ChipCorder®
TECHNOLOGY BY ISD

APPLICATIONS NOTE 2

Message Management in the ISD33000 Series

This note suggests methods of message management that make the ISD33000 series easier to use in multiple message applications.

Many types of products for sound record and playback require message management capabilities. They need to be able to record, play or erase an individual message as needed. Some technologies are not well suited for this, such as magnetic tape-based recording systems which are unable to erase a message in the tape's middle then reuse during a new random length recording. Because of message management features built into the ISD33000 SPI-controlled record and playback IC these problems are eliminated.

MESSAGE MANAGEMENT RECORD

In a typical application, several messages are initially recorded into the chip in sequence, for example: Message #1, Message #2, Message #3, and Message #4. Since the ISD33000 device is able to read back the internal message pointer at the conclusion of each recording, a simple message address table is constructed to save the address location of each message in a separate digital, nonvolatile memory. A specific message, number three, for instance, may be selected and played back when needed.

A more usual application of message management is deleting nonconsecutive messages and reusing the space for future recordings (for example deleting messages one and three). Because of the message address table earlier constructed, the device is aware of the beginning and ending of each message previously recorded. During record the End Of Message (EOM) flags that mark these messages are not accessible; these flags are used only during playback. The recording process erases any EOM previously written at the end of a message. An alternative method is therefore needed to determine that the end of old messages (messages one and three) is found in real-time during recording. Once at the end of these messages the device needs to "jump" to a new address space without losing any analog data. The following explain how this is accomplished.

Two special message addressing features are built into the ISD33000 device: the Row Address Clock (RAC) output as well as a special mode included in the address management logic of the chip. To understand how these features work, it is first necessary to understand the memory architecture of the device.

The ISD33000's memory consists of a number of rows of memory cells, each row consisting of 1200 columns. When addressing the device, select the start of a row to begin record or playback. It is not possible to address inside a row. All operations in a row begin at column 1. For example, in the 8 KHz sample rate device, the ISD33120, samples are taken at an interval of 125 μs (consult the ISD33000 data sheets for the timing of other members of the family). Thus, 125 μs times 1200 columns per row, then each row contains 150 ms of sound. That defines the address resolution of this device at 150 ms and each record or playback cycle begins at the start of these 150 ms "blocks" of audio memory. The ISD33120 has 800 rows of memory and therefore (800 x .150) 120 seconds total memory.

The RAC pin of the ISD33000 series indicateS that record or playback is near the end of a row of memory. Using the ISD33120 as an example, the RAC pin goes LOW 12.5 ms before the end of each row and goes back HIGH exactly at the end of each row. It therefore becomes a "row clock" which determines where the device is in memory during record or playback operations. This is a slow clock, with a period of 150 ms (in this example). In a microcontroller environment it is relatively easy to use software polling of the RAC pin to determine when the end of each row has been reached.

In the message example above, the length of Message #1 and #2 are known and therefore the number of rows in each message space is easily computed. If Message #1 is *n* rows long, simply count *n* – 1 rows and employ the second message management feature built into the ISD33000.

After *n* – 1 RAC cycles and after the RAC pin has returned to HIGH, the device is now recording on row *n*, the last available row in this block (the old Message #1 space) of memory. Once recording begins on a row, it is now possible to utilize the second major message management feature of the ISD33000 family: a forced address jump. In the example, recording a new message continues after using up the space previously taken up by the old Message #1. To jump to the beginning of the next available address space (the beginning of old Message #3), without losing Message #2, carry out the following sequence:

1. RAC goes HIGH after *n* – 1 RAC cycles. Indicating that we are now recording on the last row of the old message #1 address space.
2. A new SPI addressed record cycle is initiated with the IAB bit LOW. The address is the location of old Message #3's beginning. Since recording is already under way, this operation will be ignored until the end of the current row.
3. The final falling edge of the RAC signals the end of the last row of the old Message #1 is near. When the RAC pin returns to HIGH, recording continues at the new address with no loss of audio samples.
4. While recording proceeds, the microcontroller controlling the operation must erect a message address table describing the location of each segment of the now-fragmented message.
5. At this time another SPI control cycle is sent to the ISD33120 device with the IAB bit set HIGH. This causes recording to continue in sequence through the old message three space.
6. If recording continues long enough so that all of the old Message #3 space is used up, a new address must be input so that recording can continue at the appropriate place after Message #4.
7. When recording ends, a SPI STOP command causes an EOM bit to be set indicating the end of the current message.

MESSAGE MANAGEMENT PLAYBACK

Playback of this new message proceeds in the same manner as record. The message address table is used to calculate where and when each jump occurs during playback. Only one EOM will be set in the entire fragmented message space to indicate the final end of this message.

