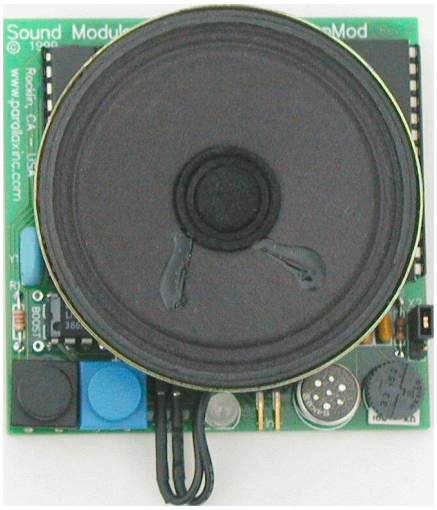
599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: www.parallax.com/sic

Sound Module AppMod – Rev B (#29111)

Miniature Record and Playback Module with a serial interface



General Description

There is little doubt that the addition of sound or music can greatly enhance any project. Historically, this has been a difficult task to accomplish. The ICs required to do so are expensive and difficult to work with, especially for beginners. The advent of ISD's family of 'Direct Analog STorage' (DAST) chips has brought the whole process one step closer to the public. And now the Sound Module makes it so easy that everyone can implement audio record and playback capability into their projects.

Features

- **Records/plays up to 60 seconds of sound, as many messages or one big message.**
- **Simple serial interface 'auto-bauds' any standard baudrate from 2400 to 38.4K.**
- **Two pushbuttons are provided for local operation.**
- **Input is selectable: microphone or line-in.**
- **1W Audio amplifier, volume control, and speaker are on-board.**
- **Connections provided for optional 'BOOST' capacitor to make it really loud.**
- **Serially controlled disable feature protects against accidental recording.**
- **New 'M' command allows you to split any message into multiple messages.**

How it Works

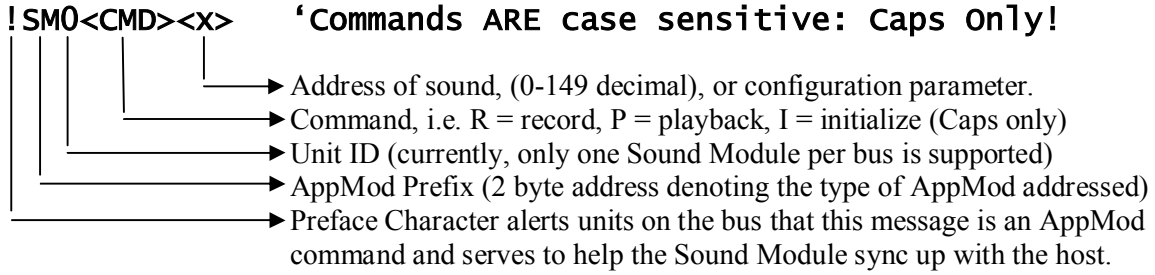
Understanding a little bit more about how the ISD chips works will help you produce better recordings and allow you to use the Sound Module to its fullest potential.

The ISD chip operates at 8 kHz. This means that when the chip is recording, every 122 uSec, (0.000122 Sec), the ISD chip samples and stores the analog input value. Recording is initiated by a low signal on the CE (chip enable) pin on the ISD chip while the P/R (Play/Record selector) pin is held low. When recording ends (CE goes high), the ISD chip automatically appends an EOM (end of message) tag. Playback is initiated by a low-going pulse on the CE pin, while the P/R pin is high. When playing back, the ISD chip reads the cells from its memory and updates the analog output every 122 uSec until the EOM tag is encountered, then playback is terminated. At the beginning of both playback and record, the address bus is read to determine the location in memory to start recording or playing back. Fear not, the firmware chip on board controls the address bus and handles all of these details for you.

* ISD and DAST are registered trademarks of Information Storage Devices, Inc.

Serial Protocol

The host transmits commands serially to the Sound Module. The Sound Module will not reply unless it has been configured to output an EOP (end of playback) pulse. The EOP pulse will be discussed later in this document. The command syntax is structured to facilitate the allowed commands and prevent bus contention in the event that more than one AppMod is on the same serial line. The Sound Module uses I/O pin 4 (P4) for the serial interface. Since the Sound Module needs a little time for power up and in-between messages, it is necessary to have a delay before each serial message. A 'pause 50' command is usually a sufficient delay.



