SwiftNav Binary Protocol

Protocol Specification v0.15

Contents

The Swift Navigation Binary Protocol (SBP) is a fast, simple, and minimal binary protocol for communicating with Swift devices. It is the native binary protocol used by the Piksi GPS receiver to transmit solutions, observations, status and debugging messages, as well as receive messages from the host operating system, such as differential corrections and the almanac. As such, it is an important piece of interfacing with your Piksi receiver and integrating it with other systems.

This document provides language-agnostic specification and documentation for messages used with SBP. SBP client libraries in a variety of programming languages are available at http://docs.swiftnav.com/wiki/SwiftNav_Binary_Protocol.

1 Message Structure

SBP consists of two pieces: (i) an over-the-wire message framing format and (ii) structured payload definitions. As of Version 1.0, the packet consists of a 6-byte binary header section, a variable-sized payload field, and a 16-bit CRC value. SBP uses the CCITT CRC16 (XMODEM implementation) for error detection.

Name	Size	Description
Preamble	1	Denotes the start of frame transmission. Always 0x55.
Message Type	2	Identifies the payload contents.
Sender	2	A unique identifier of the sending hardware. Set to the 2 least significant bytes of the Piksi serial number.
Length	1	Length in bytes of the Payload field.
Payload	N	Binary data of the message.
CRC	2	Cyclic Redundancy Check of the packet's binary data from the Message Type up to the end of Payload (does not include the Preamble).
	$N+8$	

Table 1.0.1: Swift Binary Protocol message structure

2 Basic Formats and Payload Structure

The binary payload of an SBP message can be decoded into structured data based on the message type defined in the header. SBP uses several primitive numerical and collection types for defining the contents of these payloads:

Name	Size	Description
s8	1	Signed 8-bit integer
s16	2	Signed 16-bit integer
s32	4	Signed 32-bit integer
s64	8	Signed 64-bit integer
u8	1	Unsigned 8-bit integer
u16	2	Unsigned 16-bit integer
u32	4	Unsigned 32-bit integer
u64	8	Unsigned 64-bit integer
bool	—	Boolean
float	4	Single-precision float
double	8	Double-precision float
array	—	Fixed or variable length array of any fill type
bytes	—	Fixed or variable length array of bytes
string	—	Fixed or variable length string (NULL terminated)
bitfield	—	A primitive type, such as a u8, can encode boolean status flags.

Table 2.0.2: SBP primitive types

As an example, consider this deframed series of bytes read from a serial port:

`55 02 02 cc 04 14 70 3d d0 18 cf ef ff ff ef e8 ff ff f0 18 00 00 00 00 05 00 43 94`

This cryptic byte array decodes into a `MSG_BASELINE_ECEF` (see pg. 7), which reports the baseline position solution of the rover receiver relative to the base station receiver in Earth Centered Earth Fixed (ECEF) coordinates. The segments of this byte array and its contents breakdown as follows:

Field Name	Type	Value	Bytestring Segment
Preamble	u8	0x55	55
Message Type	u16	0x0202	02 02
Sender	u16	0x4cc	cc 04
Length	u8	20	14
Payload		—	70 3d d0 18 cf ef ff ff ef e8 ff ff f0 18 00 00 00 00 05 00
`MSG_BASELINE_ECEF`			
.tow	u32	416300400 sec	70 3d d0 18
.x	s32	−4145 mm	cf ef ff ff
.y	s32	−5905 mm	ef e8 ff ff
.z	s32	6384 mm	f0 18 00 00
.accuracy	u16	0	00 00
.nsats	u8	5	05
.flags	u8	0	00
CRC	u16	0x9443	43 94

Table 2.0.3: SBP breakdown for `MSG_BASELINE_ECEF`

3 Message Types

Packages define a logical collection of SBP messages. By convention, the contents and layout of messages in packages marked **stable** are unlikely to change in the future. On the other hand, the contents of **draft** messages will change with future development, and are included here purely for informational purposes only. Many draft messages are implementation-defined, and some collections, such as the bootloader package, are intended for internal development.