MESSAGE ADDRESS TABLE

The Message Address Table (MAT) is an important part of the message management structure. This table keeps track of the beginning address of each message as well as the beginning and ending address of each fragment of a message. Additionally, a more sophisticated MAT may also contain information about message numbers, message order and other factors such as whether or not the message has been listened to since it was recorded.

The MAT may be structured in a variety of ways. For example, every row of the ISD33000 device may be treated as an independent block of memory with an entry in the MAT to indicate it's relationship to the various messages stored in the device. This may require more memory than practical, however, as the members of the ISD33000 series have 400 to 800 rows of memory storage. Tracking 800 rows not only requires 800 memory locations in the MAT, but also a digital word larger than 8 bits. Consequently, such a MAT might need as many as 1600 digital bytes, which may not be cost effective in some applications.

It is more practical to work with multiple rows serviced together as a single, larger, memory block. An ISD33120 segmented into 100 blocks, each 8 rows long, could be serviced by a MAT as small as 100 bytes. The following discusses one method of accomplishing this.

THE MESSAGE ADDRESS TABLE—AN EXAMPLE

The ISD33120 requires 10 bits to address any of its 800 rows of memory. Assume that each sequential 8 row block of memory is addressed as a single block, the lower 3 bits of memory are not needed. They are always zero. Now, only 7 bits of address are required to access any of 100 blocks in the message memory of the device. In this example each block is 0.8 seconds long.

The MAT in this example is a memory stack of 100 bytes where each byte may be associated with one of the blocks of memory in the ISD33000 device. As messages are recorded and erased, the specific association may change. How this works will become more apparent later in this section.

The data in the MAT memory byte either indicates no association with a block of memory or the status of a specific block of memory in the ISD33000 device. Bit 7 of each MAT memory byte has a special function. A "1" in this bit indicates that this byte points to the first block of memory in a message. The remaining 7 bits of each byte contain the address of one of the blocks of memory. There are four possibilities:

1. The data in the byte is all zeros (0000 0000). This indicates that there is no association with any memory block in the ISD33000.
2. The data in the byte has a "1" in bit position 7 (the MSB) and a seven bit address in bits 0–6 (1xxx__xxxx). The 1 in bit position 7 indicates that this is the first block of a message. The seven bit address points to the location of the first block of memory in the message.
3. The data in the byte has a "0" in bit position 7 and a valid seven bit address in bits 0–6 (0xxx xxxx). The 0 indicates that this is not the first block of a message in memory. The valid seven bit address points to the current block of memory in the message.
4. A special case of number two above occurs when addressing the first block of memory in the ISD33000 device. By convention, the first block of memory will always be the start of a message. As proven in this example, this is always the case. The data in the byte has a "1" in bit position 7 and all "0"s in the rest of the byte (1000 0000).

BUILDING THE MAT

While recording several messages, the MAT will be built from the top of the table, down, as space in the ISD33000 is consumed. One byte of the MAT is employed as each block of memory in the IC is used. Bytes associated with a single message in the MAT will always be sequential. The block addresses in a single message made up of multiple MAT entries will always be in numerical order but may not be sequential.

After recording and erasing a number of messages, an example MAT might look like Table 1.

This memory currently holds four messages of varying lengths. Notice that Message #1 is composed of blocks 5, 6 and 9, and Message #2 of blocks 2 and 8. This 100-byte MAT applies only the first 12 bytes and leaves the remaining 87 bytes filled with zeros.

Table 1: MAT Example

Message Add. Table (MAT)	=	Bit 7	Block Address	Block Add (dec)	Msg No.
1000 0101		1	000 0101	5	
0000 0110		0	000 0110	6	One
0000 1001		0	000 1001	9	
1000 0010		1	000 0010	2	Two
0000 1000		0	000 1000	8	
1000 1010		1	000 1010	10	
0000 1011		0	000 1011	11	Three
0000 1111		0	000 1111	15	
1000 0111		1	000 0111	3	
0000 1100		0	000 1100	4	Four
0000 1101		0	000 1101	7	
0000 1110		0	000 1110	14	
0000 0000 (87 bytes follow, all zeros)					

PLAYBACK ALGORITHM

To demonstrate the playback algorithm, Message #3 will be played back. The microcontroller controlling the system begins at the top of the MAT and searches down for bytes with bit 7 set to a "1." In this playback example, the microcontroller will stop on the third occurrence of this condition, pointing at the first pointer byte for Message #3. From this byte, the microcontroller knows that block 10 is the beginning block of ISD33000 memory for Message #3. The playback of Message #3 starts with block 10 being addressed through the SPI port. Once playback of the message commences, a second SPI cycle is input without an address and the IAB bit set so that playback will continue sequentially through the 8 rows of memory in this first message block.

