AcPinger: Programmable pinger

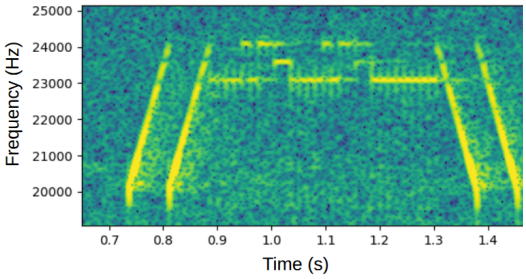
Document Version: 1.1

Updated: 2/11/26
Erin Fischell, PhD
Question? Contact: support@acbotics.com

System Description

Affordable, programmable, open architecture 15-30 kHz pinger for localization and one-way, low-rate data transmission e.g. device ID. Serial interface to configure CWs, LFMs, and FSK data bytes; system can be pre-configured with a ping sequence and duty cycle, or commanded via serial in real time. 124 dB re 1 uPa at 1 m. 50 m depth rating.

Programmable Signal
Signals are programmed for duty cycled send or commanded in real time via serial. Multiple signals may be transmitted as part of a sequence per duty cycle.



Signal Type	Command options	Config Variables
FSK	Fixed value byte, duty cycled Sensor derived byte, duty cycled Serial commanded	High Frequency Low Frequency Bit Duration
LFM	Fixed configuration, duty cycled Serial commanded	Start Frequency Stop Frequency Duration
CW	Fixed configuration, duty cycled Serial commanded	Frequency Duration

Standard Configuration
The AcPinger comes with a connector for sequence configuration, battery charging, and tethered operation. OEM and other configurations available on request.

Property	Value	Notes
Frequency	15-30 kHz	May be used 6-40 kHz with SPL loss.
Dimensions	11” long, 3.5” max diameter	Other housings available on request.
Amplitude	124 dB re 1 uPa 1m	At 25 kHz.

Battery	Lithium rechargeable or NiMH rechargeable.	Extended battery housings available.
Battery Life	48 hours at 10% duty cycle.	Depends on signal sequence and duty cycle.
Sensors	Blue Robotics Bar30	Other sensors can be integrated for use with FSK broadcast, e.g. IMU, DO, conductivity.
Console and charging	Cobalt 6 connector on endcap.	Connect to shore cable for charging and console.

Standard Operation

Opening the housing

1) Unscrew ring	2) Pull two sides apart until full disengaged	3) Place on surface.

Closing the housing

1) Align and insert transducer-end rack into housing	2) Ensuring no wires are in the seal, firmly hold the two sides of the face seal.	3) Screw on ring until fully tight.

Charging battery

AcPinger-Science-3	AcPinger-Basic
1. Unplug dummy plug 2. Plug in console/charge cable 3. Connect wallwort to wall; battery will begin charging.	1. Open housing 2. Plug USB-C cable into battery 3. Plug other end into USB power; battery will begin charging

Turning On/Off

AcPinger-Science3

For 3-penetrator AcPinger science units (photo below) which include external switch:

Turning ON: Turn the On/OFF switch clockwise until the LED comes on.
Turning OFF: Turn the On/OFF switch clockwise until the LED turns off.

AcPinger-Basic

Turning ON		
1) Open housing	2) Connect Battery Cable	3) Re-seal housing

Turning OFF		
1) Open housing	2) Disconnect Battery Cable	3) Re-seal housing

Configuring

For units with external connector (AcSense-Science2 and AcSense-Science3):

1) Unplug dummy plug
2) Plug in provided console/charging cable

For units without external connector:

1. Open housing
2. Plug in provided serial converter cable

Structures/Types within code

Structure	Description	Init code

State	Holds config, the active sequence, the mode, and the sleep-between flag; *config:* see Config structure *active_sequence:* sets ID for which sequence will be run by default *mode:* 0=IDLE, 1=ONCE or 2=REPEAT *sleep_between:* sleep-between puts the system into deep sleep between pings (warning: if sleep_between=1, it makes serial hard to use)	typedef struct __attribute__((packed)){ config_t config; uint32_t active_sequence; pinger_mode_t mode; bool sleep_between; }state_t;
Config	Holds 3 sequences sequences: sequence structure array *Default_active_sequence: i*ndex of which sequence to run on boot *Default_mode*: mode to use on boot 0=IDLE, 1=ONCE or 2=REPEAT *Default_sleep_between:* sleep-between puts the system into deep sleep between pings (warning: if sleep_between=1, it makes serial hard to use) *Valid:* fingerprint that shows that a config has been properly applied.	#define NUM_SEQUENCES 3 typedef struct __attribute__((packed)) { sequence_t sequences[NUM_SEQUENCES]; uint32_t default_active_sequence; pinger_mode_t default_mode; bool default_sleep_between; uint32_t valid; } config_t;
Sequence	Defines up to 16 messages (i.e. signals) to include in a sequence. *Msgs:* message structure array of messages to include in sequence; always 16 long *Num_messages:* how many to actually use (e.g. if you only want 3 messages, use 3) *delay_ms:* how long to delay after end the sequence to play again if in repeat mode	#define MAX_MESSAGES 16 typedef struct __attribute__((packed)) { pinger_message_t msgs[MAX_MESSAGES]; uint32_t num_messages; uint32_t delay_ms; } sequence_t;
Message	Structure defining a single message/signal within a sequence *Start_frequency:* 1) Sweep_linear or sweep_hyperbolic: frequency to start a sweep 2) Data: zero bit of FSK *Stop_frequency:* 1) Sweep_linear or sweep_hyperbolic: frequency to end a sweep 2) Data: one bit of FSK *Length_ms:* 1) Sweep_linear or sweep_hyperbolic: length of the sweep 2) Data: length of one bit of the FSK *Msg_type:* 0=MSG_UNKNOWN: failure/error state 1=MSG_SWEEP_LINEAR: linear frequency modulated chirp (note: CW can be sent by setting start and stop the same) 2= MSG_DATA: Frequency shift keying (FSK) data message 3=MSG_SWEEP_HYPERBOLIC: hyperbolic frequency modulated chirp	typedef struct __attribute__((packed)) { int32_t start_frequency; int32_t stop_frequency; uint32_t length_ms; e_msg_type msg_type; uint8_t data; } pinger_message_t;