Package	Message	Name	Size	Description
Stable				
Navigation	0x0100	MSG_GPS_TIME	11	GPS Time
	0x0206	MSG_DOPS	14	Dilution of Precision
	0x0200	MSG_POS_ECEF	32	Single-point position in ECEF
	0x0201	MSG_POS_LLH	34	Geodetic Position
	0x0202	MSG_BASELINE_ECEF	20	Baseline Position in ECEF
	0x0203	MSG_BASELINE_NED	22	Baseline in NED
	0x0204	MSG_VEL_ECEF	20	Velocity in ECEF
	0x0205	MSG_VEL_NED	22	Velocity in NED
System	0xFF00	MSG_STARTUP	4	System start-up message
	0xFFFF	MSG_HEARTBEAT	4	System heartbeat message
Draft				
Acquisition	0x0015	MSG_ACQ_RESULT	13	Satellite acquisition result
Bootload	0x00B0	MSG_BOOTLOADER_HANDSHAKE	1	Bootloading handshake
	0x00DD	MSG_NAP_DEVICE_DNA	8	Send FPGA device DNA over UART
File Io	0x00A8	MSG_FILEIO_READ	25	Read file from the file system
	0x00A9	MSG_FILEIO_READ_DIR	24	List files in a directory
	0x00AC	MSG_FILEIO_REMOVE	20	Delete a file from the file system
	0x00AD	MSG_FILEIO_WRITE	24	Write to file
Flash	0x00E0	MSG_FLASH_PROGRAM	5	Program addresses of the STM or M25 flash
	0x00E0	MSG_FLASH_DONE	1	Flash response message
	0x00E1	MSG_FLASH_READ	5	Read STM or M25 flash address
	0x00E2	MSG_FLASH_ERASE	2	Erase sector of Piksi flash memory
	0x00E3	MSG_STM_FLASH_LOCK_SECTOR	1	Lock sector of STM flash memory
	0x00E4	MSG_STM_FLASH_UNLOCK_SECTOR	1	Unlock sector of STM flash memory
	0x00E5	MSG_STM_UNIQUE_ID	12	Read STM32F4's hardcoded unique ID
Logging	0x0010	MSG_PRINT	1	Plaintext logging messages
Observation	0x0045	MSG_OBS	$13N+20$	GPS satellite observations
	0x0044	MSG_BASE_POS	24	Base station position
Piksi	0x00B2	MSG_RESET	0	Reset the device
	0x0023	MSG_INIT_BASE	0	Initialize IAR from known baseline
	0x0017	MSG_THREAD_STATE	26	State of a CPU/RTOS thread
	0x0018	MSG_UART_STATE	58	State of the UART channels
Settings	0x00A0	MSG_SETTINGS	1	Read/write Piksi configuration settings
	0x00A1	MSG_SETTINGS_SAVE	0	Save settings to flash
Tracking	0x0016	MSG_TRACKING_STATE	$6N+6$	Satellite tracking channel states
	0x001A	MSG_EPHEMERIS	175	WGS84 satellite orbit ephemeris parameters

Table 3.0.4: SBP message types

4 Stable Message Definitions

4.1 Navigation

Geodetic navigation messages reporting GPS time, single-point position, and RTK baseline position solutions.

MSG_GPS_TIME — 0x0100

This message reports the GPS time, an integer time scale beginning at January 6, 1980 midnight. GPS time counts the weeks and seconds of the week. The weeks begin at the Saturday/Sunday transition. GPS week 0 began at the beginning of the GPS time scale. Within each week number, the GPS time of the week is between between 0 and 604800 seconds (=60*60*24*7).

Offset	Size	Format	Units	Name	Description
0	2	u16	weeks	`wn`	GPS week number
2	4	u32	ms	`tow`	GPS time of week rounded to the nearest ms
6	4	s32	ns	`ns`	Nanosecond remainder of rounded tow
10	1	u8		`flags`	Status flags (reserved)
	11				

Table 4.1.1: MSG_GPS_TIME 0x0100 message structure

MSG_DOPS — 0x0206

This dilution of precision (DOP) message describes the effect of navigation satellite geometry on positional measurement precision.

Offset	Size	Format	Units	Name	Description
0	4	u32	ms	`tow`	GPS Time of Week
4	2	u16	0.01	`gdop`	Geometric Dilution of Precision
6	2	u16	0.01	`pdop`	Position Dilution of Precision
8	2	u16	0.01	`tdop`	Time Dilution of Precision
10	2	u16	0.01	`hdop`	Horizontal Dilution of Precision
12	2	u16	0.01	`vdop`	Vertical Dilution of Precision
	14				

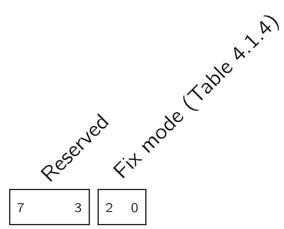
Table 4.1.2: MSG_DOPS 0x0206 message structure

MSG_POS_ECEF — 0x0200

The single-point position solution message reports absolute Earth Centered Earth Fixed (ECEF) coordinates and the status (single point absolute vs RTK) of the position solution. If the rover receiver knows surveyed position of the base station and has an RTK solution, this reports a pseudo-absolute position solution using the base station position and the rover's RTK baseline vector.

