Extended Serial Protocol
User's Guide

Revision 3, January 11, 2012

VALENTINE RESEARCH, INC. “ESP” SOFTWARE LICENSE AGREEMENT

Thank you for choosing Valentine Research, Inc. ("Valentine", "we," "us," or "our") and our ESP Serial Protocol (the “Software”) and hardware related to the ESP Serial Protocol (the “Hardware) for use with Valentine’s VALENTINE ONE® radar detectors. Please review this Software License Agreement (the “License”) carefully before using the Software and Hardware as described herein. If you do not agree to the terms of this License, do not use the Software or Hardware.

1. General License. Valentine grants you a limited, non-exclusive, non-transferable, royalty-free, worldwide license to download, install, and use the Software or Hardware and any trademarks of Valentine, only as described in this License. The terms of this License will also govern any upgrades to the Software or Hardware, unless such an upgrade is accompanied by a separate license.

2. Permitted License Uses and Restrictions. You may use the Software and Hardware to create third party applications or software designed to be used with or including the Software and/or Hardware, or with Valentine’s products that use the Software or Hardware as a serial protocol. You may not copy, decompile, reverse engineer, disassemble, modify, or create derivative works of the Software or Hardware. You may not make the Software available over a network where it could be used by multiple computers or available to others for downloading and installation. The Software shall not be downloaded, exported or re-exported in violation of any law, including without limitation, any country subject to any applicable U.S. restrictions. The Hardware shall not be exported, or re-exported in violation of any law, including without limitation, any country subject to any applicable U.S. restrictions.

3. Intellectual Property. The Software and any associated proprietary code, inventions, or patents resulting from the Software or Hardware, and all trademarks and trade dress of Valentine remain the property of Valentine. By accessing the Software, you acquire no ownership rights to the Software or Hardware. In the event you use any trademarks of Valentine, you agree to do so only in the manner described in this License. Specifically, you agree only to use the trademarks of Valentine to describe any application or software you create using the Software or Hardware as “Works in connection with Valentine’s ESP Serial Protocol” or similar statement attributing ownership of any trademarks or the Software and Hardware to Valentine. You agree not to do anything that would compromise Valentine’s rights in and to the Software, Hardware, or Valentine’s trademarks.

4. Quality Assurance. Upon request, and with no obligation to return, you agree to provide suitable specimens of any application, software, or other protocol you create using the Software or Hardware to Valentine to verify your compliance with this License.

5. You agree to allow Valentine to use your name and/or product name in an index of products that work with the Software and Hardware and Valentine’s products.

6. AS IS Basis. THE SOFTWARE AND HARDWARE ARE PROVIDED TO YOU ON AN “AS IS” BASIS, AND YOU ARE SOLELY RESPONSIBLE FOR THEIR USE. VALENTINE DISCLAIMS ALL WARRANTIES REGARDING THE SOFTWARE AND HARDWARE, INCLUDING WARRANTIES OF NON-INFRINGEMENT. VALENTINE IS NOT LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, OR PUNITIVE DAMAGES ARISING FROM OR RELATING TO YOUR USE OF THE SOFTWARE OR HARDWARE.

7. Indemnification. You agree to defend, indemnify, and hold harmless Valentine against any and all loss, liability, claims, causes of action, or suits arising out of or related to your use of the Software or Hardware, or any applications, software, or other protocols you create using the Software or Hardware, whatever their nature.

8. Support. Valentine has no obligation to provide any type of support for the Software or Hardware, or for any applications, software, or protocols you create using the Software or Hardware.

9. Termination. Valentine may terminate this License immediately, without notice to you. This License shall also terminate automatically upon your breach of any of the terms and conditions of this License. In the event of termination of this License, you will immediately cease all use of Valentine’s trademarks, and the Software.

10. No Assignment. You may not transfer or assign this License or the Software.

11. Miscellaneous. Any dispute arising under this License shall be resolved exclusively by the state and federal courts of the State of Ohio, applying Ohio law without regard to any conflict of law provisions.

Table of Contents

1 ESP Overview

The Extended Serial Protocol (ESP) is a serial protocol that allows accessories to communicate with either the Valentine One Radar Locator or other accessories. All communication is performed on a single wire. This is accomplished using a method called time slicing, which has strict timing requirements with the Valentine One acting as the bus controller. Using ESP, the Valentine One can provide display data to all accessories as well as respond to requests from accessories to take an action, such as muting the audio or turning off the Valentine One display, or to provide more detailed information about the current threat environment.

Scope

This specification applies to all Valentine Ones with version 3.892 and higher. In this document, ESP capable Valentine Ones and accessories will be referred to as ESP devices. Non-ESP capable Valentine Ones and accessories will be referred to as Legacy devices.

Operating Modes

All ESP devices manufactured by Valentine Research, Inc. will support two modes of communication: Legacy Mode and ESP Mode.

Legacy Mode allows ESP devices to operate on the network with non-ESP devices. In this mode, the Valentine One will output the Concealed Display Output Stream (CDOS) data in the same format used by previous versions. The format is described in *Concealed Display Output Detail.pdf*. While this specification makes one change to the CDOS so that it can be used for device discovery, the operation of Legacy devices remains unchanged. Device discovery is the process that all ESP devices go through to determine what mode the data bus is operating in. Refer to the discovery section for more information.

ESP Mode is the communication protocol discussed in this specification.

Unless noted, all Valentine Research accessories marked with the ESP logo will support both modes and will be able to dynamically switch between modes. Third party accessories may support one or both modes at the manufacturer's discretion.

Hardware Requirements

All devices that support Legacy Mode must be able to provide a pull up resistor on the data line. The line should be pulled up to 1.2V to allow the Valentine One to detect the presence of the accessory. This pull up must be switchable so that it can be turned off when operating in ESP Mode.

All devices that support ESP Mode must have a UART available. The UART's transmit pin (TXD) must be able to switch between TXD output and high impedance. This will allow the UART's receive pin (RXD) to use the same data line (half-duplex mode of operation). The TXD line must be set to high impedance whenever the device is not actively transmitting or when the device is operating in Legacy Mode.

Refer to the ESP Hardware Specification at http://www.valentine1.com/v1info/ESP/PDF/ESP Hardware Specification.pdf for more detailed information on the hardware requirements for an ESP accessory.

Audio Data

The audio data sent by the Valentine One remains unchanged by this specification. The audio format is outside the scope of this document.

Display Data

In Legacy Mode, the display data will be transferred using the format used by non-ESP capable Valentine Ones, as described in *Concealed Display Output Detail.pdf*. The only change to the data format is for discovery mode.

In ESP Mode, the display data will be transmitted in a packet with the infDisplayData packet identifier. The data will be sent at semi-regular intervals and may be interspersed with other packets depending on the current accessory environment.

2 Protocol Description

Serial Format

All ESP communication will take place at 57600 baud using 8 data bits, no parity, 1 stop bit and no flow control.

Packet Format

All ESP communication is done using data packets. A packet consists of six framing bytes and payload data. The payload data is not required for all packets. There are two ESP packet formats that all accessories must support. The format is established by the Valentine One controlling the bus. A Valentine One with a device identifier of $9 uses the Non-Checksum Packet Format. A Valentine One with a device id of $A uses the Checksum Packet Format. Unless specified, all payload lengths discussed in this document assume checksums are being used. All accessories on the bus must use the same packet format used by the Valentine One that is controlling the bus.

Table 2.1 - Non-Checksum Packet Format

Byte No.	Name	Value	Description
0	SOF	$AA	Start of Frame
1	DI	$D0 + Dest ID	Destination Identifier
2	OI	$E0 + Send ID	Originator Identifier
3	PI	$XX	Packet Identifier
4	PL	$XX	Payload Length
5 : PL + 5 - 1	PD	$XX	Payload Data (Not present if PL = 0)
5 + PL	EOF	$AB	End of Frame

Table 2.2 - Checksum Packet Format

Byte No.	Name	Value	Description
0	SOF	$AA	Start of Frame
1	DI	$D0 + Dest ID	Destination Identifier
2	OI	$E0 + Send ID	Originator Identifier
3	PI	$XX	Packet Identifier
4	PL	$XX	Payload Length (including checksum)
5 : PL + 5 - 2	PD	$XX	Payload Data (Not present if PL = 1)
5 + PL - 1	CS	$XX	Packet Checksum[1]
5 + PL	EOF	$AB	End of Frame

Note 1: The checksum is calculated as an 8 bit summation with no attention to carries. The formula is

$$CS = SOF + DI + OI + PI + PL + PD_1 + PD_2 + \ldots + PD_{PL-1}$$

Non-Checksum infDisplayData Packet Example:

AA D8 E9 31 08 5B 5B 1F 38 28 0C 00 00 AB

Checksum infDisplayData Packet Example

AA D8 EA 31 09 5B 5B 1F 38 28 0C 00 00 E7 AB

Packet Naming Convention

- Packets that are used to request data or request that an action be taken are prefixed with '*req*". Examples: *reqVersion, reqSerialNumber, reqBatteryVoltage*.
- Packets that are used to respond to a request are prefixed with '*resp*'. Examples: *respVersion, respSerialNumber, respBatteryVoltage*.
- Packets that provide data without a corresponding request and packets that provide error information are considered informational and are prefixed with '*inf*'. Examples: *infDisplayData, infV1Busy*.

Time Slicing

ESP devices use a technique called time slicing to communicate over the ESP bus. Each accessory is assigned a specific time interval, called a time slice, in which they are allowed to transmit. Accessories may only transmit during their assigned time slice and must be listening for packets the rest of the time. The Valentine One will initiate the time slices by sending out an *infDisplayData* packet. After receiving the EOF (End of Frame byte) in the *infDisplayData* packet from the Valentine One, all accessories will immediately start a timer to wait for their allotted time slice. While waiting for the time slice, the accessory should continue to listen for requests or responses from other accessories. Once the accessory's time slice has arrived, the accessory should respond to any pending requests. Failure to send a response in the next allotted time slice may cause another device to resend the request. If the accessory is not responding to a request during its time slice, it may initiate communication with another device. *The packet transmission should start at the beginning of the time slice and must be completed before the end of the time slice.* Refer to the Time Slice Specification section for a more detailed discussion of time slicing.

