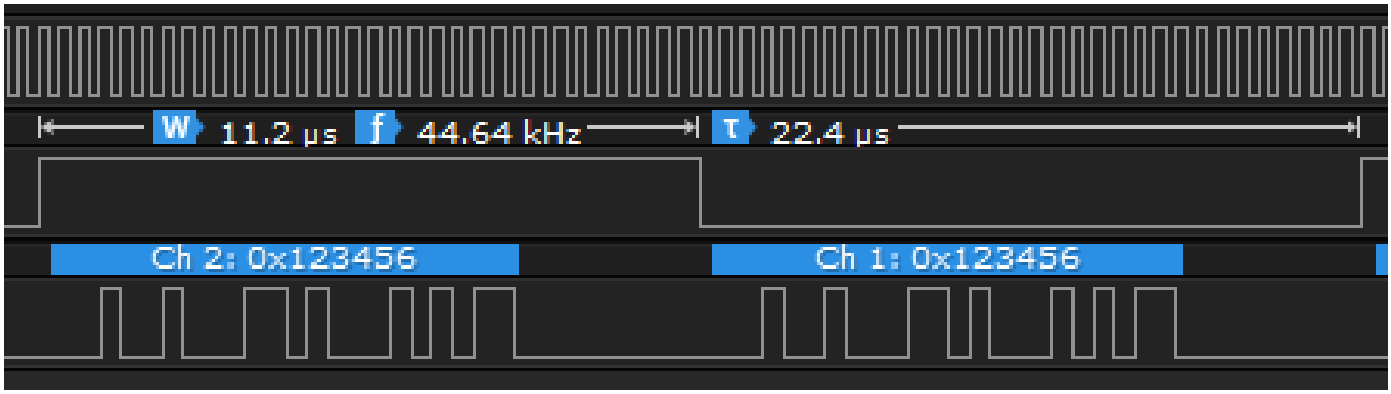

```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = txBuffer[fifoIndex] & 0xFFFF0000;
}
```

图D.左对齐32位/右对齐32位格式



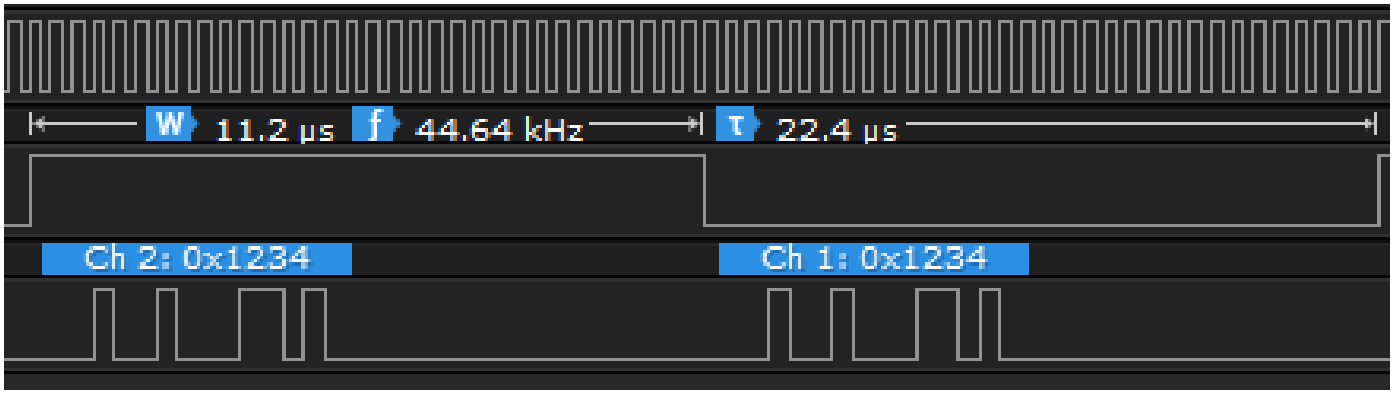

```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = txBuffer[fifoIndex];
}
```

图E.左对齐24位格式



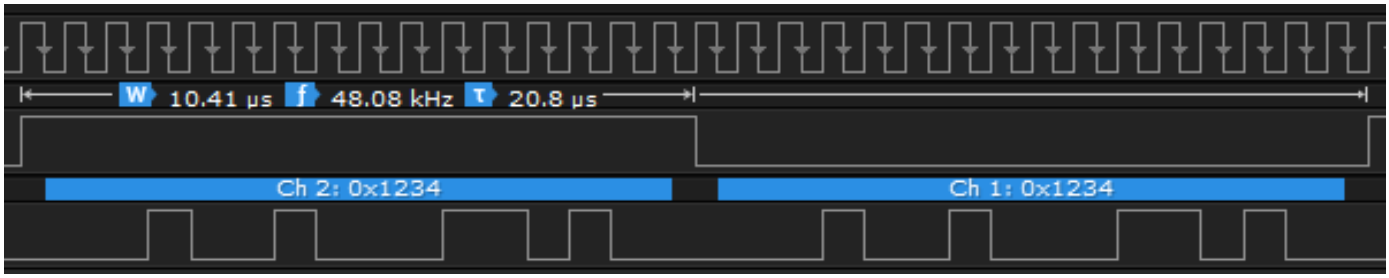
```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = txBuffer[fifoIndex] & 0xFFFFFF00;
}
```