Offset	Size	Format	Units	Name	Description
0	4	u32	ms	`tow`	GPS Time of Week
4	8	double	m	`x`	ECEF X coordinate
12	8	double	m	`y`	ECEF Y coordinate
20	8	double	m	`z`	ECEF Z coordinate
28	2	u16	mm	`accuracy`	Position accuracy estimate
30	1	u8		`n_sats`	Number of satellites used in solution
31	1	u8		`flags`	Status flags
	32				

Table 4.1.3: MSG_POS_ECEF 0x0200 message structure



Field 4.1.1: Status flags (`flags`)

Value	Description
0	Single Point Positioning (SPP)
1	Float RTK
2	Fixed RTK

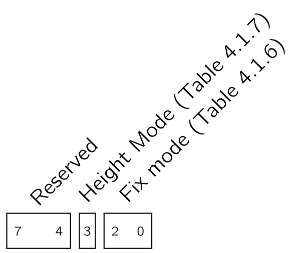
Table 4.1.4: Fix mode values (`flags[0:2]`)

MSG_POS_LLH — 0x0201

This single-point position solution message reports the absolute geodetic coordinates and the status (single point absolute vs RTK) of the position solution. If the rover receiver knows the surveyed position of the base station and has an RTK solution, this reports a pseudo-absolute position solution using the base station position and the rover's RTK baseline vector.

Offset	Size	Format	Units	Name	Description
0	4	u32	ms	`tow`	GPS Time of Week
4	8	double	deg	`lat`	Latitude
12	8	double	deg	`lon`	Longitude
20	8	double	m	`height`	Height
28	2	u16	mm	`h_accuracy`	Horizontal position accuracy estimate
30	2	u16	mm	`v_accuracy`	Vertical position accuracy estimate
32	1	u8		`n_sats`	Number of satellites used in solution
33	1	u8		`flags`	Status flags
	34				

Table 4.1.5: MSG_POS_LLH 0x0201 message structure



Field 4.1.2: Status flags (`flags`)

Value	Description
0	Single Point Positioning (SPP)
1	Fixed RTK
2	Float RTK

Table 4.1.6: Fix mode values (`flags[0:2]`)

Value	Description
0	Height above Ellipsoid
1	Height above mean sea level

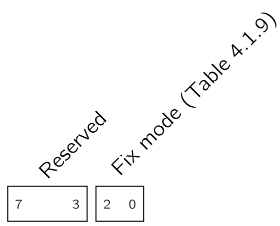
Table 4.1.7: Height Mode values (`flags[3]`)

MSG_BASELINE_ECEF — 0x0202

This message reports the baseline position solution in Earth Centered Earth Fixed (ECEF) coordinates.

Offset	Size	Format	Units	Name	Description
0	4	u32	ms	`tow`	GPS Time of Week
4	4	s32	mm	`x`	Baseline ECEF X coordinate
8	4	s32	mm	`y`	Baseline ECEF Y coordinate
12	4	s32	mm	`z`	Baseline ECEF Z coordinate
16	2	u16	mm	`accuracy`	Position accuracy estimate
18	1	u8		`n_sats`	Number of satellites used in solution
19	1	u8		`flags`	Status flags
	20				

Table 4.1.8: MSG_BASELINE_ECEF 0x0202 message structure



Field 4.1.3: Status flags (`flags`)

Value	Description
0	Float RTK
1	Fixed RTK

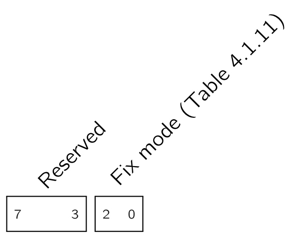
Table 4.1.9: Fix mode values (`flags[0:2]`)

MSG_BASELINE_NED — 0x0203

This message reports the baseline position solution in North East Down (NED) coordinates.

Offset	Size	Format	Units	Name	Description
0	4	u32	ms	`tow`	GPS Time of Week
4	4	s32	mm	`n`	Baseline North coordinate
8	4	s32	mm	`e`	Baseline East coordinate
12	4	s32	mm	`d`	Baseline Down coordinate
16	2	u16	mm	`h_accuracy`	Horizontal position accuracy estimate
18	2	u16	mm	`v_accuracy`	Vertical position accuracy estimate
20	1	u8		`n_sats`	Number of satellites used in solution
21	1	u8		`flags`	Status flags
	22				

Table 4.1.10: MSG_BASELINE_NED 0x0203 message structure



Field 4.1.4: Status flags (`flags`)

Value	Description
0	Float RTK
1	Fixed RTK

Table 4.1.11: Fix mode values (`flags[0:2]`)

MSG_VEL_ECEF — 0x0204

This message reports the velocity in Earth Centered Earth Fixed (ECEF) coordinates.

Offset	Size	Format	Units	Name	Description
0	4	u32	ms	`tow`	GPS Time of Week
4	4	s32	mm/s	`x`	Velocity ECEF X coordinate
8	4	s32	mm/s	`y`	Velocity ECEF Y coordinate
12	4	s32	mm/s	`z`	Velocity ECEF Z coordinate
16	2	u16	mm/s	`accuracy`	Velocity accuracy estimate
18	1	u8		`n_sats`	Number of satellites used in solution
19	1	u8		`flags`	Status flags (reserved)
	20				

Table 4.1.12: MSG_VEL_ECEF 0x0204 message structure

MSG_VEL_NED — 0x0205

This message reports the velocity in local North East Down (NED) coordinates.