Time Slice Hold Off

To insure rapid response time in situations where close audio and display synchronization is desired (i.e. during the sign-on sequence), the Valentine One has the ability to deny all accessories a time slice. This process is called holding off the time slices and is accomplished by setting the *TS Holdoff* bit in the *infDisplayData* packet. When the *TS Holdoff* bit is set, all accessories will ignore their time slices until they receive an *infDisplayData* packet with the *TS Holdoff* bit cleared. This means that while the *TS Holdoff* bit is set, the Valentine One retains control of the bus after the *infDisplayData* packet is sent out and can send out display updates without waiting for the accessories.

Valentine One® Request Processing

The Valentine One will attempt to process all incoming requests as quickly as possible. However, alert processing remains the Valentine One's highest priority, so some requests may be rejected or delayed. If a request is going to be delayed, the Valentine One will respond with an *infV1Busy* packet before the next *infDisplayData* packet. The *infV1Busy* packet will contain a list of up to five pending commands that the Valentine One is working on. Accessories that are waiting for a response are responsible for checking the payload of the *infV1Busy* response. If the accessory's request is pending, the accessory should not send the request again until a response is received or the request is not found in the *infV1Busy* response. If a request is rejected by the Valentine One, the Valentine One will respond with a *respRequestNotProcessed* packet. The accessory should wait until an *infDisplayData* packet is received without a preceding *infV1Busy* packet until reiterating the request.

Valentine One Discovery

An ESP capable Valentine One will automatically determine the type of connected accessories and operate in the highest level possible. This process is called discovery. If there are only ESP accessories connected, the Valentine One will operate in ESP mode. If there are any Legacy devices connected, the Valentine One will operate in Legacy mode.

Because Legacy devices are required to have a pull up on the data line, the Valentine One detects the accessory type by reading the voltage on the data line. If the voltage is above 1.2 volts then the Valentine One will operate in Legacy mode. If the voltage is below 1.2 volts, the Valentine One will operate in ESP mode. In both modes, the Valentine One will periodically read the voltage on the data line to look for a change in the accessory environment.

ESP Mode Discovery

The Valentine One will perform discovery in ESP mode after all accessory time slices have elapsed. This can be done without coordinating with the accessories because at that point all accessories should be acting as receivers with the Valentine One in control of the bus.

Legacy Mode Discovery

The Valentine One must coordinate the discovery attempt when in Legacy mode with the accessories. Since ESP capable accessories will have their pull-ups turned on in Legacy mode, they must turn off their pull-ups for the Valentine One to properly discover any true Legacy accessories attached to the bus. To coordinate with the accessories, the Valentine One will embed a special character in the unused bits of the CDOS to inform the accessory that a discovery attempt is about to occur. The special character will be placed in the bits labeled as '*fill*' in the *Concealed Display Output Detail.pdf* according to Table 2.5.

Table 2.3 - Legacy Mode Check bit locations

CDOS Bit Number	CDOS Bit Name	Discovery Value
33	Fill bit 1	1
34	Fill bit 2	0
35	Fill bit 3	1
36	Fill bit 4	0

When an ESP capable accessory recognizes the *reqModeCheck* request it must temporarily turn off its pull-up resistor. As shown in Figure 4, the pull up circuit must be turned off no later than 1 mS after the rising edge of the CDOS idle period and must remain off for at least 9.58 mS. If no Legacy devices are on the bus the voltage will fall below the threshold and the Valentine One will switch to ESP mode. Therefore if an ESP accessory receives ESP data while its pull-up is turned off, the accessory should enter ESP mode (its pull-up will not be turned back on). However, if there are Legacy devices on the bus then the voltage will remain elevated and the Valentine One will stay in Legacy mode. Refer to the *reqModeCheck* packet description for more details.

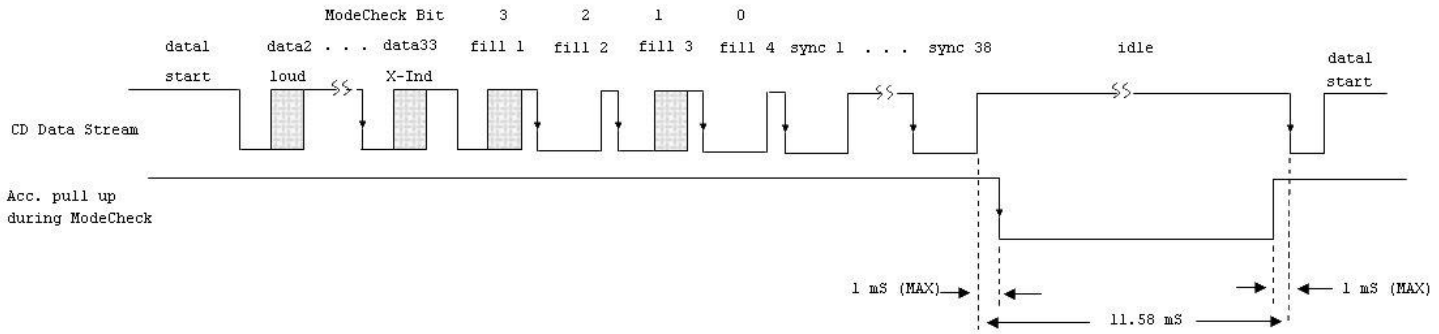


Figure 2.1 - Legacy Mode Check bit locations

Accessory Discovery

Accessories that support both ESP and Legacy modes must be able to change modes dynamically. However, accessories that support only ESP mode do not need to implement the discovery algorithm discussed in this section.

ESP devices should power up in ESP mode. The accessory will then toggle between the two modes until valid data is found or it is determined that the data stream has been lost. Valentine Research recommends waiting for data for 150 mS before changing modes. The flowchart in Appendix 12.2 demonstrates an example implementation that continually monitors for loss of data while still processing the data stream. In that example, the accessory will consider the data stream lost after attempting to find data in both modes twice. When the data is lost, some accessories may require special processing. For example, the Concealed Display must blank its display when the data stream is lost. Once all post-data-loss processing is complete, the accessory should enter a low current sleep mode to prevent battery drain if the device is connected to a non-switched voltage supply. It should be noted that the accessory is in Legacy mode (i.e. pull-ups are turned on) when it enters sleep mode. This is done to allow the accessory to be woken up by a hardware interrupt triggered by either a legacy Valentine One's CDOS or an ESP packet from an ESP capable Valentine One. Refer to the hardware requirements for more information.

Appendix 12.2 also demonstrates the accessory's implementation of the *reqModeCheck* request described above. When a valid CDOS packet is received, the accessory must switch to ESP mode by turning off its pull-ups and enabling the receiver on the UART. This can easily be integrated into the normal data processing and discovery timeout functionality by allowing the timeout interval to be modified. In the example in Appendix 12.2 the timeout starts at 150 mS and will stay that way until a *reqModeCheck* request is received in the CDOS. The accessory will then change the timeout to 9.5 mS and change to ESP mode, just as it would if a 150 mS timeout had occurred. If ESP data is received before the 9.5 mS timeout, the accessory will change the timeout back to 150 mS and stay in ESP mode. If no ESP data is received, the accessory will switch to Legacy mode and change the timeout back to 150 mS and wait for data.

3 Time Slice Specifications

Table 3.1 - Time Slice Parameter Specification

Parameter	Symbol	Value	Units
Number of Time Slices	TS_{Max}	8	
Number of Framing Bytes[1]	$Bytes_{Frame}$	6	Bytes
Maximum Payload Length	PL_{Max}	16	Bytes
Byte Transmission Time[2]	t_{Byte}	173.611	µS
Guard Time	t_{Guard}	173.611	µS
Pacing Time	t_{Pacing}	173.611	µS
Maximum Pacing Time	$t_{PacingMax}$	64	mS
Total Time Slice Time[3]	t_{Slice}	7.639	mS

Notes 1: Framing bytes are *SOF*, *DI*, *OI*, *PI*, *PL* and *EOF*

2: $t_{Byte} = (1/baud) * 10 = (1/57600) * 10 = 173.611$ µS

3: $t_{Slice} = (t_{Byte} * Bytes_{Frame}) + (t_{Byte} * PL_{Max}) + (t_{Pacing} * (Bytes_{Frame} + PL_{Max}))$

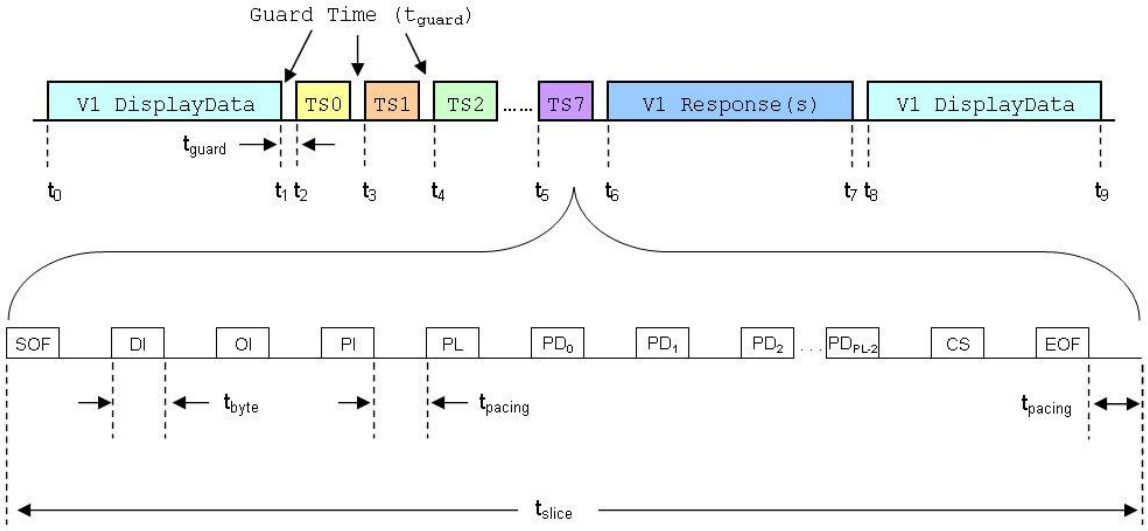


Figure 3.1 - Time Slice Specification

To insure every device has time to process each byte, all devices must use inter-packet pacing, as shown in Figure 3.1. If the time between bytes exceeds $t_{PacingMax}$, the receiving device should abandon the current packet and start looking for the next valid packet, indicated by the *SOF* byte.

