SmartSensor Advance Data Protocol V1.3

June 6, 2007 ©Wavetronix LLC
Revision 4.00

Wavetronix Contact Information

Address:	380 S. Technology Ct. Lindon, Utah 84042 USA
Voice:	(801) 764-0277
Fax:	(801) 764-0208
Web:	www.smartsensor.us
E-mail:	sales@smartsensor.us support@smartsensor.us

Copyright / Trademarks

Table of Contents

Communicating with SmartSensor Advance

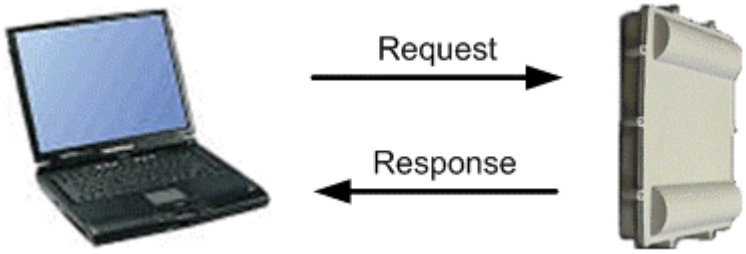
SmartSensor Advance is a leading edge traffic monitoring radar providing passage, ETA, range, and speed detection measurements in a consistently reliable and accurate manner.

To make installation and integration of SmartSensor Advance into a traffic monitoring system simpler and more reliable the SmartSensor has two built in serial ports, RS-232 and RS-485. These serial ports can be configured to operate from 9600 to 115200 bps.

In most traffic control applications, the SmartSensor pushes actuation data directly to a Click! contact closure device or another interface to a traffic control unit. Data can be pushed on either the RS-485 port or RS-232 port.

In some traffic data monitoring applications, the system polls the SmartSensor Advance unit, which responds with the data requested as shown in Figure 1. For example, a command is sent from a PC over a serial connection, and SmartSensor Advance then responds to that command.

Figure 1: Request-Response Polling Communication



The protocols used to communicate with the SmartSensor are called the Simple SmartSensor Protocol, and Multi-drop SmartSensor Protocol.

Simple SmartSensor Protocol Specification

The Simple SmartSensor Protocol allows an application to communicate with the SmartSensor over a dedicated link. With SmartSensor Advance, this protocol allows for the transfer of actuation data for up to eight group alerts. Each group alert can signal the controller when (speed and ETA) conditions are met within up to eight zones. Consult the SmartSensor installation manual to understand more about the configuration of group alerts and zones.

The protocol defines a set of messages using ASCII characters. The ASCII code assigns letters, numbers, punctuation marks, and other common characters to the decimal numbers 0 to 127 (See Appendix A). For example, the character ‘0’ is assigned

the decimal number 48. Each message of the protocol consists of a string of ASCII characters. For example, the 3-character string "X1\r" is sent by an application to request the current actuation data. (The "\r" character is used throughout this document and by some programming languages to represent an ASCII "carriage return". In the ASCII code, a "carriage return" is assigned the decimal number 13.)

When using a polling communication paradigm, the messages can be divided into two groups: requests and responses. In the Simple SmartSensor Protocol, each request-response pair shares the same header substring. For example, both the actuation data request and response begin with the substring "X1". The header is at the beginning of each message.

The header is followed by the payload and footer as diagrammed in Figure 2 for the response message: "X1000A~\r\r".

Figure 2: Format of Example Message

	Header	Payload	Footer	
			Checksum	Terminator
Example	"X1"	"000A"		"~\r\r"

The payload is the data portion of the message. In subsequent sections of this document, the format of the payload substring is defined for each request and response message.