Offset	Size	Format	Units	Name	Description
0	4	u32	ms	`tow`	GPS Time of Week
4	4	s32	mm/s	`n`	Velocity North coordinate
8	4	s32	mm/s	`e`	Velocity East coordinate
12	4	s32	mm/s	`d`	Velocity Down coordinate
16	2	u16	mm/s	`h_accuracy`	Horizontal velocity accuracy estimate
18	2	u16	mm/s	`v_accuracy`	Vertical velocity accuracy estimate
20	1	u8		`n_sats`	Number of satellites used in solution
21	1	u8		`flags`	Status flags (reserved)
	22				

Table 4.1.13: MSG_VEL_NED 0x0205 message structure

4.2 System

Standardized system messages from Swift Navigation devices.

MSG_STARTUP — 0xFF00

The system start-up message is sent once on system start-up. It is intended to be used to notify the host or other attached devices that the system has started and is now ready to respond to commands or configuration requests.

Offset	Size	Format	Units	Name	Description
0	4	u32		`reserved`	Reserved
	4				

Table 4.2.1: MSG_STARTUP 0xFF00 message structure

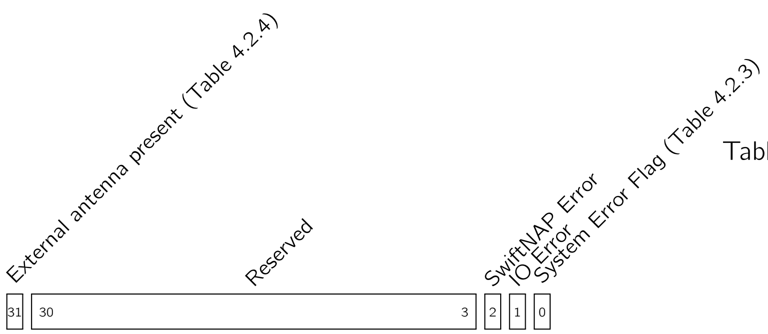
MSG_HEARTBEAT — 0xFFFF

The heartbeat message is sent periodically to inform the host or other attached devices that the system is running. It is intended to be used to monitor for system malfunctions and also contains status flags that indicate to the host the status of the system and if it is operating correctly.

The system error flag is used to indicate that an error has occurred in the system. To determine the source of the error the remaining error flags should be inspected.

Offset	Size	Format	Units	Name	Description
0	4	u32		`flags`	Status flags
	4				

Table 4.2.2: MSG_HEARTBEAT 0xFFFF message structure



Field 4.2.1: Status flags (`flags`)

Value	Description
0	System Healthy
1	An error has occurred

Table 4.2.3: System Error Flag values (`flags[0]`)

Value	Description
0	No external antenna detected
1	External antenna is present

Table 4.2.4: External antenna present values (`flags[31]`)

5 Draft Message Definitions

5.1 Acquisition

Satellite acquisition messages from the Piksi.

MSG_ACQ_RESULT — 0x0015

This message describes the results from an attempted GPS signal acquisition search for a satellite PRN over a code phase/carrier frequency range. It contains the parameters of the point in the acquisition search space with the best signal-to-noise (SNR) ratio.

Offset	Size	Format	Units	Name	Description
0	4	float		`snr`	SNR of best point
4	4	float	chips	`cp`	Code phase of best point
8	4	float	hz	`cf`	Carrier frequency of best point
12	1	u8		`prn`	PRN identifier of the satellite signal for which acquisition was attempted
	13				

Table 5.1.1: MSG_ACQ_RESULT 0x0015 message structure

5.2 Bootload

Messages for the bootloading configuration on the Piksi. These are in the implementation-defined range (0x0000-0x00FF), and intended for internal-use only. Note that some of these messages taking a request from a host and a response from the Piksi share the same message type ID.

MSG_BOOTLOADER_HANDSHAKE — 0x00B0

The bootloader continually sends a handshake message to the host for a short period of time, and then jumps to the firmware if it doesn't receive a handshake from the host. If the host replies with a handshake the bootloader doesn't jump to the firmware and nwaits for flash programming messages, and the host has to send a MSG_BOOTLOADER_JUMP_TO_APP when it's done programming. On old versions of the bootloader (less than v0.1), hardcoded to 0. On new versions, return the git describe string for the bootloader build.

Offset	Size	Format	Units	Name	Description
0	1	u8		`handshake`	Handshake value
	1				

Table 5.2.1: MSG_BOOTLOADER_HANDSHAKE 0x00B0 message structure

MSG_NAP_DEVICE_DNA — 0x00DD

The device DNA message from the host reads the unique device DNA from the Swift Navigation Acceleration Peripheral (SwiftNAP), a Spartan 6 FPGA. By convention, the host message buffer is empty; the Piksi returns the device DNA in a MSG_NAP_DEVICE_DNA message.