Figure 3.1 shows a typical time slice cycle. The cycle starts at t_0 and continues through t_7. The next cycle begins at t_8. Note that the ellipses between TS2 and TS7 represent TS3, TS4, TS5 and TS6, which have been omitted for clarity.

Table 3.2 - A typical ESP cycle.

Time	Event
t0	The Valentine One starts transmitting an *infDisplayData* packet.
t1	The Valentine One finishes transmitting the *infDisplayData* packet and relinquishes control of the bus. All accessories start timers to wait for their assigned time slice.
t2	The device assigned to TS0 has control of the bus and may begin transmitting if necessary.
t3	The device assigned to TS1 has control of the bus and may begin transmitting if necessary.
t4	The device assigned to TS2 has control of the bus and may begin transmitting if necessary.
t5	The device assigned to TS7 has control of the bus and may begin transmitting if necessary.
t6	The Valentine One takes control of the bus and sends out responses or informational packets as needed. There may be 0 or more packets sent. The Valentine One responses and informational packets will be sent out in the order specified in Table 3.3.
t7	The Valentine One finishes sending responses and informational packets. The Valentine One still has control of the bus.
t8	The next ESP cycle starts as the Valentine One starts transmitting an *infDisplayData* packet.

Table 3.3 - Valentine One responses and informational packets transmission order

Output Number	Description
0	All *respRequestNotProcessed* packets
1	All available responses, including *respDataError* packets.
2	Alert Table data packets if the Alert Table data is turned on.
3	Custom Sweep Definitions if they have been requested.
4	*infV1Busy* message if the Valentine One has any pending requests.
5	*infDisplayData* packet.

Time Slice Assignment

Table 3.4 – Time Slice Assignment

Device ID	Description	Time Slice	Guard Start	Slice Start[1]	Slice End[1]
$00	Concealed Display	0	0.000 µS	173.611 µS	7.813 mS
$01	Remote Audio	1	7.813 mS	7.986 mS	15.63 mS
$02	Savvy	2	15.63 mS	15.80 mS	23.44 mS
$03	Available for 3rd party use	3	23.44 mS	23.61 mS	31.25 mS
$04	Available for 3rd party use	4	31.25 mS	31.42 mS	39.06 mS
$05	Available for 3rd party use	5	39.06 mS	39.24 mS	46.88 mS
$06	Reserved for Valentine Research	6	46.88 mS	47.05 mS	54.69 mS
$07	Reserved for Valentine Research	7	54.69 mS	54.86 mS	62.50 mS
$08	General Broadcast[2]	NA	NA	NA	NA
$09	Valentine One without checksums	NA	NA	NA	NA
$0A	Valentine One with checksums	NA	NA	NA	NA

Notes
1: All times are relative to the reception of the EOF byte of an *infDisplayData* packet whose Originator ID (OI) is the Valentine One.
2: The General Broadcast Device ID ($08) can only be used as a Destination Identifier (DI) and may only be used in the *infDisplayData* and *infV1Busy* packets.

Device identifiers three, four and five have been made available for third party devices. All other device identifiers are for use by Valentine Research, Inc. products only. ESP device manufacturers are encouraged to publish the device identifier used by their ESP capable products. This will allow the users to determine which products can be put on the bus at the same time. A useful feature for third party devices would be the ability to allow the user to select the device identifier to be used.

Time Slice Priority

Time slice assignments are in order of increasing priority. If the Valentine One receives requests from two devices during the same time slice cycle, the request from the device with the highest priority will be processed. The request from the device with the lower priority will not be processed and a *respRequestNotProcessed* packet will be sent by the Valentine One.

4 ESP Packet Quick Reference

S = The device may send the packet
P = The device is capable of correctly receiving and processing the packet
blank = The device does not support the packet

Packet Group	Packet Name	Packet ID (PI)	Payload Length[1] (PL)	Concealed Display	Remote Audio	Savvy	Valentine One V3.892
Device Information	reqVersion	$01	1	P	P	P	P
	respVersion	$02	8	S	S	S	S
	reqSerialNumber	$03	1			P	P
	respSerialNumber	$04	11			S	S
User Setup Options	reqUserBytes	$11	1				P
	respUserBytes	$12	7				S
	reqWriteUserBytes	$13	7				P
	reqFactoryDefault	$14	1				P
Custom Sweep	reqWriteSweepDefinition	$15	6				P
	reqAllSweepDefinitions	$16	1				P
	respSweepDefinition	$17	6				S
	reqDefaultSweeps	$18	1				P
	reqMaxSweepIndex	$19	1				P
	respMaxSweepIndex	$20	2				S
	respSweepWriteResult	$21	2				S
	reqSweepSections	$22	1				P
	respSweepSections	$23	6,11 or 16				S
Display and Audio	infDisplayData	$31	9	P	P	P	S
	reqTurnOffMainDisplay	$32	1				P
	reqTurnOnMainDisplay	$33	1				P
	reqMuteOn	$34	1				P
	reqMuteOff	$35	1			S	P
	reqChangeMode	$36	2				P
Alert Output	reqStartAlertData	$41	1				P
	reqStopAlertData	$42	1				P
	respAlertData	$43	8				S
Miscellaneous	respDataReceived	$61	1	S			P
	reqBatteryVoltage	$62	1				P
	respBatteryVoltage	$63	3				S
	respUnsupportedPacket	$64	1			SP	SP
	respRequestNotProcessed	$65	2			P	SP
	infV1Busy	$66	2-6	P	P	P	S
	respDataError	$67	2				S
Savvy Specific	reqSavvyStatus	$71	1			P	
	respSavvyStatus	$72	3			S	
	reqVehicleSpeed	$73	1			P	
	respVehicleSpeed	$74	2			S	
	reqOverrideThumbwheel	$75	2			P	
	reqSetSavvyUnmuteEnable	$76	2			P	

Note 1: All payload lengths include a checksum byte. Subtract 1 from the payload length indicated if checksums are not being used

5 Device Information Packets

reqVersion

The *reqVersion* packet is used to request the firmware version from any ESP capable device.

Valentine One version		V3.892 and above
Packet ID		$01
Payload Length		1
Payload Bytes	0	Checksum

respVersion

The *respVersion* packet is used to respond to a *reqVersion* packet.

Valentine One version		V3.892 and above
Packet ID		$02
Payload Length		8
Payload bytes	0	The version identification letter for the responding device 'V' for Valentine One 'C' for Concealed Display 'R' for Remote Audio 'S' for Savvy
	1	ASCII value of the major version number.
	2	Decimal point ('.')
	3	ASCII value of the minor version number.
	4	ASCII value of the first digit of the revision number.
	5	ASCII value of the second digit of the revision number.
	6	ASCII value of the Engineering Control Number.
	7	Checksum

reqSerialNumber

The *reqSerialNumber* packet is used to request a serial number from any serialized ESP capable accessory. ESP capable accessories without serial numbers are not required to respond to this packet.

Valentine One version		V3.892 and above
Packet ID		$03
Payload Length		1
Payload Bytes	0	Checksum

respSerialNumber

The *respSerialNumber* packet is used to respond to a *reqSerialNumber* packet. Unused characters will be NULL ($00).

Valentine One version		V3.892 and above
Packet ID		$04
Payload Length		11
Payload bytes	0	The first character of the serial number string, in ASCII.
	1	The second character of the serial number string, in ASCII
	2	The third character of the serial number string, in ASCII
	3	The fourth character of the serial number string, in ASCII
	4	The fifth character of the serial number string, in ASCII
	5	The sixth character of the serial number string, in ASCII
	6	The seventh character of the serial number string, in ASCII
	7	The eighth character of the serial number string, in ASCII
	8	The ninth character of the serial number string, in ASCII
	9	The tenth character of the serial number string, in ASCII
	10	Checksum

6 User Configuration

reqUserBytes

The *reqUserBytes* packet is used to request the current user modifiable settings in the Valentine One.

Valentine One version		V3.892 and above
Packet ID		$11
Payload Length		1
Payload Bytes	0	Checksum

respUserBytes

The *respUserBytes* packet is used by the Valentine One to respond to a *reqUserBytes* request. Refer to Appendix 12.1 for a description of the user bytes.

Valentine One version		V3.892 and above
Packet ID		$12
Payload Length		7
Payload bytes	0	User Byte 0
	1	User Byte 1
	2	User Byte 2
	3	User Byte 3
	4	User Byte 4
	5	User Byte 5
	6	Checksum

reqWriteUserBytes

The *reqWriteUserBytes* packet is used by an accessory to update the user configuration settings inside the Valentine One. Refer to Appendix 12.1 for a description of the user bytes.

Valentine One version		V3.892 and above
Packet ID		$13
Payload Length		7
Payload bytes	0	User Byte 0
	1	User Byte 1
	2	User Byte 2
	3	User Byte 3
	4	User Byte 4
	5	User Byte 5
	6	Checksum

reqFactoryDefault

The *reqFactoryDefault* packet is used to reset a device to its factory settings. The effect of the *reqFactoryDefault* packet will vary according to the target device. Refer to the table 6.1 for the packet's effect on each device that complies with this specification.

Valentine One version		V3.892 and above
Packet ID		$14
Payload Length		1
Payload Bytes	0	Checksum

Table 6.1 – Effects of the *reqFactoryDefault* packet

Device ID	Description	*reqFactoryDefault* Effect
$00	Concealed Display	No effect
$01	Remote Audio	No effect
$02	Savvy	Force speed setting to be set by the thumb wheel and re-enable the unmute functionality.
$03	Third party device	Refer to the device manufacturer's documentation
$04	Third party device	Refer to the device manufacturer's documentation
$05	Third party device	Refer to the device manufacturer's documentation
$06	Reserved for VRI	Not applicable
$07	Reserved for VRI	Not applicable
$08	General Broadcast	Not applicable
$09	Valentine One without checksums	Force all settings back to the factory default. This includes the mode, all user settings (refer to the respUserBytes packet) and custom sweep definitions.
$0A	Valentine One with checksums	Force all settings back to the factory default. This includes the mode, all user settings (refer to the respUserBytes packet) and custom sweep definitions.

7 Custom Sweep Packets

All ESP capable Valentine Ones allow the user to define custom Ka sweeps in Euro Mode. If any custom sweeps are defined, the mode character on the front panel will indicate that custom sweeps are being used. If K and Ka bands are enabled, the display will change from ‘U’ to ‘C’. When only Ka band is enabled, the display will change from ‘u’ to ‘c’.