Commands

C Configure the sound module as specified by x.

```
serout 4,188+$8000,["!SM0C",1]          'EOP pulse enabled
```

This command will configure the Sound Module to pulse the serial data line low for 128 mSec at the end of a message, (**E**nd **O**f **P**lay pulse), that is played back. Some applications require that the host unit (usually a BASIC Stamp) knows when each message has completed playback. While in this mode, the Sound Module will not listen for serial commands until after both the playback and the EOP pulse have completed.

```
serout 4,188+$8000,["!SM0C",0]          'EOP pulse disabled
```

This command will revert the Sound Module to its default configuration: no EOP pulse. While in this mode, the Sound Module will listen for serial commands. If you command the Sound Module to play a message will the previous message is still playing, nothing will happen. If you need to end a message playback prematurely, you must send an 'I' command.

R Record starting at the address specified by the **x** parameter (in this case **x** = **0**).

```
serout 4,188+$8000,["!SM0R",0]
```

This command will set the Sound Module's internal address pointer to point to message slot **0** (the very beginning of the sound memory), and start recording. It is up to the user to stop the recording process sometime before the end of sound memory is reached. You may record a message at any address from 0 – 149, (each integer represents 0.4 Seconds of time), but care should be taken to not record beyond the end of the Sound Module's memory else an overflow condition will result.

I Initialize the Sound Module and reset the ISD chip.

```
serout 4,188+$8000,["!SM0I"]
```

This command will reset the Sound Module and change the mode to playback. The write enable/disable will remain unchanged. Although the Sound Module will be set to play mode, it will not play a message until instructed to do so. Similarly, the ISD sound chip will be reset, but will retain any sounds that were previously recorded. This command may be used to reset the Sound Module if it goes into overflow mode. It is normal to hear a click when the ISD chip is reset.

M Mark the address specified as the end of a message.

```
serout 4,188+$8000,["!SM0M",12]
```

This command will set the Sound Module's internal address pointer to point to message slot 12, (which is 4.8 seconds from the beginning of memory), and place an EOM tag there. This can be used at least two ways: to set the internal address pointer for manual (pushbutton) operation, and truncate an existing message.

P Playback the sound at the address specified.

```
serout 4,188+$8000,["!SM0P",75]
```

This command will set the Sound Module's internal address pointer to point to message slot 75, (which is 75 * 0.4 Seconds/slot = 30.0 seconds into the sound memory or ½ way through the memory). The message will play until the Sound Module receives an 'I' command, or when the end of that particular message is reached, or until overflow (the overflow state is discussed later).

E End the current recording session.

```
serout 4,188+$8000,["!SM0E"]
```

This command will cause the Sound Module to stop recording a message. Please note that there is no parameter needed for this command. This command will not stop a message that is currently playing.

N Enable recording.

```
serout 4,188+$8000,["!SM0N"]
```

This command will allow the Sound Module to record messages. Please note that there is no parameter needed for this command. The LED will be left as either RED (record mode) or GREEN (playback mode) after receiving this message.

D Disable recording.

```
serout 4,188+$8000,["!SM0D"]
```

This command will prevent the Sound Module from recording messages. Please note that there is no parameter needed for this command. The LED will toggle between RED and GREEN after receiving this command to indicate the record-disabled state. Playback will function normally. Please note that the Sound Module's default state upon powering up is 'Disabled'.

Pushbutton Mode

Local operation is possible using the two pushbuttons located on the Sound Module. The Black pushbutton switches the mode. The mode is indicated by the bi-color LED. When the LED is green, the Sound Module is in playback mode. When the LED is RED, the Sound Module is in RECORD mode. The Blue pushbutton causes the action, (playback or recording), to occur. When the LED is toggling between RED and GREEN, the unit is in protected mode and recording is not possible. The following will guide you through recording and playing a message.

First you must enable recording by sending a serial message to the Sound Module. Here is a BS2 program that will accomplish this:

```
Start:    pause  500
          serout 4, 84+$8000, ["!SM0N"]
          end
```

Note that the pause is required to allow a little time for the Sound Module to power up, initialize, and ready itself to receive serial messages.