Offset	Size	Format	Units	Name	Description
0	8	u8[8]		`dna`	57-bit SwiftNAP FPGA Device DNA
	8				

Table 5.2.2: MSG_NAP_DEVICE_DNA 0x00DD message structure

5.3 File Io

Messges for using Piksi's onboard flash filesystem functionality from the Contiki project. This allows data to be stored persistently in the microcontroller's program flash with wear-levelling using a simple filesystem interface. The Contiki file system interface (CFS) defines an abstract API for reading directories and for reading and writing files. These are in the implementation-defined range (0x0000-0x00FF), and intended for internal-use only. Note that some of these messages taking a request from a host and a response from the Piksi share the same message type ID.

MSG_FILEIO_READ — 0x00A8

The file read message reads a certain length (up to 255 bytes) from a given offset into a file, and returns the data in a MSG_FILEIO_READ message where the message length field indicates how many bytes were succesfully read. If the message is invalid, a followup MSG_PRINT message will print "Invalid fileio read message".

Offset	Size	Format	Units	Name	Description
0	4	u32	bytes	`offset`	File offset
4	1	u8	bytes	`chunk_size`	Chunk size to read
5	20	string		`filename`	Name of the file to read from (NULL terminated)
	25				

Table 5.3.1: MSG_FILEIO_READ 0x00A8 message structure

MSG_FILEIO_READ_DIR — 0x00A9

The read directory message lists the files in a directory on the Piksi's onboard flash file system. The offset parameter can be used to skip the first n elements of the file list. Returns a MSG_FILEIO_READ_DIR message containing the directory listings as a NULL delimited list. The listing is chunked over multiple SBP packets and the end of the list is identified by an entry containing just the character 0xFF. If message is invalid, a followup MSG_PRINT message will print "Invalid fileio read message".

Offset	Size	Format	Units	Name	Description
0	4	u32		`offset`	The offset to skip the first n elements of the file list
4	20	string		`dirname`	Name of the directory to list
	24				

Table 5.3.2: MSG_FILEIO_READ_DIR 0x00A9 message structure

MSG_FILEIO_REMOVE — 0x00AC

The file remove message deletes a file from the file system. If message is invalid, a followup MSG_PRINT message will print "Invalid fileio remove message".

Offset	Size	Format	Units	Name	Description
0	20	string		`filename`	Name of the file to delete (NULL terminated)
	20				

Table 5.3.3: MSG_FILEIO_REMOVE 0x00AC message structure

MSG_FILEIO_WRITE — 0x00AD

The file write message writes a certain length (up to 255 bytes) of data to a file at a given offset. Returns a copy of the original MSG_FILEIO_WRITE message to check integrity of the write. If message is invalid, a followup MSG_PRINT message will print "Invalid fileio write message".

Offset	Size	Format	Units	Name	Description
0	20	string		`filename`	Name of the file to write to (NULL terminated)
20	4	u32	bytes	`offset`	Offset into the file at which to start writing in bytes
	24				

Table 5.3.4: MSG_FILEIO_WRITE 0x00AD message structure

5.4 Flash

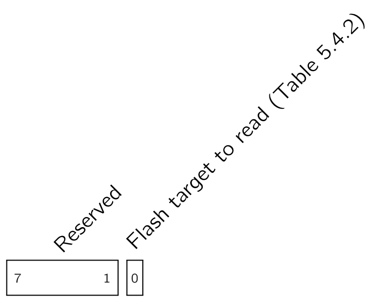
Messages for reading/writing the Piksi's onboard flash memory. These are in the implementation-defined range (0x0000-0x00FF), and largely intended for internal-use only.

MSG_FLASH_PROGRAM — 0x00E0

The flash program message programs a set of addresses of either the STM or M25 flash. The Piksi replies with either a MSG_FLASH_DONE message containing the return code FLASH_OK (0) on success, or FLASH_INVALID_LEN (2) if the maximum write size is exceeded. Note that the sector-containing addresses must be erased before addresses can be programmed.

Offset	Size	Format	Units	Name	Description
0	1	u8		`target`	Target flags
1	3	u8[3]	bytes	`addr_start`	Starting address offset to program
4	1	u8	bytes	`addr_len`	Length of set of addresses to program, counting up from starting address
	5				

Table 5.4.1: MSG_FLASH_PROGRAM 0x00E0 message structure



Field 5.4.1: Target flags (`target`)

Value	Description
0	FLASH_STM
1	FLASH_M25

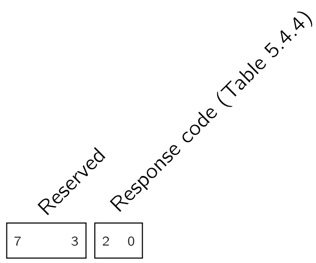
Table 5.4.2: Flash target to read values (`target[0]`)

MSG_FLASH_DONE — 0x00E0

This message defines success or failure codes for a variety of flash memory requests from the host to the Piksi. Flash read and write messages, such as MSG_FLASH_READ or MSG_FLASH_WRITE, may return this message on failure.