The capabilities of custom sweeps may change with product evolution or software revision, so the user must read the custom sweep parameters from the Valentine One before writing a custom sweep. The first parameter that must be read from the Valentine One is the definition of the sweep sections. The sweep sections are predefined sections of the Ka sweep used for calibration by the Valentine One. When writing a new sweep definition, the user must insure that the custom sweep does not cross a boundary of any sweep section. The second parameter to be read is the number of custom sweeps supported by the Valentine One. Once the Valentine One capabilities are known, the user can write custom sweeps using the *reqWriteSweepDefinition* packet.

All sweeps the user is using must be written back to the Valentine One, even if the sweep is not being changed. Any unwritten sweeps will be turned off. The recommended procedure is

1. Read the sweep section definitions
2. Read the number of supported custom sweeps
3. Read the current custom sweeps
4. Add, remove and modify sweep definitions as needed
5. Write the sweeps that should be used to the Valentine One. The last packet sent must have the *Commit* bit set in the Aux0 byte.
6. Read the sweeps again to determine the actual frequencies selected by the Valentine One and to confirm the intended changes have been applied.

Refer to Example 7.1 for an example of the packet flow when defining custom sweeps. The example shows all operations required for setting up custom sweeps. In this example, the Valentine One starts with the default settings for V3.892, which is two sweep sections, support for six custom sweeps and four custom sweeps defined. The accessory will add a single custom sweep and leave the others unmodified. Note that after this procedure, there will be five custom sweeps defined, so one custom sweep will be turned off. Table 7.1 shows the packets that are transferred in the example.

To insure accuracy in the custom sweeps, the Valentine One does not allow all frequencies in the sweep sections to be used as end points for a custom sweep. Instead, the Valentine One will find the closest calibrated frequency to the requested end point. Therefore it is important for the user to read the custom sweeps back after they are written to determine what frequencies the Valentine One is using for the sweep end points. For example, Table 7.1 shows that the new custom sweep requested at index 4 in Example 7.1 had an upper edge of 36000 MHz and a lower edge of 35500 MHz. However, the new sweep that was created has an upper edge of 36013 MHz and a lower edge of 35541MHz because those are closest calibrated frequencies to the requested sweep end points.

Example 7.1 – Writing a custom sweep

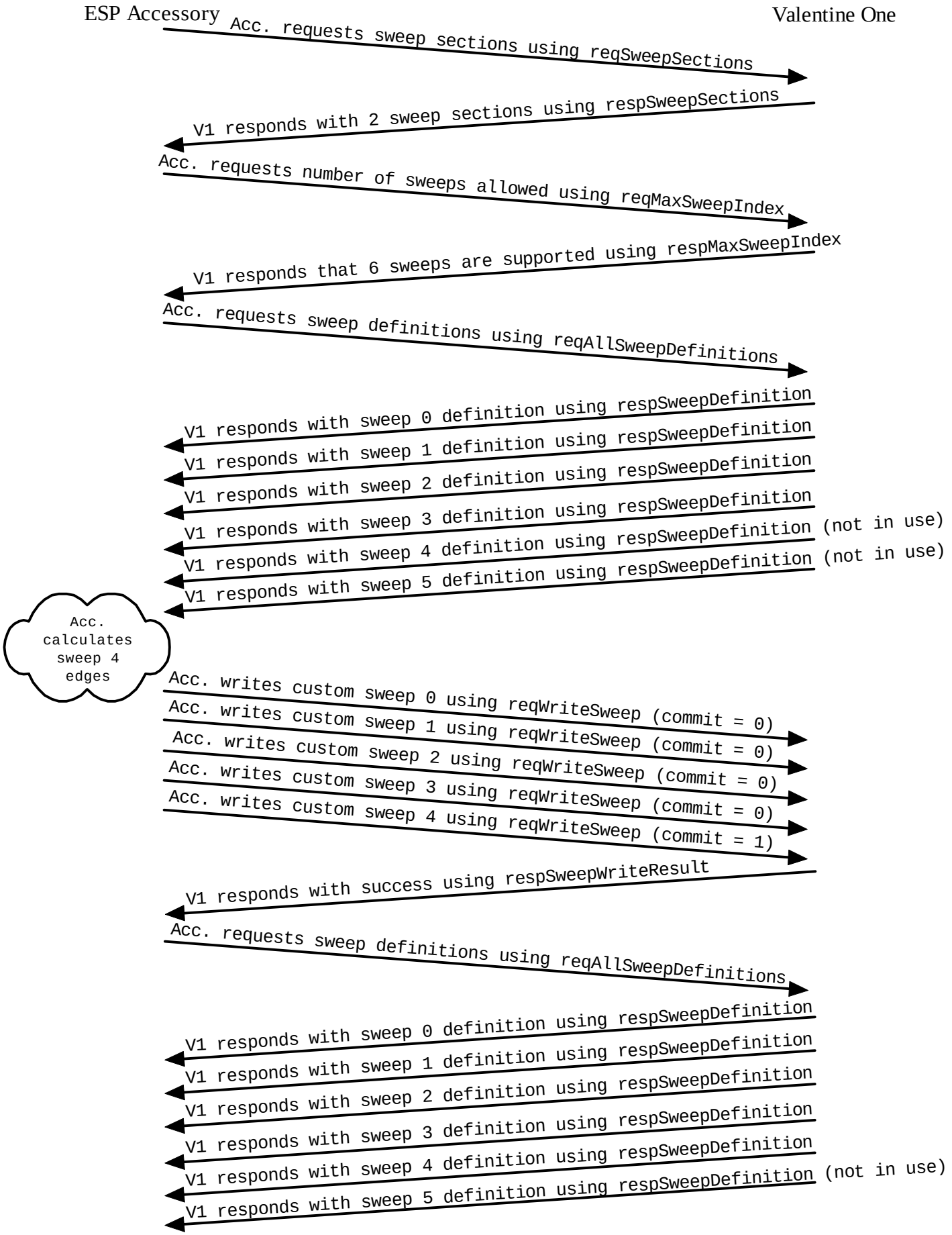


Table 7.1 – Packets for Custom Sweep Writing Example

Event	Packets
Acc. requests sweep sections using reqSweepSections	AA DA E6 22 01 8D AB
V1 responds with 2 sweep sections using respSweepSections	AA D6 EA 23 0B 12 8C E8 87 D6 22 87 D2 82 67 DF AB
Acc. requests number of sweeps allowed using reqMaxSweepIndex	AA DA E6 19 01 84 AB
V1 responds that 6 sweeps are supported using respMaxSweepIndex	AA D6 EA 20 02 05 91 AB
Acc. requests sweep definitions using reqAllSweepDefinitions	AA DA E6 16 01 81 AB
V1 responds with sweep 0 definition using respSweepDefinition	AA D6 EA 17 06 80 85 3A 84 6C B6 AB
V1 responds with sweep 1 definition using respSweepDefinition	AA D6 EA 17 06 81 86 AB 85 84 42 AB
V1 responds with sweep 2 definition using respSweepDefinition	AA D6 EA 17 06 82 87 5C 87 03 76 AB
V1 responds with sweep 3 definition using respSweepDefinition	AA D6 EA 17 06 83 8A C6 8A 8B 6F AB
V1 responds with sweep 4 definition using respSweepDefinition (not in use)	AA D6 EA 17 06 84 00 00 00 00 0B AB
V1 responds with sweep 5 definition using respSweepDefinition (not in use)	AA D6 EA 17 06 85 00 00 00 00 0C AB
Acc. writes custom sweep 0 using reqWriteSweep (commit = 0)	AA DA E6 15 06 80 85 3A 84 6C B4 AB
Acc. writes custom sweep 1 using reqWriteSweep (commit = 0)	AA DA E6 15 06 81 86 AB 85 84 40 AB
Acc. writes custom sweep 2 using reqWriteSweep (commit = 0)	AA DA E6 15 06 82 87 5C 87 03 74 AB
Acc. writes custom sweep 3 using reqWriteSweep (commit = 0)	AA DA E6 15 06 83 8A C6 8A 8B 6D AB
Acc. writes custom sweep 4 using reqWriteSweep (commit = 1)[1]	AA DA E6 15 06 C4 8C A0 8A DE DD AB
Acc. requests sweep definitions using reqAllSweepDefinitions	AA DA E6 16 01 81 AB
V1 responds with sweep 0 definition using respSweepDefinition	AA D6 EA 17 06 80 85 3A 84 6C B6 AB
V1 responds with sweep 1 definition using respSweepDefinition	AA D6 EA 17 06 81 86 AB 85 84 42 AB
V1 responds with sweep 2 definition using respSweepDefinition	AA D6 EA 17 06 82 87 5C 87 03 76 AB
V1 responds with sweep 3 definition using respSweepDefinition	AA D6 EA 17 06 83 8A C6 8A 8B 6F AB
V1 responds with sweep 4 definition using respSweepDefinition[1]	AA D6 EA 17 06 84 8C AD 8A D5 A3 AB
V1 responds with sweep 5 definition using respSweepDefinition (not in use)	AA D6 EA 17 06 85 00 00 00 00 0C AB

Notes 1: The new sweep that was requested has an upper edge of 36000 MHz and a lower edge of 35500 MHz. However, the new sweep that was returned has an upper edge of 36013 MHz and a lower edge of 35541 MHz.

reqWriteSweepDefinition

Use the *reqWriteSweepDefinition* packet to define a custom sweep. The custom sweep will be used instead of the factory sweeps when in Euro Mode. All used sweep definitions must be written every time any change is made or the changes will not take effect. The number of supported sweeps is Valentine One version specific and can be obtained using the reqMaxSweepIndex packet. The valid sweep sections can be read using the reqSweepSections packet. If a sweep definition is invalid, such as when a sweep definition crosses a sweep section boundary, the sweep will not be used. Refer to the Custom Sweep section for more details.