The footer is the used to validate and terminate the message. Validation is performed using a checksum on critical information within a message. Some messages (like the "X1" response in Figure 2) have no checksum at all. However, when used, the checksum is a 4-character hexadecimal string formulated by adding the numerical ASCII codes of all the characters in a critical substring of the message. As an example, if a checksum was calculated on the payload substring in Figure 2 then the checksum would be determined in the following manner (See Appendix A for ASCII codes):

```
                '0'  '0'  '0'  'A'
Checksum  = 48+48+48+65
          = 209        (Decimal)
          = D1         (Hexadecimal)
          = "00D1"     (4-character hexadecimal string)
```

Multi-drop SmartSensor Protocol Specification

The Multi-drop SmartSensor Protocol allows an application to communicate with multiple SmartSensors on a shared bus. The Multi-drop SmartSensor Protocol is an

extension of the Simple SmartSensor Protocol. The multi-drop protocol prepends a multi-drop prefix to the header of each message. For example, “Z00001X1” is the header for a multi-drop message that retrieves the next actuation data from a SmartSensor with a multi-drop ID of “0001”. Each sensor on a shared bus should be assigned a different multi-drop ID. The format of the prefix is outlined in Figure 3.

Figure 3: Multi-drop Prefix

	Version ID	Multi-drop ID
Length	2 characters	4 characters
Example	“Z0”	“0001”

The 2-character string “Z0” identifies that this message conforms to version 1.0 of the Multi-drop protocol. The 4-character Multi-drop ID is the address selected during configuration of the sensor using SmartSensor Manager. The default Multi-drop ID is the last 4 characters of the SmartSensor’s Serial Number. The characters of the Multi-drop ID must be numeric characters ‘0’ –‘9’.

Getting Actuation Data

To query the sensor whether vehicles monitored by the sensor meet specified conditions, transmit the message request “X1\r”. The response from the SmartSensor contains a four-character payload substring that contains a hexadecimal integer. The lower 8 bits of the binary representation of this integer indicate the presence of one or more vehicles meeting the conditions of the 8 alerts. (Alert conditions can specify range, speed, ETA, and logical Boolean requirements.) The least significant bit corresponds to alert 1 and the most significant bit corresponds to alert 8.

To illustrate, suppose that a sensor is actively monitoring 4 alerts and responds with the string “X1000A~\r\r”.

Converting “000A” to its binary equivalent yields “00001010” for the lower 8 bits. Figure 4 shows how these 8 bits indicate activity of alerts 1-8.

Figure 4: Example Parsing of Actuation Information

	MSB							LSB
Zone	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Bit Value	0	0	0	0	1	0	1	0
Presence	No	No	No	No	Yes	No	Yes	No

Getting Track File Data

SmartSensor Advance can simultaneously track the range and speed of up to 25 vehicle clusters. To request the current track file data, transmit “XT\r”. SmartSensor

Advance will respond with a 76-byte payload. The 76-byte payload of the “XT” response does not use the ASCII code. Instead these 76-bytes are bit and binary encoded. The response payload will be preceded with an “XT” header. The two-byte header is then followed by the first byte of the payload. The first byte is binary-encoded and indicates the length of the rest of the payload. The first byte has a decimal value of 75.

The last 75 bytes of the payload contain 25 track files in sequential order. Each track file is three bytes long and records the status, range, and speed of a tracked vehicle cluster. The first byte of each track file is the status byte, the second is the speed byte, and the third is the range byte of the vehicle cluster. The track files are listed in order 1 to 25, with the status, range, and speed of the first track file preceding the data for the status, range, and speed of the second track file, etc…

The status byte indicates whether a track file is active or inactive. An active track file is currently monitoring a vehicle cluster. An inactive track file is not currently monitoring a vehicle cluster.

The status byte also indicates when an active track file is ready to be read. Track file speed and range data should be ignored if an active track file is not ready to be read.

The status byte also indicates the first time a track file is active and ready for reading. This indication is provided in order to flag that the data in the track file corresponds to a newly discovered vehicle cluster. The polling application can use this flag to assign a unique identification to vehicle cluster tracked by the sensor.

The status byte also indicates whether a vehicle is approaching the sensor (or departing away from the sensor).

The status byte also indicates whether the vehicle is in the selected traffic direction (e.g. correct direction).