Offset	Size	Format	Units	Name	Description
0	1	u8		`response`	Response flags
	1				

Table 5.4.3: MSG_FLASH_DONE 0x00E0 message structure



Field 5.4.2: Response flags (`response`)

Value	Description
0	FLASH_OK
1	FLASH_INVALID_FLASH
2	FLASH_INVALID_LEN
3	FLASH_INVALID_ADDR
4	FLASH_INVALID_RANGE
5	FLASH_INVALID_SECTOR

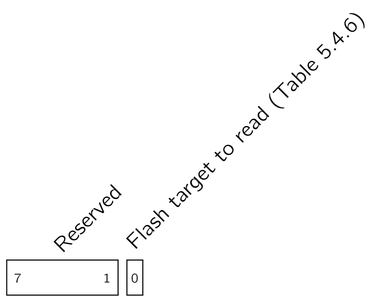
Table 5.4.4: Response code values (`response[0:2]`)

MSG_FLASH_READ — 0x00E1

The flash read message reads a set of addresses of either the STM or M25 onboard flash. The Piksi replies with a MSG_FLASH_READ message containing either the read data on success or a MSG_FLASH_DONE message containing the return code FLASH_INVALID_LEN (2) if the maximum read size is exceeded or FLASH_INVALID_ADDR (3) if the address is outside of the allowed range.

Offset	Size	Format	Units	Name	Description
0	1	u8		`target`	Target flags
1	3	u8[3]	bytes	`addr_start`	Starting address offset to read from
4	1	u8	bytes	`addr_len`	Length of set of addresses to read, counting up from starting address
	5				

Table 5.4.5: MSG_FLASH_READ 0x00E1 message structure



Field 5.4.3: Target flags (`target`)

Value	Description
0	FLASH_STM
1	FLASH_M25

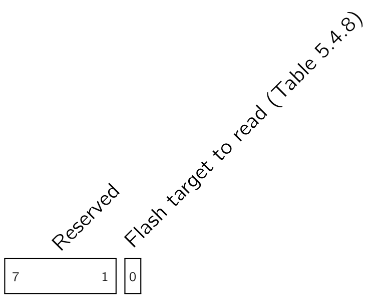
Table 5.4.6: Flash target to read values (`target[0]`)

MSG_FLASH_ERASE — 0x00E2

The flash erase message from the host erases a sector of either the STM or M25 onboard flash memory. The Piksi will reply with a MSG_FLASH_DONE message containing the return code - FLASH_OK (0) on success or FLASH_INVALID_FLASH (1) if the flash specified is invalid.

Offset	Size	Format	Units	Name	Description
0	1	u8		`target`	Target flags
1	1	u8		`sector_num`	Flash sector number to erase (0-11 for the STM, 0-15 for the M25)
	2				

Table 5.4.7: MSG_FLASH_ERASE 0x00E2 message structure



Field 5.4.4: Target flags (`target`)

Value	Description
0	FLASH_STM
1	FLASH_M25

Table 5.4.8: Flash target to read values (`target[0]`)

MSG_STM_FLASH_LOCK_SECTOR — 0x00E3

The flash lock message locks a sector of the STM flash memory. The Piksi replies with a MSG_FLASH_DONE message.

Offset	Size	Format	Units	Name	Description
0	1	u8[1]		`sector`	Flash sector number to lock
	1				

Table 5.4.9: MSG_STM_FLASH_LOCK_SECTOR 0x00E3 message structure

MSG_STM_FLASH_UNLOCK_SECTOR — 0x00E4

The flash unlock message unlocks a sector of the STM flash memory. The Piksi replies with a MSG_FLASH_DONE message.

Offset	Size	Format	Units	Name	Description
0	1	u8[1]		`sector`	Flash sector number to unlock
	1				

Table 5.4.10: MSG_STM_FLASH_UNLOCK_SECTOR 0x00E4 message structure

MSG_STM_UNIQUE_ID — 0x00E5

This message reads the STM32F4's hardcoded unique ID. The Piksi returns STM32F4 unique ID (12 bytes) back to host.

Offset	Size	Format	Units	Name	Description
0	12	string		`stm_id`	STM32F4 unique ID
	12				

Table 5.4.11: MSG_STM_UNIQUE_ID 0x00E5 message structure

5.5 Logging

Logging and debugging messages from the Piksi. These are in the implementation-defined range (0x0000-0x00FF).

MSG_PRINT — 0x0010

This message contains a human-reabable payload string from the Piksi containing errors, warnings and informational messages at ERROR, WARNING, DEBUG, INFO logging levels. These message may also contain information tagged by filename, as well as debug info on function entry/exit when enabled within the firmware.

Offset	Size	Format	Units	Name	Description
0	1	string		`text`	Informative, human-readable string
	1				

Table 5.5.1: MSG_PRINT 0x0010 message structure

5.6 Observation

Satellite observation messages from the Piksi.

MSG_OBS — 0x0045

The GPS observations message reports all the pseudo range and carrier phase observations for the satellites being tracked by the Piksi.