Recording from the microphone is accomplished by first selecting the microphone as the source of recording by positioning the shunt on X2 high, thereby shorting pins 1 & 2. Pin 1 is closest to the PCB text "X2". Next, you need to push the black button until the LED is RED. The last step is to press the blue button to record. You must hold the blue button as long as you wish to record. Release the blue button when you are finished recording. As you record, the internal address pointer of the ISD chip increments. If this address pointer should increment all the way to the very end of its memory, an overflow condition will result. If you overflow the chip, the ISD may not respond to subsequent commands until it is reset. To reset the ISD chip, you will need to either cycle power, send a 'P' (play) message, manually play a message using the pushbuttons, or send an 'I' command to reset the ISD chip. The following command will reset the ISD chip:

```
serout 4,188+$8000,["!SM0I"]
```

Recording from the line input is accomplished by first selecting the line input as the source of recording by positioning the shunt on X2 low, thereby shorting pins 2 & 3. Pin 3 is farthest from the PCB text "X2". Connect the line level source to the +In- pins observing polarity. The amplitude of the incoming signal should be close to 90mVp-p. When you are ready to record, you need to push the black button until the LED is RED. The last step is to push and hold the blue button for the duration of the record time.

Playback is accomplished by first pushing the black button until the LED is green. Then momentarily push the blue button once. The current message will play until its end, or until it overflows, whichever comes first. If you were to push the Blue button again, the same message would playback. To play a different message, you must first use the 'M' command to set the address pointer manually.

Overflow is the state of the ISD chip that occurs when the internal address pointer is allowed to increment to the very last location of memory. Overflow can be caused by either a play or a record cycle. If a message was recorded to the point of overflow, playing that same message back will cause the ISD to overflow once again. The most obvious symptom of overflow is the lack of response of the ISD chip to commands. Note 3 of the Programming Notes section of this mini-manual offers code that will fix the dreaded overflow state.

Serial Mode

Serial operation is easily mastered with the BS2. Here is a sample program that automatically records four messages and plays them back:

```
x           var     byte                          'loop counter
Init:       pause  500                            'wait for power up
            serout 4,84+$8000,["!SM0N"]           'Enable recording
            pause  500                            'Wait a little time
Start:      for x = 0 to 3                        'For 4 different addresses:
             serout 4,84+$8000,["!SM0R",(X*5)]    'Record a message
             pause 1950                           'for 1.95 seconds
             serout 4,84+$8000,["!SM0E"]          'then End recording
             pause 1000                           'and one second later
             serout 4,84+$8000,["!SM0P",(X*5)]    'Play that message back
             pause 2000                           'Wait for message
            next                                  'repeat until done
            for x = 0 to 3
             serout 4,84+$8000,["!SM0P",(X*5)]    ' Play 4 messages back
             pause 2000
            next
            end
```

Note that the LED is left it playback (GREEN) mode. You could write protect the device by inserting the following command just before the 'end' command:

```
            serout 4,84+$8000,["!SM0D"]           'Disable record mode
```

To clarify where your sounds will be stored inside the ISD chip, it is good to make a memory map that correlates the addresses used by the stamp program to the physical locations within the ISD chip.

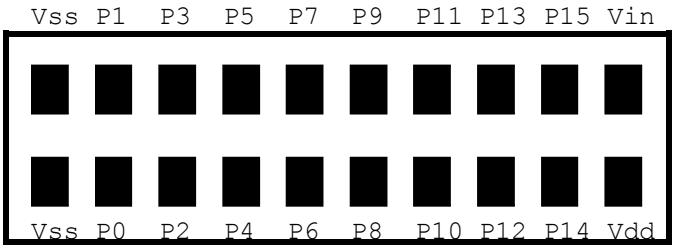
X	Time Slot(X*5)	Time(X*5*0.4S)
0	0	0.00 S
1	5	2.00 S
2	10	4.00 S
3	15	6.00 S

Electrical Specifications

The Sound Module AppMod requires 6 – 12 Volts DC and will draw 29mA (idle) to 90mA (full blast). When using an external speaker (8 Ohms only), it is best that the supplied voltage be 12 Vdc to ensure the best sound quality, although it will work properly at 6 Vdc.

As with all AppMods, the Sound Module AppMod has a special stack-through 2x10 header. This header allows you to connect the Sound Module to either the Board of Education, the BOE-Bot, the Stamp Activity Board, the Super Carrier Board, and of course, other AppMods. We've provided the pin out of the AppMod header below, as viewed from the top.