The status byte is bit-encoded as shown in the following table.

The range byte is binary encoded and represents the range of a tracked vehicle cluster with respect to the sensor. The distance to the sensor is encoded in 5-foot increments, meaning that a value of 1 corresponds to 5 feet, and a value of 255 corresponds to 1275 feet.

The speed byte is binary encoded and represents the speed in 1-mph increments. The minimum speed is 1 mph and the maximum speed is 100 mph.

Field		Description								Encoding	Length (Dec)	Start (Dec)
Residual Payload length		One byte-specifying the length of the payload (after the current byte).								Unsigned Integer Binary	1	0
Track File 1	**Status**	This indicates the status of the tracked object.								Bit-Binary	1	2
		B7	B6	B5	B4	B3	B2	B1	B0			
		Reserved	Reserved	Reserved	Approaching Sensor	Correct Direction	Ready to Read	Newly Discovered	Active			
	Distance to sensor	This indicates the range from the sensor to the range of the tracked object in 5-foot increments. The encoding for each tracker is an 8-bit unsigned integer that represents distances from 0 to 1275 feet.								Unsigned Integer Binary	1	3
	Speed	This indicates the speed of the tracked object in 1 mph increments from 1 to 100 mph. The encoding for each tracker is an 8-bit unsigned integer.								Signed Integer Binary	1	4
Track File 2		Status, Distance to sensor, Speed								Mix	3	5
…		…								…	…	…
Track File 25		Status, Distance to sensor, Speed								Mix	3	73
Total Size												76

The 76-byte payload is followed by a 4-byte checksum and the normal “~\r\r” terminator sequence.

A polling application receiving this message response should parse for the entire length of the payload (as specified by the length byte) instead of stopping pre-maturely if a value of 13 (ASCII carriage return) is detected. The value of the payload length byte

and the subsequent checksum can be used to determine if the message was reliably communicated.

It is anticipate that applications may poll for the track file data at about a 5 Hz rate. The length of the response is as long as 91 bytes (with a multi-drop prefix). The length of the request is 9 bytes (with a multi-drop prefix) Over a serial connection with stop/start bit encoding, a link that can support greater than 4.95 kbps is required to poll for this data at a 5 Hz rate.

Appendix A: ASCII Codes (Decimal) for Select Characters

Character	**Line Feed**			**Carriage Return ('/r')**					**Space**			**'~'**	
Code	10			13					32			126	
Character	**'0'**	**'1'**	**'2'**	**'3'**	**'4'**	**'5'**	**'6'**	**'7'**	**'8'**	**'9'**			
Code	48	49	50	51	52	53	54	55	56	57			
Character	**'A'**	**'B'**	**'C'**	**'D'**	**'E'**	**'F'**	**'G'**	**'H'**	**'I'**	**'J'**	**'K'**	**'L'**	**'M'**
Code	65	66	67	68	69	70	71	72	73	74	75	76	77
Character	**'N'**	**'O'**	**'P'**	**'Q'**	**'R'**	**'S'**	**'T'**	**'U'**	**'V'**	**'W'**	**'X'**	**'Y'**	**'Z'**
Code	78	79	80	81	82	83	84	85	86	87	88	89	90
Character	**'a'**	**'b'**	**'c'**	**'d'**	**'e'**	**'f'**	**'g'**	**'h'**	**'i'**	**'j'**	**'k'**	**'l'**	**'m'**
Code	97	98	99	100	101	102	103	104	105	106	107	108	109
Character	**'n'**	**'o'**	**'p'**	**'q'**	**'r'**	**'s'**	**'t'**	**'u'**	**'v'**	**'w'**	**'x'**	**'y'**	**'z'**
Code	110	111	112	113	114	115	116	117	118	119	120	121	122

Appendix B: Establishing a Connection using HyperTerminal™

Configuring the SmartSensor for communications is a simple process. Here is an example of how to connect a computer to the SmartSensor using an RS-232 link. To connect a PC to the SmartSensor, attach a null modem cable from a PC to the RS-232A connector of the SmartSensor cable (See Figure 1).