NOTE *An easy way for the microcontroller to compute the address for the ISD33000 is to first clear bit 7, if set then load the block address into a register and shift left 3 bits with "0" shifted into the LSB bit position. As bits are shifted out of the MSB bit position, they need to be saved in a second byte. The original 7-bit address will now be a 10-bit address that may be used to start message playback.*

While the current block of the message plays back, the microcontroller now starts polling the RAC pin of the ISD33000. This pin has the same period as one row of memory of the device, that is, 150 ms in the ISD33120. The RAC pin goes LOW for the last 12.5 ms of each row. It's high-going edge is synchronous with the end of the current row. The microcontroller polling, therefore must be fast enough to "catch" the RAC pin while it is in the LOW state. The microcontroller must count seven complete RAC cycles, at the conclusion of the seventh RAC cycle, the ISD33000 is playing back the last row (the eighth row) in the current block.

When the eighth row of the block starts playing back, the microcontroller must return to the MAT and pull the next byte from the table. In the case of Message #3, the next MAT byte is 0000 1011. This indicates that the next block to be played is 11. Since playback has already begun on the last row of the block, the SPI may now input data as if to begin a new playback at the new address, that is, the address of block 11. This new SPI cycle does not have an immediate effect. The new address is buffered up awaiting the end of the row.

In this specific case, playback continues to the next sequential block in the ISD33000 device, and hence, the next sequential row in the memory of the device. It is possible to design a "smart" algorithm which detects playback (or record) proceeding in sequence from block to block and to ignore this SPI cycle. This is not explained in this section.

When playback reaches the end of this last row of the first block of Message #3, it automatically jumps to the first row of block 11 and continues recording without losing a sample. When playback reaches this point, (after the start of row 1 of block 11), it is again necessary to input a second SPI playback cycle without an address and with the IAB bit set to 1. This causes playback to proceed sequentially through the 8 rows of block 11.

When row 8 of the second block is reached, another SPI cycle inputs the address of the third and final block of the message. MAT shows that this is block 15. As with the first and second block of the message, input an immediate second SPI cycle to set the IAB bit.

Playback is now proceeding through the third and final block of Message #3. Since this is the last block of the message, assume that an EOM bit is set in one of the block's 8 rows. When the EOM bit is encountered, an EOM interrupt occurs and playback ends. If this is not the case, the message was improperly recorded or this is the last row in the ISD33000's memory and playback was terminated with an overflow interrupt.

A third special case situation may exist: the message may stop at the end boundary with a completely full block. If this occurs, then the microcontroller will go to the MAT a fourth time and read a byte of data. The data it reads shows the start of a new message (bit 7 = 1) and a block address of 7. When the message start flag is detected, however, the microcontroller assumes that this must be the end of the message and does not input a new SPI cycle. The message ends with an EOM interrupt to signal that the message is finished.

If the last message in the MAT table is being played and the above situation occurs, the microcontroller will read an all-zero byte. It will treat this situation the same as in the above paragraph: that this is the end of the message.

The software in the microcontroller must also be intelligent enough to know when the 100th byte of the MAT is being read. In this record/playback algorithm demonstrates that this block must be the end of a message.

Cascaded ISD33000 devices are not covered in this example. In a cascade application, a bit (or bits) in the MAT are reserved to indicate which ISD33000 device contains which block of memory.

RECORD ALGORITHM

To demonstrate the record algorithm, we record a new message, Message #5, and assume there is a current MAT that looks like Table 2.

When recording begins, the microcontroller must find an unused block in the ISD33000 device. There are a number of ways to accomplish this. The easiest is for the microcontroller to start with block zero and search through the MAT to see if that block is already in use. If zero is already in use, it then searches for block 1 and does the same test, then block 2, and so on. When an unused block is located, recording begins on that block in the ISD33000.

Table 2: New Entry in MAT Message

Message Add. Table (MAT)	=	Bit 7	Block Address	Block Add (dec)	Msg. No.
1000 0101		1	000 0101	5	One
through					
0000 1101		0	000 1101	7	Four
0000 1110		0	000 1110	14	
1000 0000		1	000 0000	0	Five
0000 0000 (86 bytes follow, all zeros)					

In the demonstration example, according to the MAT, block zero is available. The microcontroller starts recording at block zero by executing an SPI record cycle to the correct address (in this case, the address is zero). Recording begins immediately at the end of the SPI cycle. As in playback, a second SPI cycle must now immediately be input without an address and with the IAB bit set so that recording will proceed sequentially through the rows of block zero.

At the same time, the microcontroller must create a new entry in the MAT for Message #5. Since the message starts at block zero, the new MAT byte will be 1000 0000. Table 2 shows this new entry for Message #5.