Serial Interface

NOTE: all interrupts are disabled while the system is running/sending pings. Ensure you see an echo in the console for smooth operation.

Subsystem: Config

All config commands begin with :C and are used to set configuration parameters that can then be written to non-volatile memory; once written, config will be preserved on re-boot.

Command	**Example pipe to serial**	**Description**
CW	:CW	Write the current configuration into non-volitile memory. **Unless this command is run, config changes made in the console will be lost in reset!!!**
CP	:CP	Print the current config out of RAM
CD	:CD	Load the default manufacturer's configuration into RAM
CL	:CL	Reload the config from non-volatile memory into RAM. **THIS will overwrite any changes you made in the console!!!**
CB	:CB	Enable using pressure sensor for sensor data transmission; reads pressure sensor each time before broadcasting. Must send this before using pressure message. Cannot be turned off without restoring defaults.
CM=<0/1/2/3>	:CM=0 :CM=1 :CM=2	Sets default mode; 0=IDLE, 1=SINGLE, 2=REPEAT
CA=<active sequence>	:CA=1 :CA=2	Sets active sequence, 1=sequence 1, 2=sequence 2, 3=sequence 3 as

	:CA=3	defined in sequence config array
C<SEQNO><PARAM>=<VALUE>	:C1S1=25000 (see below for specific meaning of Params and values)	Sets parameter values for a given sequence number. E.g. :C1S1=25000 would set the sequence 1 first message start frequency to 25000.
C<SEQNO>S<MSGIDX>=<START FREQ>	:C1S1=25000	Set start frequence for a message index in a given sequence.
C<SEQNO>P<MSGIDX>=<STOP FREQ>	:C1S1=28000	Set end frequency for a message index in a given sequence.
C<SEQNO>X<MSGIDX>=<base 10 value up to 255>	:C1X1=123	Set the byte value to send. For message type 5, the bottom 4 bits only will be sent as a id.
C<SEQNO>T<MSGIDX>=<time in ms>	:C1T1=100	Set length in ms; for sweep_linear or sweep_hyperbolic this is chirp length of the sweep, for data this is length of one bit of the FSK
C<SEQNO>F<MSGIDX>=<Message Type>	:C1F1=1	Set message type for a message index in a given sequence. 1=MSG_SWEEP_LINEAR: linear frequency modulated chirp (note: CW can be sent by setting start and stop the same) 2= MSG_DATA: Frequency shift keying (FSK) data message 3=MSG_SWEEP_HYPERBOLIC: hyperbolic frequency modulated chirp 4=Pressure in dBar absolute; also run :CB command to transmit pressure 5=Checksum/id combination, with top 4 bits checksum of any FSK that previously occurred in the sequence not including itself, bottom 4 bits are bottom 4 bits of what you set for that message. 6=Message count, running 8 bit counter. This will transmit a send number 0-255 then roll over.
C<SEQNO>N=<Num messages>	:C1N=4	Explicitly set number of messages to use; this lets you shrink the number of messages being sent.
C<SEQNO>D=<Delay in ms>	:C1D=1000	Set delay in ms between sequences when in repeat mode

NOTE: if you indicate a message index beyond the num_message defined for the sequence, the num_messages parameter will be adjusted to included the additional message set.

Misc. Interface- Serial

Command	Example pipe to serial	Description
?	:?	Print out help message, then print state information and config information from RAM
C	:C	Config Subsystem
M<NUM>	:M0 :M1 :M2	Sets mode system is currently in: 0=IDLE, 1=Once, 2=REPEAT **Note: does not affect config, just sets current mode**
V	:V	Returns voltage if voltage monitoring is onboard
W	:W	Read water detect signal if installed
T	:T	Read the temperature if sensor is installed
R	:R	Reads the read switch state if installed
L	:L	Activate sleep_between state **Note: does not affect config, just sets current state** **NOTE: your console will likely stop working if you run this command**

Example config:

Example configuration to set up a depth pinger with checksum and counter: pipe the following to the serial console of the pinger.

:CB

:C1S1=20000
:C1P1=24000
:C1T1=75
:C1F1=1
:C1S2=20000
:C1P2=24000
:C1T2=75
:C1F2=1

:C1S3=23000
:C1P3=24000
:C1T3=15
:C1F3=4

:C1S4=23500
:C1P4=23500
:C1T4=30
:C1F4=1

:C1S5=23000
:C1P5=24000
:C1T5=15
:C1F5=6

:C1S6=23500
:C1P6=23500
:C1T6=30

:C1S7=23000
:C1P7=24000
:C1T7=15
:C1F7=5

:C1P8=20000

:C1S8=24000

:C1T8=75

:C1P9=20000

:C1S9=24000

:C1T9=75