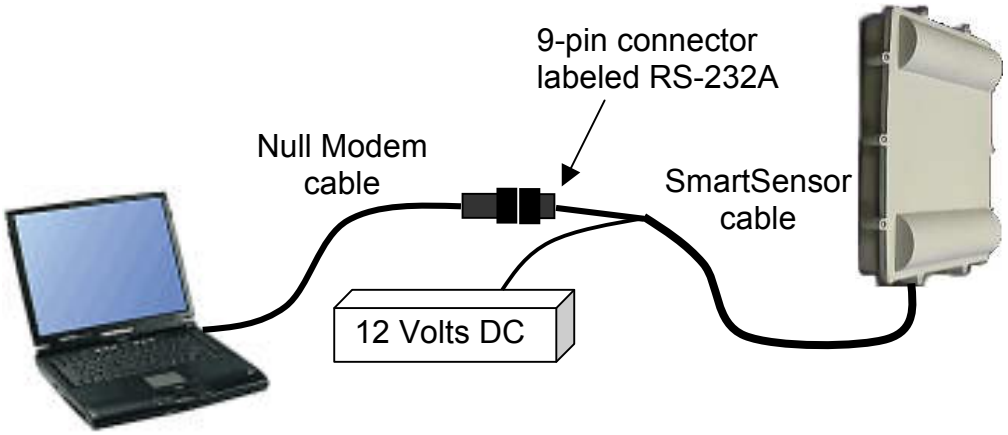


Figure 1. Connecting a PC to the SmartSensor

Open up a terminal program, such as HyperTerminal™, and connect directly to the Com port that is attached to the SmartSensor cable.

Figure 2. Selecting a COM Port in HyperTerminal

The RS-232 port is able to communicate at baud rates of 9600, 19200, 38400, 57600 and 115200. The RS-485 port is able to communicate at baud rates of 9600, 19200, 38400, 57600, 115200, 230400, 460800 and 921600.

The SmartSensor's default baud rate for the RS-232 interface is 9600. The default RS-485 baud rate is 115200.

The SmartSensor uses 8 data bits, no parity, 1 stop bit and no flow control. Select these settings under the COM port Properties window as shown in Figure 3.

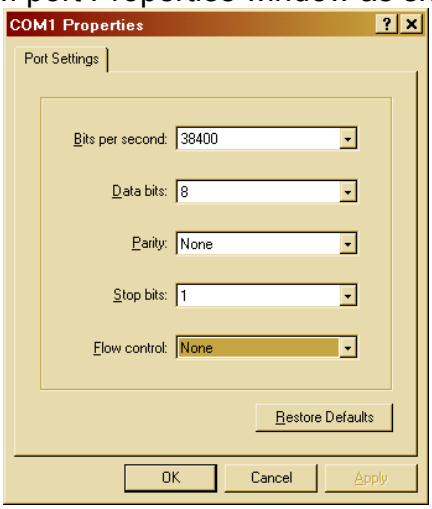


Figure 3. Setting the COM Port Properties

After clicking on OK the connection in HyperTerminal is made and commands can be sent to the SmartSensor. Also make the selections shown in Figure 4, under the menu File→Properties→Settings→ASCII.

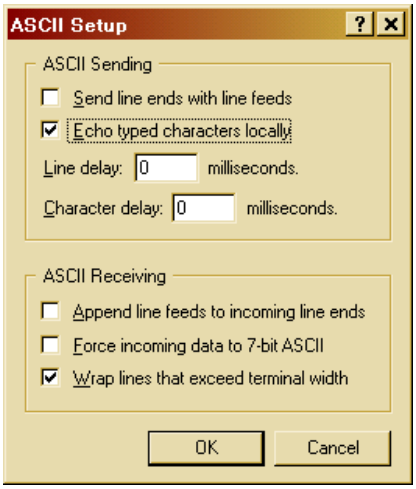


Figure 4. ASCII Setup