Offset	Size	Format	Units	Name	Description
0	4	u32	ms	`header.t.tow`	Milliseconds since start of GPS week
4	2	u16	week	`header.t.wn`	GPS week number
6	1	u8		`header.n_obs`	Total number of observations. First nibble is the size of the sequence (n), second nibble is the zero-indexed counter (ith packet of n)
$13N + 7$	4	u32	cm	`obs[N].P`	Pseudorange observation
$13N + 11$	4	s32	cycles	`obs[N].L.i`	Carrier phase whole cycles
$13N + 15$	1	u8	cycles / 255	`obs[N].L.f`	Carrier phase fractional part
$13N + 16$	1	u8	dB Hz	`obs[N].cn0`	Carrier-to-Noise density
$13N + 17$	2	u16		`obs[N].lock`	Lock indicator. This value changes whenever a satellite signal has lost and regained lock, indicating that the carrier phase ambiguity may have changed. There is no significance to the value of the lock indicator
$13N + 19$	1	u8		`obs[N].prn`	PRN identifier of the satellite signal
	$13N + 20$				

Table 5.6.1: MSG_OBS 0x0045 message structure

MSG_BASE_POS — 0x0044

This may be the position as reported by the base station itself or the position obtained from doing a single point solution using the base station observations.

Offset	Size	Format	Units	Name	Description
0	8	double	deg	`lat`	Latitude
8	8	double	deg	`lon`	Longitude
16	8	double	m	`height`	Height
	24				

Table 5.6.2: MSG_BASE_POS 0x0044 message structure

5.7 Piksi

System health, configuration, and diagnostic messages specific to the Piksi L1 receiver, including a variety of legacy messages that may no longer be used. These messages are in the implementation-defined range (0x0000-0x00FF), and largely intended for internal-use only.

MSG_RESET — 0x00B2

This message from the host resets the Piksi back into the bootloader. It ensures that all outstanding memory accesses including buffered writes are completed before reset begins.

Offset	Size	Format	Units	Name	Description
	0				

Table 5.7.1: MSG_RESET 0x00B2 message structure

MSG_INIT_BASE — 0x0023

This message initializes the Integer Ambiguity Resolution (IAR) process on the Piksi to use an assumed baseline position between the base station and rover receivers. Warns via MsgPrint if there aren't a shared minimum number (4) of satellite observations between the two.

Offset	Size	Format	Units	Name	Description
	0				

Table 5.7.2: MSG_INIT_BASE 0x0023 message structure

MSG_THREAD_STATE — 0x0017

The thread usage message from the Piksi reports RTOS thread usage statistics for the named thread. The reported values require renormalization.

Offset	Size	Format	Units	Name	Description
0	20	string		`name`	Thread name (NULL terminated)
20	2	u16	Utilization percentage /1000	`cpu`	Percentage cpu use for this thread. Ranges from 0 - 1000 and needs to be renormalized to 100
22	4	u32	kB	`stack_free`	Free stack space for this thread
	26				

Table 5.7.3: MSG_THREAD_STATE 0x0017 message structure

MSG_UART_STATE — 0x0018

The UART message reports data latency and throughput of the UART channels providing SBP I/O. On the default Piksi configuration, UARTs A and B are used for telemetry radios, but can also be be host access ports for embedded hosts, or other interfaces in future.

Offset	Size	Format	Units	Name	Description
0	4	float	kB/s	`uart_a.tx_throughput`	UART transmit throughput
4	4	float	kB/s	`uart_a.rx_throughput`	UART receive throughput
8	2	u16		`uart_a.crc_error_count`	UART CRC error count
10	2	u16		`uart_a.io_error_count`	UART IO error count
12	1	u8	Utilization /255	`uart_a.tx_buffer_level`	UART transmit buffer percentage utilization. Ranges from 0 - 255 and needs to be renormalized to 100
13	1	u8	Utilization /255	`uart_a.rx_buffer_level`	UART receive buffer percentage utilization. Ranges from 0 - 255 and needs to be renormalized to 100
14	4	float	kB/s	`uart_b.tx_throughput`	UART transmit throughput
18	4	float	kB/s	`uart_b.rx_throughput`	UART receive throughput
22	2	u16		`uart_b.crc_error_count`	UART CRC error count
24	2	u16		`uart_b.io_error_count`	UART IO error count
26	1	u8	Utilization /255	`uart_b.tx_buffer_level`	UART transmit buffer percentage utilization. Ranges from 0 - 255 and needs to be renormalized to 100
27	1	u8	Utilization /255	`uart_b.rx_buffer_level`	UART receive buffer percentage utilization. Ranges from 0 - 255 and needs to be renormalized to 100
28	4	float	kB/s	`uart_ftdi.tx_throughput`	UART transmit throughput
32	4	float	kB/s	`uart_ftdi.rx_throughput`	UART receive throughput
36	2	u16		`uart_ftdi.crc_error_count`	UART CRC error count
38	2	u16		`uart_ftdi.io_error_count`	UART IO error count
40	1	u8	Utilization /255	`uart_ftdi.tx_buffer_level`	UART transmit buffer percentage utilization. Ranges from 0 - 255 and needs to be renormalized to 100
41	1	u8	Utilization /255	`uart_ftdi.rx_buffer_level`	UART receive buffer percentage utilization. Ranges from 0 - 255 and needs to be renormalized to 100
42	4	s32	ms	`latency.avg`	Average latency
46	4	s32	ms	`latency.lmin`	Minimum latency
50	4	s32	ms	`latency.lmax`	Maximum latency
54	4	s32	ms	`latency.current`	Smoothed estimate of the current latency
	58				