Valentine One version		V3.892 and above
Packet ID		$15
Payload Length		6
Payload bytes	0	Aux0 - Sweep Index (zero based) . *Sweep Index byte definition* 07 06 05 04 03 02 01 00 | | | | | | | | | | | | | | | \\-- Index bit 0 | | | | | | \\----- Index bit 1 | | | | | \\------- Index bit 2 | | | | \\---------- Index bit 3 | | | \\-------------- Index bit 4 | | \\---------------- Index bit 5 | \\-------------------- Commit Changes (see Custom Sweeps) \\---------------------- Reserved
	1	MSB of upper frequency sweep edge in MHz
	2	LSB of upper frequency sweep edge in MHz
	3	MSB of lower frequency sweep edge in MHz
	4	LSB of lower frequency sweep edge in MHz
	5	Checksum

Example frequency: If user wishes 34.106 GHz the user would send $85(MSB) $3A(LSB)

reqAllSweepDefinitions

Use the *reqAllSweepDefinitions* packet to request the custom sweep definitions. The Valentine One will respond to this packet with all of the available sweep definitions. Each sweep definition will be sent out in its own *respSweepDefinition* packet. The sweep definitions will be interleaved with the *infDisplayData* packets.

Valentine One version		V3.892 and above
Packet ID		$16
Payload Length		1
Payload Bytes	0	Checksum

respSweepDefinition

The *respSweepDefinition* packet defines a single custom sweep and is sent in response to a *reqAllSweepDefinitions* packet. These packets will be interleaved with the *infDisplayData* packets. One of these packets will be sent for each available sweep definitions. Unused sweeps have the upper and lower edges set to zero.

Valentine One version		V3.892 and above
Packet ID		$17
Payload Length		6
Payload bytes	0	Aux0 - Sweep Index. Refer to byte definition in the reqWriteSweepDefinition section.
	1	MSB of upper sweep edge in MHz
	2	LSB of upper sweep edge in MHz
	3	MSB of lower sweep edge in MHz
	4	LSB of lower sweep edge in MHz
	5	Checksum

reqDefaultSweeps

Use the *reqDefaultSweeps* packet to reset all custom sweep definitions back to their default. When operating in Euro Mode, this will change the display from 'C' to 'U' or 'c' to 'u' depending on the user's mode selection.

Valentine One version		V3.892 and above
Packet ID		$18
Payload Length		1
Payload Bytes	0	Checksum

reqMaxSweepIndex

Use the *reqMaxSweepIndex* packet to determine how many sweeps the current Valentine One version supports.

Valentine One version		V3.892 and above
Packet ID		$19
Payload Length		1
Payload Bytes	0	Checksum

respMaxSweepIndex

The *respMaxSweepIndex* packet is sent in response to the *reqMaxSweepIndex* packet. This packet will tell the user the maximum number of sweeps supported by the Valentine One. Because the sweep indices are zero-based, the value returned is the number of supported sweeps minus one.

Valentine One version		V3.892 and above
Packet ID		$20
Payload Length		2
Payload Bytes	0	The maximum sweep index for the current Valentine One version.
	1	Checksum

respSweepWriteResult

The *respSweepWriteResult* packet is sent in response to the *reqWriteSweepDefinition* packet with the *Commit* in the Aux0 byte set.

Valentine One version		V3.892 and above
Packet ID		$21
Payload Length		2
Payload Bytes	0	The sweep write result. 0 = Sweep Write Successful Any Other Value = The number of the first sweep with invalid parameters. The error number returned will be the sweep index + 1, where sweep index is the index from the *reqWriteSweepDefinition* packet.
	1	Checksum

reqSweepSections

The *reqSweepSections* packet is used to request the available custom sweep sections. The sweep definitions are returned using the *respSweepSections* packet.

Valentine One version		V3.892 and above
Packet ID		$22
Payload Length		1
Payload Bytes	0	Checksum

respSweepSections

One or more *respSweepSections* packets will be sent in response to the *reqSweepSections* request. This response packet will contain the definition for one, two or three custom sweep sections. The payload length will determine how many definitions are in the packet. If more than three custom sweep sections are available, multiple *respSweepSections* packets will be sent and they will be interleaved with *infDisplayData* packets. The upper nibble of the Sweep Definition Index byte will indicate how many custom sweep sections are available while the lower nibble indicates the overall index of the section definition.

Valentine One version		V3.892 and above		
Packet ID		$23		
Payload Length		6 if one section definition is in the Packet Data (PD) 11 if two section definitions are in the Packet Data (PD) 16 if three section definitions are in the Packet Data (PD)		
# Of section in PD		1	2	3
	0	Sweep Definition Index	Sweep Definition Index	Sweep Definition Index
	1	MSB of upper edge[1]	MSB of upper edge[1]	MSB of upper edge[1]
	2	LSB of upper edge[1]	LSB of upper edge[1]	LSB of upper edge[1]
	3	MSB of lower edge[1]	MSB of lower edge[1]	MSB of lower edge[1]
	4	LSB of lower edge[1]	LSB of lower edge[1]	LSB of lower edge[1]
	5	Checksum	Sweep Definition Index	Sweep Definition Index
	6		MSB of upper edge[1]	MSB of upper edge[1]
	7		LSB of upper edge[1]	LSB of upper edge[1]
	8		MSB of lower edge[1]	MSB of lower edge[1]
	9		LSB of lower edge[1]	LSB of lower edge[1]
	10		Checksum	Sweep Definition Index
	11			MSB of upper edge[1]
	12			LSB of upper edge[1]
	13			MSB of lower edge[1]
	14			LSB of lower edge[1]
	15			Checksum

Note 1: All sweep section edges are returned as the frequency in MHz.

```
Sweep Definition Index Byte
07 06 05 04 03 02 01 00
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  \-- Count B0 (The number of sections available)
|  |  |  |  |  |  \----- Count B1
|  |  |  |  |  \-------- Count B2
|  |  |  |  \----------- Count B3
|  |  |  \-------------- Index B0 (The index of this sweep section)
|  |  \----------------- Index B1
|  \-------------------- Index B2
\----------------------- Index B3
```

8 Display and Audio Packets

Reproducing the Valentine One Display

The *infDisplayData* packet provides information about the Bogey Count seven segment display, the signal strength bar graph, the band indicators and the arrows. This data can be used to reproduce the Valentine One display on another device. One important feature of reproducing the Valentine One display is the ability to blink the band, arrow and Bogey Count indicators. This is accomplished using the different image bytes in the *infDisplayData* packet. Two image bytes are provided for the Bogey Count seven segment as well as the band and arrow indicators. When there is no blinking, Image 1 and Image 2 bytes are the same. When there is a blinking indicator, Image 2 will have the corresponding 'on' bits turned 'off'. To produce a blinking effect, a display device simply needs to toggle between Image 1 and Image 2 at the desired blink rate. Valentine Research recommends a blink rate of 10.416 Hz, which requires toggling the display every 96 mS.

For example, assume the Valentine One has detected a single X Band signal and a single Ka Band signal to the front and that the Ka Band signal is the priority alert. In this scenario, the Ka Band indicator would be blinking on the Valentine One display. The *infDisplayData Band and Arrow Indicator Image 1* byte would be $2A to indicate that the front arrow, X Band and Ka Band indicators are on with all other band and arrow indicators off. However, the *infDisplayData Band and Arrow Indicator Image 2* byte would be $28 to indicate that the front arrow and X Band indicators are on with all other band and arrow indicators off. Therefore the Ka indicator can be blinked by toggling between *Band and Arrow Indicator Image 1* and *Band and Arrow Indicator Image 2*. The same techniques should be used for blinking the Bogey Count seven segment display. The signal strength bar graph does not blink so only one image byte is provided.

infDisplayData

The *infDisplayData* packet will provide accessories with all of the information needed to rebuild the front panel display. Status bytes are also provided to inform the accessories about the current operating condition of the Valentine One.

<table>
<tr><td colspan="2">Valentine One version</td><td>V3.892 and above</td></tr>
<tr><td colspan="2">Packet ID</td><td>$31</td></tr>
<tr><td colspan="2">Payload Length</td><td>9</td></tr>
<tr><td rowspan="9">Payload Bytes</td><td>0</td><td>Bogey Counter 7 Segment Image 1
07 06 05 04 03 02 01 00
| | | | | | | |
| | | | | | | \-- Seg a
| | | | | | \----- Seg b
| | | | | \-------- Seg c
| | | | \----------- Seg d
| | | \-------------- Seg e
| | \----------------- Seg f
| \-------------------- Seg g
\---------------------- dp</td></tr>
<tr><td>1</td><td>Bogey Counter 7 Segment Image 2 – uses the same byte definition as byte 0</td></tr>
<tr><td>2</td><td>Signal Strength Bar Graph Image
07 06 05 04 03 02 01 00
| | | | | | | |
| | | | | | | \- b0(left)
| | | | | | \---- b1
| | | | | \------- b2
| | | | \---------- b3
| | | \------------- b4
| | \---------------- b5
| \-------------------- b6
\---------------------- b7(right)</td></tr>
<tr><td>3</td><td>Band and Arrow Indicator Image 1
07 06 05 04 03 02 01 00
| | | | | | | |
| | | | | | | \-- LASER
| | | | | | \----- Ka BAND
| | | | | \-------- K BAND
| | | | \----------- X BAND
| | | \-------------- Reserved
| | \----------------- FRONT
| \-------------------- SIDE
\---------------------- REAR</td></tr>
<tr><td>4</td><td>Band and Arrow Indicator Image 2 – uses the same byte definition as byte 3</td></tr>
<tr><td>5</td><td>Aux0 – Refer to Table 8.1for more information
07 06 05 04 03 02 01 00
| | | | | | | |
| | | | | | | \- Soft
| | | | | | \---- TS Holdoff
| | | | | \------- Sys. Status
| | | | \---------- Display On
| | | \------------- Euro Mode
| | \---------------- Custom Sweep
| \-------------------- Reserved
\---------------------- Reserved</td></tr>
<tr><td>6</td><td>Aux1 – Reserved for future use</td></tr>
<tr><td>7</td><td>Aux2 – Reserved for future use</td></tr>
<tr><td>8</td><td>Checksum</td></tr>
</table>

Table 8.1 – Bit descriptions for *infDisplayData* packet's Aux0 byte

Bit #	Bit Name	Bit Description
0	Soft	This bit describes the current mute status: 0 - indicates the audio is not muted. 1 - indicates the audio is muted.
1	TS Holdoff	This bit tells the accessories if a time slice is allowed. A 0 indicates that all accessories are allowed to have a time slice following this packet. A 1 indicates that none of the accessories are allowed to have a time slice following this packet.
2	System Status	This bit describes the Valentine One status. A 0 indicates that the Valentine One is not actively searching for alerts. A 1 indicates that the Valentine One has successfully signed on and is actively searching for alerts.
3	Display On	This bit describes the status of the Valentine One display. A 0 indicates the Valentine One has turned off the main display. A 1 indicates the main display is turned on.
4	Euro Mode	This bit describes the Valentine One European Mode status. A 0 indicates the Valentine One is not operating in Euro Mode. A 1 indicates the Valentine One is operating in Euro Mode.
5	Custom Sweep	This bit describes the custom sweep status. A 0 indicates that custom sweeps have not been defined. A 1 indicates that custom sweeps have been defined and custom modes will be used if operating in Euro Mode.
6	Reserved	This bit has been reserved for future use
7	Reserved	This bit has been reserved for future use

reqTurnOffMainDisplay

The *reqTurnOffMainDisplay* packet is used to force the Valentine One to blank the main display. The results of this request can be verified using the Aux0 byte in the *infDisplayData* packet.