图F.左对齐16位格式



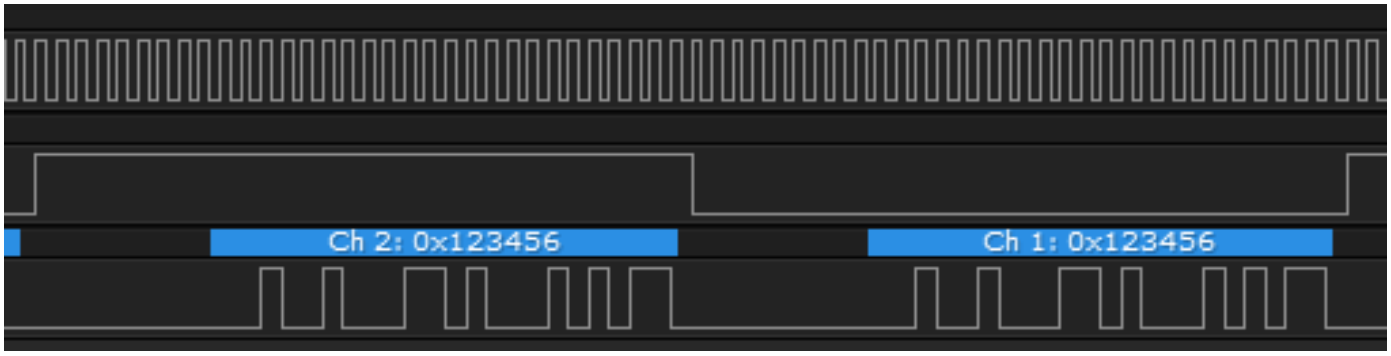

```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = txBuffer[fifoIndex] & 0xFFFF0000;
}
```

图G.左对齐16位格式（插槽大小为16）



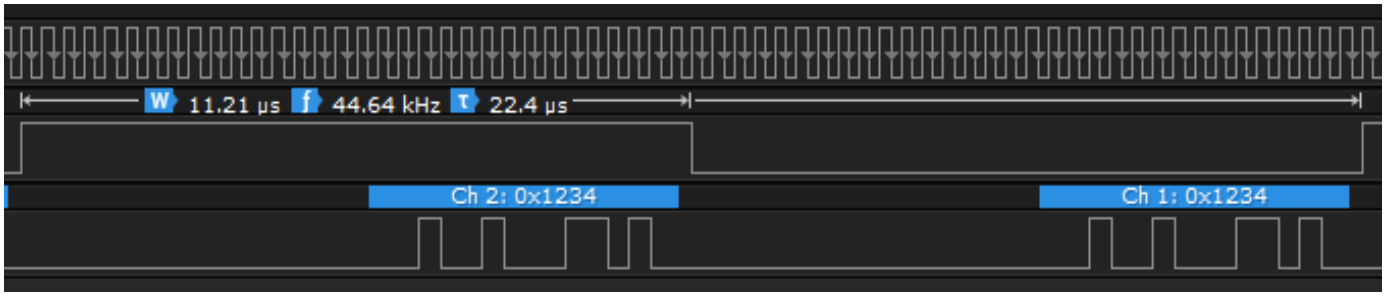

```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = txBuffer[fifoIndex] & 0xFFFF0000;
}
```

图H.右对齐24位格式



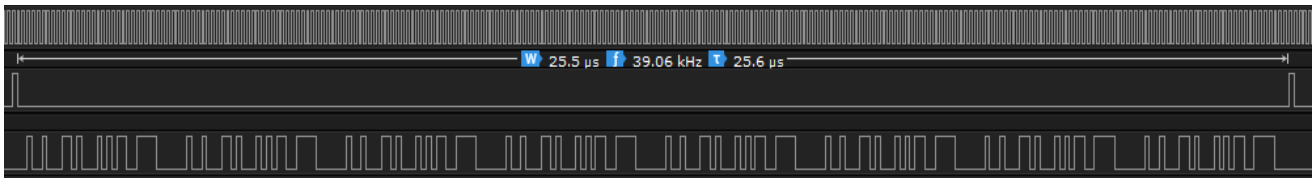

```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = (txBuffer[fifoIndex] >> 8) & 0x00FFFFFF;
}
```

图I. 右对齐16位格式




```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = (txBuffer[fifoIndex] >> 16) & 0x0000FFFF;
}
```

图J.TDM，32位格式，8个通道




```
uint8_t fifoIndex = 0;
uint32_t txBuffer[8] = {0x12345678, 0x12345678, 0x12345678, 0x12345678,
                        0x12345678, 0x12345678, 0x12345678, 0x12345678};
for (fifoIndex = 0; fifoIndex < 8; fifoIndex++)
{
    SAI0->TDR[0] = txBuffer[fifoIndex];
}
```

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN12202
Rev. 0
11/2018