After the new MAT table entry is entered, the microcontroller begins looking for another unused block of memory. A smart algorithm "remembers" that a search through the memory to the point where the first unused block was found already occurred. The algorithm begins at this point (block 1 in this example) and seeks the next unused block (block 1).

As in the playback algorithm, the microcontroller must count RAC cycles to determine when the eighth row of the block is reached. After recording on the eighth row starts (at the end of the seventh RAC cycle), the microcontroller enters a new SPI record cycle to record at block 1. When the row ends, recording will continue without stopping through to block 1.

Since recording has continued into the second message block, the microcontroller builds another MAT table entry and locates another available memory block. Table 3 shows the new MAT table entry. Referring back to Table 1 and remembering that blocks 0 and 1 were already used, the next available block must be 12.

Table 3: New MAT Table Entry

Message Add. Table (MAT)	=	Bit 7	Block Address	Block Add (dec)	Msg No.
1000 0101		1	000 0101	5	One
through					
0000 1101		0	000 1101	7	Four
0000 1110		0	000 1110	14	
1000 0000		1	000 0000	0	Five
0000 0001		0	000 0001	1	
0000 0000					
(85 bytes follow, all zeros)					

Two more blocks are recorded and then a STOP command is executed. The STOP command causes an EOM bit to be written into the ISD33000 memory. The MAT will resemble Table 4 when recording Message #5 is complete.

Table 4: Using the STOP Command

Message Add. Table (MAT)	=	Bit 7	Block Address	Block Add (dec)	Msg No.
1000 0101		1	000 0101	5	One
through					
0000 1101		0	000 1101	7	Four
0000 1110		0	000 1110	14	
1000 0000		1	000 00000		
0000 0001		0	000 0001	1	Five
0000 1100		0	000 1100	12	
0000 1101		0	000 1101	13	
0000 0000					
(83 bytes follows, all zero)					

ERASING A MESSAGE

To erase a message remove that message's entry from the MAT. For example, to erase the recently recorded Message #5 simply change its four table entries to all zeros.

It is more complicated to erase Message #3. Please refer to Table 1. Message #3 uses 3 bytes in the MAT to describe the location of its three message blocks of storage space. These 3 bytes are (from the top down) sequentially the sixth, seventh, and eighth bytes of the MAT stack. To erase Message #3, shift all the bytes in the stack below this message up three places, writing over the 3 bytes that used to refer to Message #3. Still referencing Table 1 after this action is complete, a new Message #3 composed of the MAT stack bytes that were Message #4. Message #3 has disappeared from the MAT and thus was "erased."

Table 5 demonstrates what the original Table 1 looks like after Message #3 is erased.

REORDERING MESSAGES

An additional advantage to this message management system is that the message order can be reordered by system requirements as needed. Again referring to Table 5, it is easier to interchange Message #1 and Message #3. To accomplish this, perform the following sequence:

1. Copy the three bytes of the Message #1 pointer to a temporary stack.
2. Shift all bytes of the MAT down 1 byte.
3. Copy the 4 bytes of the Message #3 pointer into the first 4 bytes of the MAT.
4. Copy the 3 bytes of the old Message #1 from its temporary stack to the location in the MAT below Message #2.
5. Message #1 has become Message #3 and vice versa.

The new MAT looks like Table 6.

Table 5: Erasing Messages

Message Add. Table (MAT)	=	Bit 7	Block Address	Block Add (dec)	Msg No.
1000 0101		1	000 0101	5	
0000 0110		0	000 0110	6	One
0000 1001		0	000 1001	9	
1000 0010		1	000 0010	2	Two
0000 1000		0	000 1000	8	
1000 0111		1	000 0111	3	
0000 1100		0	000 1100	4	Three
0000 1101		0	000 1101	7	
0000 1110		0	000 1110	14	
0000 0000 (90 bytes follow, all zeros)					

Table 6: Reordering Messages

Message Add. Table (MAT)	=	Bit 7	Block Address	Block Add (dec)	Msg No.
1000 0111		1	000 0111	3	
0000 1100		0	000 1100	4	One
0000 1101		0	000 1101	7	
0000 1110		0	000 1110	14	
1000 0010		1	000 0010	2	Two
0000 1000		0	000 1000	8	
1000 0101		1	000 0101	5	
0000 0110		0	000 0110	6	Three
0000 1001		0	000 1001	9	
0000 0000 (90 bytes, follow, all zeros)					

CONCLUSION

This Application Note has shown an easy method of message management in the ISD33000 voice playback and record integrated circuit. The ability to record, play or erase an individual message as required has been demonstrated. Additionally, messages can be reordered as needed by the application requirements.