Valentine One version		V3.892 and above
Packet ID		$32
Payload Length		1
Payload Bytes	0	Checksum

reqTurnOnMainDisplay

The *reqTurnOnMainDisplay* packet is used to force the Valentine One to turn on the main display. The results of this request can be verified using the Aux0 byte in the *infDisplayData* packet.

Valentine One version		V3.892 and above
Packet ID		$33
Payload Length		1
Payload Bytes	0	Checksum

Important information about changing the display state: By default, all Valentine Ones turn off the main display when a Concealed Display is connected and turn it back on when the Concealed Display is disconnected. The *reqTurnOffMainDisplay* and *reqTurnOnMainDisplay* packets change the current display state, but do not disable this feature. In other words, these packets are applied at the time they are received and then forgotten by the Valentine One. For example, if a Concealed Display is plugged in and the Valentine One receives a *reqTurnOnMainDisplay*, the display will be turned on. If the Concealed Display is disconnected and then reconnected, the Valentine One will turn the display off when it senses that the Concealed Display is back on the network. Conversely, if there is no Concealed Display connected and the Valentine One receives a *reqTurnOffMainDisplay* packet, the main display will be turned off. However, if a Concealed Display is connected and then disconnected, the main display will be turned back on when the Valentine One senses that the Concealed Display has been disconnected.

reqMuteOn

The *reqMuteOn* packet is used by an accessory to mute all alerts in the Valentine One. The results of this request can be verified using the Aux0 byte in the *infDisplayData* packet. The Valentine One treats this packet as a mute button press. Therefore, this command is only in effect until all alerts are no longer being tracked by the Valentine One.

Valentine One version		V3.892 and above
Packet ID		$34
Payload Length		1
Payload Bytes	0	Checksum

reqMuteOff

The *reqMuteOff* packet is used by an accessory to unmute all alerts in the Valentine One that were not muted by the Valentine One's internal logic. For example, alerts muted by a mute button press, the Savvy or a *reqMuteOn* packet can be unmuted using the *reqMuteOff* packet. The results of this request can be verified using the Aux0 byte in the *infDisplayData* packet.

Valentine One version		V3.892 and above
Packet ID		$35
Payload Length		1
Payload Bytes	0	Checksum

reqChangeMode

The *reqChangeMode* packet is used by an accessory to change the current mode on the Valentine One. The results of this request can be verified by decoding the *infDisplayData* packet.

Valentine One version		V3.892 and above		
Packet ID		$36		
Payload Length		2		
Payload Bytes	0	New Mode	US Operation (Default)	European Operation
		1	All Bogeys Mode	'U' Mode – K and Ka(Photo) or 'C' Mode – K and Custom Sweeps
		2	Logic Mode	'u' Mode – Ka band only or 'c' Mode – Custom Sweeps
		3	Advanced Logic Mode	Invalid
		Any others	Invalid	Invalid
	1	Checksum		

9 Alert Output Packets

All ESP Valentine Ones allow the user to read detailed information for all alerts being displayed on the Valentine One front panel. The list of alerts being displayed is called the Alert Table. The Alert Table includes the following information:

1. Number of alerts present
2. Alert frequency in MHz
3. Signal strength for front and rear
4. Band (X,K,Ka,Ku)
5. Direction calculated by the Valentine One logic
6. Indication of the highest threat (priority alert). On the Valentine One front panel, the priority alert is indicated by a flashing band indicator and, under some circumstances, a flashing arrow when more than one alert is present.

The Alert Table is sent out in a stream that is interleaved with the *infDisplayData* and other responses from the Valentine One. The stream is started by sending *reqStartAlertData* packet and continues until the Valentine One receives a *reqStopAlertData* packet or is powered off. Each alert in the Alert Table is delivered in a *respAlertData* packet, which indicates how many alerts are in the table and the alert's index in the table. When no alerts are present, an alert index of zero and a count of zero will be sent periodically. The maximum number of alerts supported is fifteen. Due to the interleaved nature of the data, when the table is full there may be up to 1.2 seconds between the receipt of the first alert in the table and the last alert in the table. For devices that wish to display alert data, it is recommended that the display is only updated after a complete Alert Table is received.

The Alert Table contains radar signals only. Laser alerts are not represented in the Alert Table.

The signal strength is reported as a unit-less number that can be used when comparing the relative strength of various signals. The reported strength of each signal can be mapped to a front panel Signal Strength Bar Graph using Table 40.1

Table 9.1 – Alert Table to Bar Graph Map

Number of LEDs	Equivalent *infDisplayData* Bar Graph Value	X Band	K/Ku Band	Ka Band
8	$FF	$D0 - $FF	$C2 - $FF	$BA - $FF
7	$7F	$C5 - $CF	$B8 - $C1	$B3 - $B9
6	$3F	$BD - $C4	$AE - $B7	$AC - $B2
5	$1F	$B4 - $BC	$A4 - $AD	$A5 - $AB
4	$0F	$AA - $B3	$9A - $A3	$9E - $A4
3	$07	$A0 - $A9	$90 - $99	$97 - $9D
2	$03	$96 - $9F	$88 - $8F	$90 - $96
1	$01	$01 - $95	$01 - $87	$01 - $8F
0	$00	$00	$00	$00

reqStartAlertData

The *reqStartAlertData* packet is used to request information on all alerts being displayed on the Valentine One front panel.

Valentine One version		V3.892 and above
Packet ID		$41
Payload Length		1
Payload Bytes	0	Checksum

reqStopAlertData

The *reqStopAlertData* packet is used to stop the Valentine One from sending *respAlertData* packets that were started using the *reqStartAlertData* packet

Valentine One version		V3.892 and above
Packet ID		$42
Payload Length		1
Payload Bytes	0	Checksum

respAlertData

The Valentine One will send the *respAlertData* packet in response to the *reqStartAlertData* packet. The data will be sent interleaved with the *infDisplayData* packets.

Valentine One version		V3.892 and above
Packet ID		$43
Payload Length		7
Payload Bytes	0	Alert Index and Count 07 06 05 04 03 02 01 00 | | | | | | | | | | | | | | | \\-- Count B0 (Number of alerts present) | | | | | | \\----- Count B1 | | | | | \\-------- Count B2 | | | | \\----------- Count B3 | | | \\-------------- Index B0 (Index of this alert) | | \\----------------- Index B1 | \\-------------------- Index B2 \\----------------------- Index B3
	1	Frequency MSB in MHz
	2	Frequency LSB in MHz
	3	Front Signal Strength
	4	Rear Signal Strength
	5	Band/Arrow definition 07 06 05 04 03 02 01 00 | | | | | | | | | | | | | | | \\-- LASER | | | | | | \\----- Ka BAND | | | | | \\-------- K BAND | | | | \\----------- X BAND | | | \\-------------- Ku Band | | \\----------------- FRONT | \\-------------------- SIDE \\----------------------- REAR
	6	Aux0 07 06 05 04 03 02 01 00 | | | | | | | | | | | | | | | \\-- Reserved | | | | | | \\----- Reserved | | | | | \\-------- Reserved | | | | \\----------- Reserved | | | \\-------------- Reserved | | \\----------------- Reserved | \\-------------------- Reserved \\----------------------- Priority Alert[1]
	7	Checksum

Notes: 1: A 1 in the priority bit indicates that the alert represented by the packet is the highest priority threat in the Alert Table.

Table 9.2 - *respAlertData* example with zero alerts present

ESP Packet	Index	Frequency (MHz)	Front SS	Rear SS	Band	Dir.	Priority
AA D6 EA 43 07 00 00 00 00 00 00 00 B4 AB	1	00000	$00	$00	n/a	n/a	No

Table 9.3 - *respAlertData* example with three alerts present

ESP Packet	Index	Frequency (MHz)	Front SS	Rear SS	Band	Dir.	Priority
AA D6 EA 43 07 13 29 1D 21 85 88 00 E8 AB	1	10525	$21	$85	X	Rear	No
AA D6 EA 43 07 23 5E 56 92 83 24 00 00 AB	2	24150	$92	$83	K	Front	No
AA D6 EA 43 07 33 87 8C B6 81 22 80 30 AB	3	34700	$B6	$81	Ka	Front	Yes

10 Miscellaneous Packets

respDataReceived

The *respDataReceived* packet is used to acknowledge a data transmission and does not need to be used unless the data received specifically requires an acknowledgement. The Concealed Display will use this packet to respond to the *infDisplayData* packet to inform the Valentine One that a Concealed Display is present and that the main display should be turned off.

Valentine One version		V3.892 and above
Packet ID		$61
Payload Length		1
Payload Bytes	0	Checksum

reqBatteryVoltage

The *reqBatteryVoltage* packet is used to request the battery voltage at the connection to the Valentine One.

Valentine One version		V3.892 and above
Packet ID		$62
Payload Length		1
Payload Bytes	0	Checksum

respBatteryVoltage

The *respBatteryVoltage* packet is sent in response to a *reqBatteryVoltage* request. The battery voltage is measured at the Valentine One connector.

Valentine One version		V3.892 and above
Packet ID		$63
Payload Length		3
Payload Bytes	0	Integer portion of the battery voltage
	1	Decimal portion of the battery voltage
	2	Checksum

Example packet for 13.1V : AA D6 EA 63 03 0D 01 DE AB

respUnsupportedPacket

The *respUnsupportedPacket* is used when an unsupported packet identifier is received and the Destination ID is not a General Broadcast.