If you will not be connecting this AppMod to a board with the mating stack-through header, all you need to do is: connect Vss to ground, connect Vin to the positive side of a 6-12 Vdc supply, and connect P4 to the serial host. **Please note: P4 can tolerate a voltage swing from 0 to 5 Vdc only**. Here's a top view of the stack-through header:



Although we have provided a stack-through header so that AppMods may be piled on top of one another, the Sound Module should be the top unit since the speaker covers most of the top of the AppMod header. Of course the speaker may be removed, replaced, or repositioned to allow the header to be exposed again.

A Line-Out connection on the Sound Module AppMod may be used once the on-board speaker is disconnected. Care should be used in connecting this output to other circuitry since this output is downstream of the audio amplifier. The volume control may be used to adjust the output to an acceptable level prior to connection.

If you need additional amplification, you may add a 10 uF 16V electrolytic capacitor across the "BOOST" pads. The negative lead of the cap should be soldered to the pad closest to the 'B' of "BOOST". The gain of the amplifier is increased tenfold. The maximum sound output rating jumps from 26dB to 46dB. If you choose to add the BOOST capacitor, you should use a larger external speaker, (8-16 Ohm), instead of the little speaker supplied with the Sound Module.

Programming Notes

1. Throughout the examples mentioned herein, I/O Pin 4 has been referenced as the serial pin for all serial communications. This is because, with respect to the AppMod header included on various products such as the Board of Education, the BOE-Bot, and the SuperCarrier Board, I/O Pin 4 has been hardwired to serve this function.

2. The examples mentioned herein assume the use of the BS2-IC. Different commands or parameters may be required if you are using something other than the BS2-IC. Please refer to the AppMod code section on the Parallax CD for code examples for the other stamps.

3. When recording, care must be used to prevent recording beyond the end of sound memory. Should this occur, you must place an EOM marker at the end of you message. To do this, write the code below and execute it once. After that, the message should play normally.

```
pause 500
serout 4,188+$8000,["!SM0I"]
pause 100
serout 4,188+$8000,["!SM0M",149]
```

4. Please note that the **0** in the **SM0P** command is a numeric (zero) and not an alphabet character. Care must be taken since the two characters look alike in many fonts. The zero has no function in this AppMod. Others AppMods may use this character location to specify a module number such that multiple AppMods of the same type may share an I/O line.

5. When using the line input for recording, please limit the input voltage to 90 mVpp. Higher voltages will cause distorted recordings and may damage the Sound Module.

6. To record multiple messages in pushbutton mode, you must send a serial command (the 'M' command) to set the address pointer in the sound chip after enabling the record mode. Write the program such that it sends these only once. Then you may record the message manually. Since the address pointer is still set to the beginning of the message, you may switch modes and listen to what you have just recorded. If it is satisfactory to you, you should then change the host program to send an 'M' command with the next address to record to (again, send this command only once). Note: another way to accomplish this is to record all of the sounds as one continuous recording. Then you may use the 'M' command to surgically divide the initial message into many smaller ones. When playing back a message, send a command to start playing at a particular location, then wait for a period of time. You must wait for the message to finish playing before sending the command to play a different message. However, if you wish to end the playback cycle earlier, you may send an 'I' command; it will end the playback cycle, but there will be an audible click. By understanding the functions of these commands, one can record/playback messages in any order.

7. If the sound module seems to ignore every message you send it, it is probably due to a 'not-so-obvious' syntax error: `serout 4,84+$8000,["!SMOP",0]`. Can you spot the error? The header **SMOP** should be **SM0P** (ess em zero pea) not **SMOP** (ess em oh pea). Depending on what font you use, **0**s and Os can look exactly alike. See note 4.

8. It is normal to hear a 'click' at the end of any message recorded in manual mode. The electret microphone is very sensitive, and is picking up the sound of the pushbutton being released when recording is completed. Of course, if you use the line input instead, no click will be heard. If you hear a click at the beginning of a message that is played back, it is because the device **was** overflowed. The Sound Module checks for an overflow condition every time it plays a message and, if indeed it is in overflow, will automatically reset the ISD chip. It is the reset operation that produces the click sound.