Table 5.7.4: MSG_UART_STATE 0x0018 message structure

5.8 Settings

Messages for reading and writing the Piksi's device settings. These are in the implementation-defined range (0x0000-0x00FF), and intended for internal-use only. Please see the accompanying description of settings configurations for more details. Note that some of these messages taking a request from a host and a response from the Piksi share the same message type ID.

MSG_SETTINGS — 0x00A0

The setting message reads and writes the Piksi's configuration.

Offset	Size	Format	Units	Name	Description
0	1	string		`setting`	A NULL delimited (and terminated) string, with the A NULL-terminated and delimited string with contents [SECTION_SETTING, SETTING, VALUE] on writes or a series of such strings on reads.
	1				

Table 5.8.1: MSG_SETTINGS 0x00A0 message structure

MSG_SETTINGS_SAVE — 0x00A1

The save settings message persists the Piksi's current settings configuration to its onboard flash memory file system.

Offset	Size	Format	Units	Name	Description
	0				

Table 5.8.2: MSG_SETTINGS_SAVE 0x00A1 message structure

5.9 Tracking

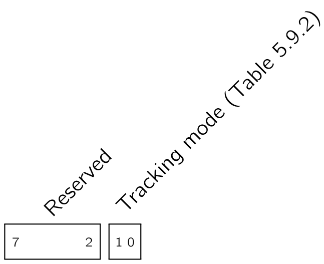
Satellite code and carrier-phase tracking messages from the Piksi.

MSG_TRACKING_STATE — 0x0016

The tracking message returns a variable-length array of tracking channel states. It reports status and code/carrier phase signal power measurements for all tracked satellites.

Offset	Size	Format	Units	Name	Description
$6N + 0$	1	u8		`states[N].state`	Status of tracking channel
$6N + 1$	1	u8		`states[N].prn`	PRN being tracked
$6N + 2$	4	float	dB Hz	`states[N].cn0`	Carrier-to-noise density
	$6N + 6$				

Table 5.9.1: MSG_TRACKING_STATE 0x0016 message structure



Field 5.9.1: Status of tracking channel (`state`)

Value	Description
0	Disabled
1	Running

Table 5.9.2: Tracking mode values (`state[0:1]`)

MSG_EPHEMERIS — 0x001A

The ephemeris message returns a set of satellite orbit parameters that is used to calculate GPS satellite position, velocity, and clock offset (WGS84). Please see the Navstar GPS Space Segment/Navigation user interfaces (ICD-GPS-200, Table 20-III) for more details.

Offset	Size	Format	Units	Name	Description
0	8	double	s	`tgd`	Group delay differential between L1 and L2 (?)
8	8	double	m	`crs`	Amplitude of the sine harmonic correction term to the orbit radius
16	8	double	m	`crc`	Amplitude of the cosine harmonic correction term to the orbit radius
24	8	double	rad	`cuc`	Amplitude of the cosine harmonic correction term to the argument of latitude
32	8	double	rad	`cus`	Amplitude of the sine harmonic correction term to the argument of latitude
40	8	double	rad	`cic`	Amplitude of the cosine harmonic correction term to the angle of inclination
48	8	double	rad	`cis`	Amplitude of the sine harmonic correction term to the angle of inclination
56	8	double	rad/s	`dn`	Mean motion difference
64	8	double	radians	`m0`	Mean anomaly at reference time
72	8	double		`ecc`	Eccentricity of satellite orbit
80	8	double	m^(1/2)	`sqrta`	Square root of the semi-major axis of orbit
88	8	double	rad	`omega0`	Longitude of ascending node of orbit plane at weekly epoch
96	8	double	rad/s	`omegadot`	Rate of right ascension
104	8	double	rad	`w`	Argument of perigee
112	8	double	rad	`inc`	Inclination
120	8	double	rad/s	`inc_dot`	Inclination first derivative
128	8	double	s	`af0`	Polynomial clock correction coefficient (clock bias)
136	8	double	s/s	`af1`	Polynomial clock correction coefficient (clock drift)
144	8	double	s/s^2	`af2`	Polynomial clock correction coefficient (rate of clock drift)
152	8	double	s	`toe_tow`	Time of week
160	2	u16	week	`toe_wn`	Week number
162	8	double	s	`toc_tow`	Clock reference time of week
170	2	u16	week	`toc_wn`	Clock reference week number
172	1	u8		`valid`	Is valid?
173	1	u8		`healthy`	Satellite is healthy?
174	1	u8		`prn`	PRN being tracked
	175				

Table 5.9.3: MSG_EPHEMERIS 0x001A message structure