Valentine One version		V3.892 and above
Packet ID		$64
Payload Length		2
Payload Bytes	0	The identifier of the unsupported packet.
	1	Checksum

respRequestNotProcessed

The *respRequestNotProcessed* packet will be sent out when a device is too busy to process an incoming request. This packet should be sent as soon as possible, i.e. before the next *infDisplayData* packet for the Valentine One and in the next time slice for accessories. The Valentine One will issue this packet for each request that is dropped due to a priority conflict.

Valentine One version		V3.892 and above
Packet ID		$65
Payload Length		2
Payload Bytes	0	The identifier of the unprocessed packet.
	1	Checksum

infV1Busy

The *infV1Busy* packet will be sent out by the Valentine One when there are pending request that will not be processed immediately. The payload will include the packet identifier of all pending requests to the Valentine One. The *infV1Busy* packet will always have a Destination ID of General Broadcast.

Valentine One version	V3.892 and above					
Packet ID	$66					
Payload Length	2 if there is one pending Packet Identifier (PI) in the queue 3 if there are two pending Packet Identifiers (PI) in the queue 4 if there are three pending Packet Identifiers (PI) in the queue 5 if there are four pending Packet Identifiers (PI) in the queue 6 if there are five pending Packet Identifiers (PI) in the queue					
# of Request In Queue		1	2	3	4	5
	0	Pending PI #1	Pending PI #1	Pending PI #1	Pending PI #1	Pending PI #1
	1	Checksum	Pending PI #2	Pending PI #2	Pending PI #2	Pending PI #2
	2		Checksum	Pending PI #3	Pending PI #3	Pending PI #3
	3			Checksum	Pending PI #4	Pending PI #4
	4				Checksum	Pending PI #5
	5					Checksum

respDataError

The *respDataError* packet will be sent if a device receives a packet with invalid data. The data may be invalid because it is not formatted correctly, e.g. the wrong number of bytes, or the device is in a state in which the data request is invalid. For example, the *respDataError* response will be used by the Valentine One if it receives an invalid mode in the *reqChangeMode* packet or if it receives a request to change to Advanced Logic mode while operating in Euro mode. This command may not be sent in response to invalid data that is sent out as a General Broadcast.

Valentine One version		V3.892 and above
Packet ID		$67
Payload Length		2
Payload Bytes	0	The packet ID of the message with the invalid data.
	1	Checksum

11 Savvy™ Specific Packets

The Savvy has two features that can be configured through the ESP interface. The first configurable feature is the mute threshold speed. By default, the Savvy uses the thumbwheel on the device to set the threshold. However, the user can override the thumbwheel and set the threshold to any speed between 1-158 MPH (1–254 KPH) using the *reqOverrideThumbwheel* packet. The second configurable feature is the ability for the Savvy to unmute signals when the vehicle speed rises above the mute threshold speed. The unmute feature is enabled by default, but can be disabled using the *reqSetSavvyUnmuteEnable* packet. Unplugging the Savvy from the OBD-II port will reset all configurable settings back to the factory default.

reqSavvyStatus

The *reqSavvyStatus* packet is used to request the current status from the Savvy.

Valentine One version		V3.892 and above
Packet ID		$71
Payload Length		1
Payload Bytes	0	Checksum

respSavvyStatus

The *respSavvyStatus* packet is sent in response to a *reqSavvyStatus* packet and contains the current Savvy status.

Valentine One version		V3.892 and above
Packet ID		$72
Payload Length		3
Payload Bytes	0	The current speed threshold in KPH.
	1	Savvy Status 07 06 05 04 03 02 01 00 | | | | | | | | | | | | | | | \\- 1 = Threshold Overridden by user | | | | | | \\---- 1 = Unmute Enabled | | | | | \\------- Reserved | | | | \\---------- Reserved | | | \\------------- Reserved | | \\---------------- Reserved | \\------------------ Reserved \\-------------------- Reserved
	2	Checksum

reqVehicleSpeed

The *reqVehicleSpeed* packet is used to request the current vehicle speed measured by the Savvy.

Valentine One version		V3.892 and above
Packet ID		$73
Payload Length		1
Payload Bytes	0	Checksum

respVehicleSpeed

The *respVehicleSpeed* packet is sent by the Savvy in response to a reqVehicleSpeed packet.

Valentine One version		V3.892 and above
Packet ID		$74
Payload Length		2
Payload Bytes	0	The vehicle speed, in KPH, as read by the Savvy
	1	Checksum

reqOverrideThumbwheel

The *reqOverrideThumbwheel* packet is used to override the Savvy mute threshold speed. The new speed setting will be used until the Savvy is unplugged from the vehicle or the thumbwheel is changed.

Valentine One version		V3.892 and above
Packet ID		$75
Payload Length		2

Payload Bytes	0	The new speed, in KPH, to be used as the mute threshold speed. The speed values are defined as
		`0x00 : "None" – Do not mute at any speed`
		`0x01 – 0xFE : The new speed in KPH`
		`0xFF : "Auto" – Mute at all speeds`

	1	Checksum

Example:

To set the Savvy mute threshold speed to 45 MPH

Convert MPH to KPH

45 MPH = 45 * 1.6093 = 72.41 KPH

Send the *reqOverrideThumbwheel* packet

AA D2 E6 75 02 48 21 AB

reqSetSavvyUnmuteEnable

The *reqSetSavvyUnmuteEnable* packet is used to enable or disable the unmute functionality in the Savvy. Use the *reqSavvyStatus* to verify the status has been set correctly.

Valentine One version		V3.892 and above
Packet ID		$76
Payload Length		2
Payload Bytes	0	The enable status for the Savvy unmute functionality. 0 = Disable unmuting 1 = Enable unmuting
	1	Checksum

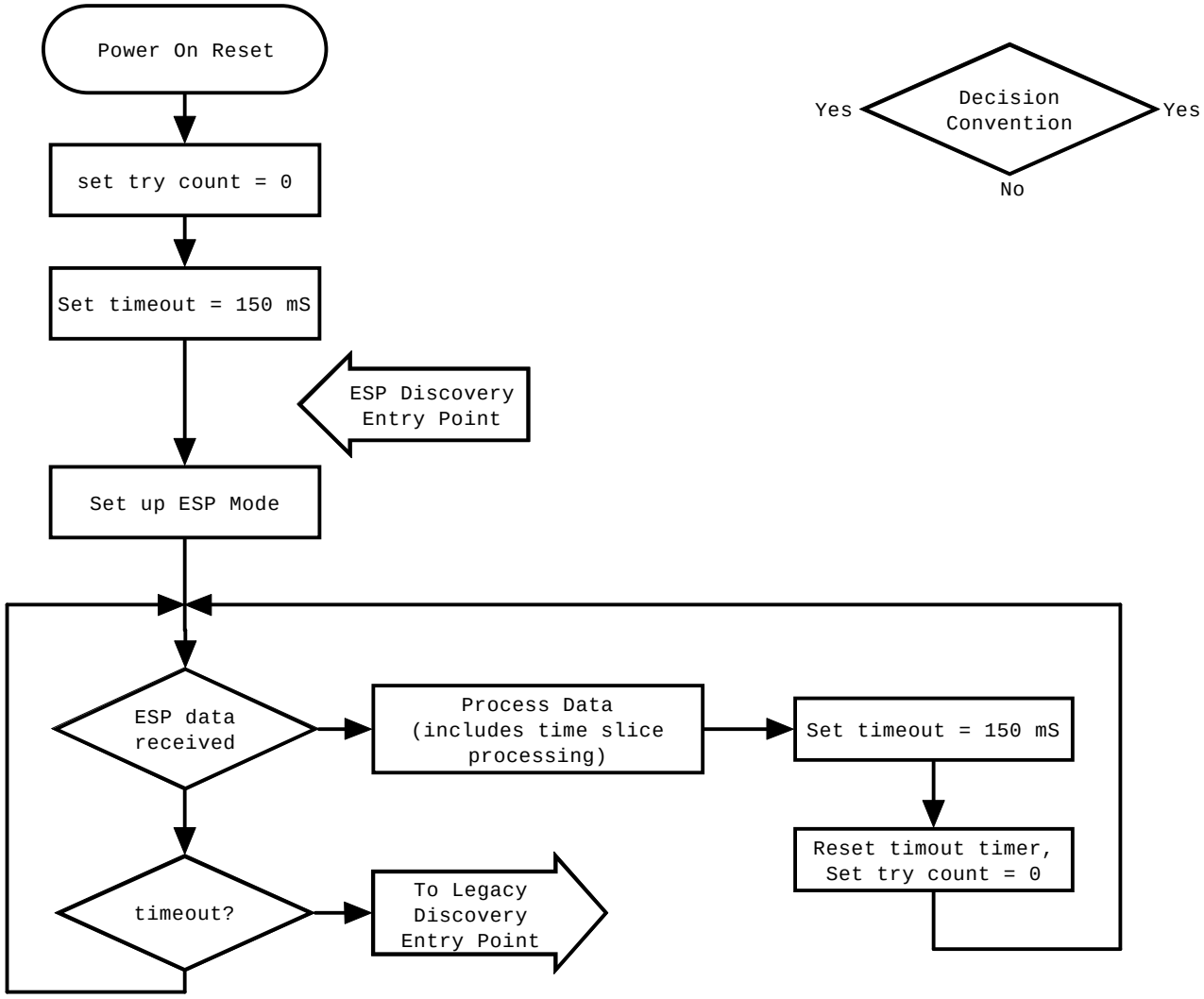
12 Appendix

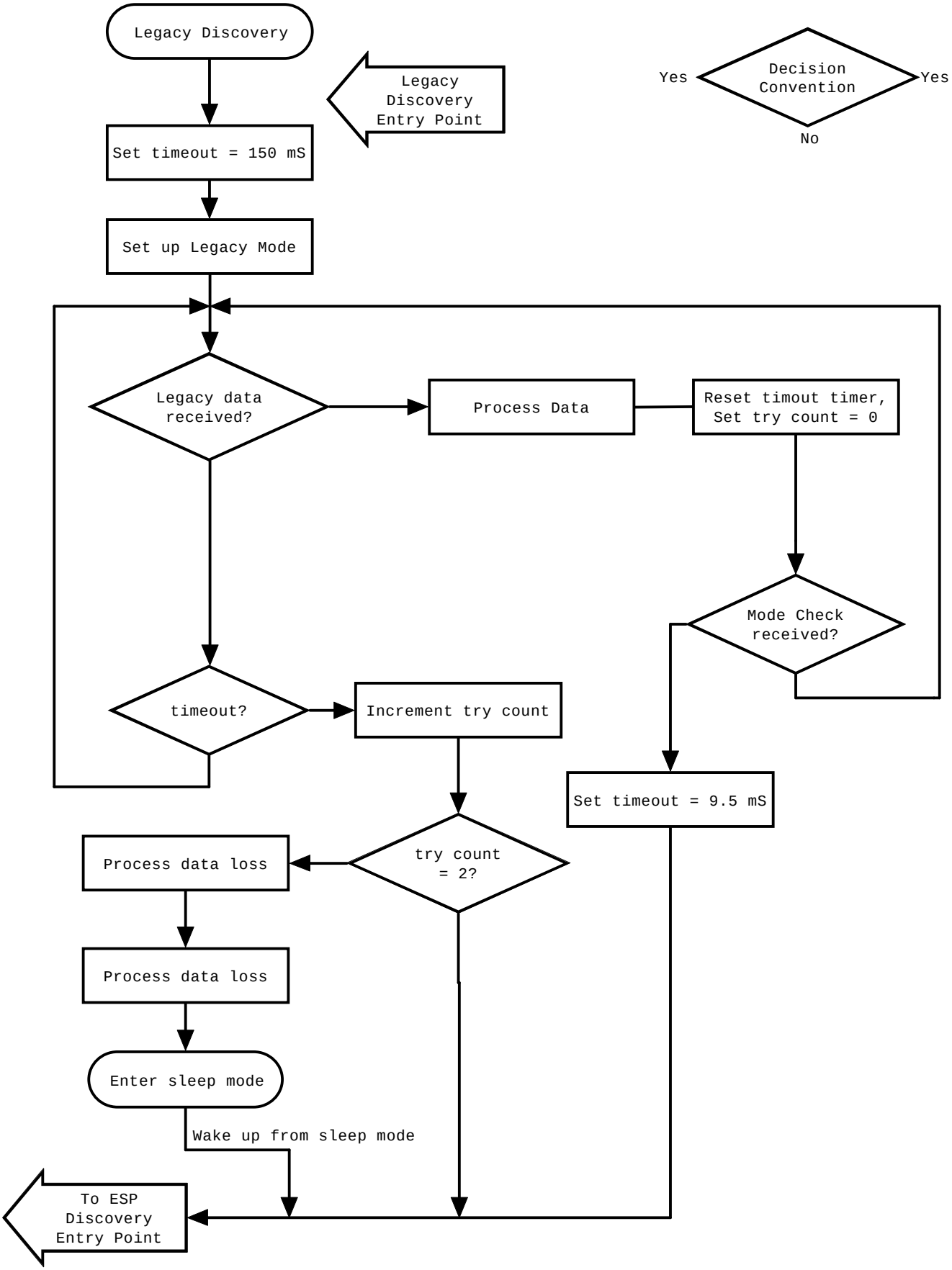
Appendix 12.1 ESP Specification User Feature versus User Byte-bit Summary

See http://www.valentine1.com\Lab\techreport3.asp for full feature description and software version sensitivity.

Feature	Description	Byte	bit	Factory
1	X band On/Off	User 0	0	On
2	K band On/Off	User 0	1	On
3	Ka band On/Off	User 0	2	On
4	Laser On/Off	User 0	3	On
5	Bargraph Normal/Responsive	User 0	4	Normal
6	Ka False Guard On/Off	User 0	5	On
7	Feature b-G(K Muting) On/Off	User 0	6	Off
8	Mute Volume Lever/Zero	User 0	7	Lever
A	Post-mute Bogey Lock Volume Lever/Knob	User 1	0	Knob
b	K Mute Timer	User 1	1	10 sec
C	"	User 1	2	
d	"	User 1	3	
E	K Initial Unmute 4 lights	User 1	4	On
F	K Persistent Unmute 6 lights	User 1	5	On
G	K Rear Mute On/Off	User 1	6	Off
H	Ku band On/Off	User 1	7	Off
I	(Factory Test)	User 2	0	Off
J	Pop On/Off	User 2	1	On
u	Euro On/Off	User 2	2	Off
u bar	Euro X band On/Off	User 2	3	Off
t	Filter On/Off	User 2	4	Off
L	Force Legacy CD	User 2	5	Off
(undef)		User 2	6	
(undef)		User 2	7	
(unused)		User 3		
(unused)		User 4		
(unused)		User 5		

Appendix 12.2 Accessory Discovery Flowchart




Legacy Discovery
Legacy Discovery Entry Point
Yes
Decision Convention
Yes
No
Set timeout = 150 mS
Set up Legacy Mode
Legacy data received?
Process Data
Reset timout timer, Set try count = 0
Mode Check received?
timeout?
Increment try count
Set timeout = 9.5 mS
try count = 2?
Process data loss
Process data loss
Enter sleep mode
Wake up from sleep mode
To ESP Discovery Entry Point

13 Glossary

Baud Rate - the rate at which serial communications takes place defined in Bits-per-second.

Byte Pacing (t_{pacing}) - a slight delay inserted after each byte within an ESP packet, including EOF, that allows the destination device time to process each data byte received.

CDOS (Concealed Display Output Stream) - the traditional communication protocol that provides one-way data flow from the Valentine One Radar Locator to provide display information to the traditional *Concealed Display* device attached to the *Concealed Display* output connection.

Checksum - an arithmetic test to determine the validity of a received *ESP Packet* that is included in Valentine One Radar Locators with a *Device ID* of $A (decimal 10).

Concealed Display - a proprietary device from Valentine Research, Inc that attaches to the Valentine One Radar Locator through the *Concealed Display output* pin on the RJ-11 power connector and allows remote indication of the Valentine One Radar Locator's visual portion of detected alerts.

Custom Sweeps - a group of user defined bands of frequencies that define an area of interest for the Valentine One Radar Locator to repetitively investigate during its normal operation.

DI (Destination Identifier) - the *Device ID* of the target of the *ESP Packet*.

Discovery - the method by which ESP devices discover the current operational mode (*Legacy mode or ESP mode)* of connected devices attached to the *ESP bus*.

Engineering Control Number - the fourth digit after the decimal point in the version number sent by all Valentine One devices. This digit is for manufacturing use by Valentine Research. This digit is not used to indicate feature or performance changes. Therefore, this digit is not shown when the Valentine One is displaying the software version.

EOF (End-of-Frame) - a fixed value position in an *ESP Packet* (value of $AB) that indicates the end of an *ESP Packet* frame.

ESP (Extended Serial Protocol) - a Valentine Research, Inc. proprietary serial communications protocol which is defined for the purpose of data exchange between devices attached to the Valentine One Radar Locator by means of the *Concealed Display* output connection.

ESP bus - the RJ-11 power connector pin that was traditionally used by the Valentine One Radar Locator to transmit display information to remote display accessories (i.e. *Concealed Display*) and is now used by the *ESP Protocol* to bi-directionally communicate to other *ESP* enabled devices.

GHz – Gigahertz or 10^9 Hertz

General Broadcast – a *Device ID* (value of $08) that can be utilized to send general information (non-*Destination ID*) to all devices attached to the *ESP bus*.

Guard Time (t_{guard}) - a period of time inserted around a *Time Slice* which insures proper synchronization between devices with slightly varying time bases.

Half-Duplex - a bidirectional communication system that allows communication in only one direction at a time.

K Band - a portion of the microwave spectrum (24.050 to 24.250 GHz)

Ka Band - a portion of the microwave spectrum (33.400 to 36.000 GHz)

KPH – Kilometers per Hour (MPH x 1.6093)

Ku Band - a portion of the microwave spectrum used in Europe exclusively (13.400 to 13.500 GHz)

PI (Packet Identifier) - the number associated with a particular packet that identifies its function (*see ESP Quick Reference* for a list of *Packet ID* versus *Packet Name*)

Legacy Mode - the operating mode where the Valentine One Radar Locator is the sole transmitter of data to peripheral devices attached to the *Concealed Display* output connection. Peripheral devices are detected by means of analog voltages that represent each peripheral device.

Laser - police speed measuring device (950nm wavelength) used for speed enforcement.

LSB - the Least Significant Byte in a 16 bit value (i.e. for '$3419' '19' would be the LSB)

MSB - the Most Significant Byte in a 16 bit value (i.e. for '$3419' '34' would be the MSB)

MHz – Megahertz or 10^6 Hertz

OI (Originator Identifier) - the *Device ID* of the source of the *ESP Packet*.

Payload - the portion of an *ESP Packet* that contains informational data to be exchanged between *ESP devices.*

PL (Payload Length) - the length of the portion of an *ESP Packet* that contains informational data to be exchanged between *ESP devices.*

Remote Audio - a proprietary device from Valentine Research, Inc that attaches to the Valentine One Radar Locator through the *Remote Audio output* pin on the RJ-11 power connector and allows separate amplification of the audio portion of the Valentine One Radar Locator's detected alerts.

Savvy - a proprietary device from Valentine Research, Inc with the ability to mute the Valentine One Radar Locator below a user adjustable vehicle speed. This device attaches to the ODB-II port of the automobile for the purpose of extracting vehicle speed.

SOF (Start-of-Frame) - a fixed value position in an *ESP Packet* (value of $AA) that indicates the start of an *ESP Packet* frame.

Sweep Section – A frequency range that specifies then allowable custom sweep frequencies. Custom sweeps must be entirely within a sweep section.

Time Slicing - a scheme which allows unique allotted time periods, after an initial synchronization marker transmitted by the Valentine One Radar Locator, for attached ESP devices to transmit to other attached ESP devices. Each attached device has a unique time slice assignment (see *Time Slice Assignment* table).

UART – Universal Asynchronous Receiver Transmitter

X Band - a portion of the microwave spectrum (10.500 to 10.550 GHz)

14 Revision History

Specification Revision History

File Revision	Date	Change Description
3.0000	01/11/12	Document Released

Valentine One Version Information

Software Version	ESP Information
3.892	Initial ESP release
3.893	Changed Valentine One Identifier from ‘9’ to ‘A’ and added checksums to all packets.