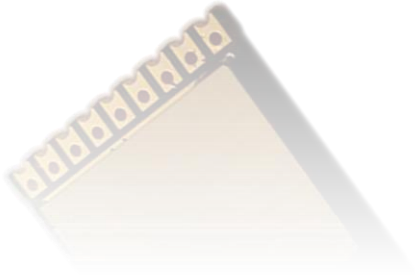
Whisker. Engine™

Module Technical User's Manual



Digital Six Laboratories Inc

www.d6labs.com

1 Table of Contents

2 Revision History

Revision	Date	Revisor	Description
0.A	12/15/14	SJM	Initial Revision
0.B	2/21/15	SJM	Added IO Channel Maps. Updated power modes
0.C	5/24/15	SJM	Updated to reflect firmware functionality of shipping modules
0.D	6/1/15	SJM	Corrected issue with IO tables
0.E	8/2/15	SJM	Added new AT commands, updated examples, updated IO Table
0.F	10/2/15	SJM	Fixed technical error in update notification documentation section.
0.G	10/20/15	SJM	Fixed errors in transparent serial mode section
0.H	6/28/16	SJM, HH	Fixed several technical errors throughout the document. Brought documentation up to date for feature changes.
0.I	8/28/2016	SJM	Fixed ordering codes
0.J	6/14/2017	SJM	Updated to reflect changes in Whisker.io™ platform

3 Introduction

The Whisker engine is a low cost, high performance, FCC pre-approved embedded wireless IO module capable of exceptional battery life and long range, making it perfect for Internet of Things applications. Actual power and range performance vary with configuration and environmental conditions. In general, for most battery configurations, the module can transmit up to 4 analog and digital channels every 5 minutes and operate for 5 years from 2xAA batteries. The height of the transmit and receive antennas (gateway and device) affect range; in most applications, the gateway antenna is elevated to 10+ feet and the devices are at least 3 feet above ground and have line of sight to the gateway antenna. In that configuration, Whisker.IO enabled devices can communicate with the gateway at ranges of 1+ miles. As the gateway is elevated higher, the range will increase.

While technologies like cellular and WiFi are great for IoT proof-of-concept projects, the short range, expense, size, and short battery life make it difficult to scale projects into production. Whisker.IO is inexpensive, easy to use, small, very low power, and has better range than cellular.

Building an IoT application using Whisker engine is straightforward and easy. Each module is equipped with analog, digital, and serial I/O and can be expanded with sensors and additional I/O via the I2C port. At the gateway, a host controls all of the Whisker.IO engine modules in the network using simple AT commands via the UART. These commands allow the host to read inputs, set output values, and send and receive bytes via the serial port on remote modules.

Whisker.IO engine modules are self-contained and require only a power supply and connection to the sensors and actuators to be monitored and controlled. Internally, they provide two 10-bit analog inputs, two digital inputs, one digital output, and a UART. The module also provides an I2C port that can be used to expand the internal I/O with high precision A/D converters, digital I/O expanders, A/D converters, and digital sensors (e.g. R/H, altimeter, accelerometer, etc.).

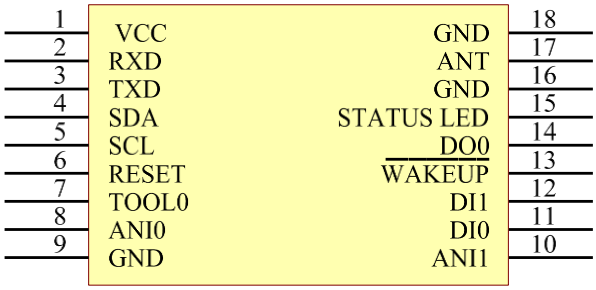
Any microcontroller, single board computer, or PC with a serial port (or USB virtual serial port) can be used as a host. Although the sensors and actuators can be separated by miles, application logic is centralized in the host and operates as if those sensors and actuators were all in the same location. If the host is connected to the Internet, application logic could be executed remotely on a cloud server, locally on the host, or some combination of the two.

Many applications are asynchronous and event driven, conforming to a "when this happens, do that" operating paradigm. Whisker engine excels in these applications. Using AT commands, the host can set triggers on any analog or digital input so that a notification is sent from the remote module when specific conditions are met.

Most applications also require periodic sampling of analog and digital values to build trends for future analysis. Interval triggers can be assigned to analog and digital inputs that will cause the remote module to sample the input and report the value at a specified interval.

Whisker engine modules can also be used as simple, long range, low power wireless serial modems for legacy applications where RS-232 or RS-485 serial communications are used. For more information, see section 11.1.

4 Pin-out

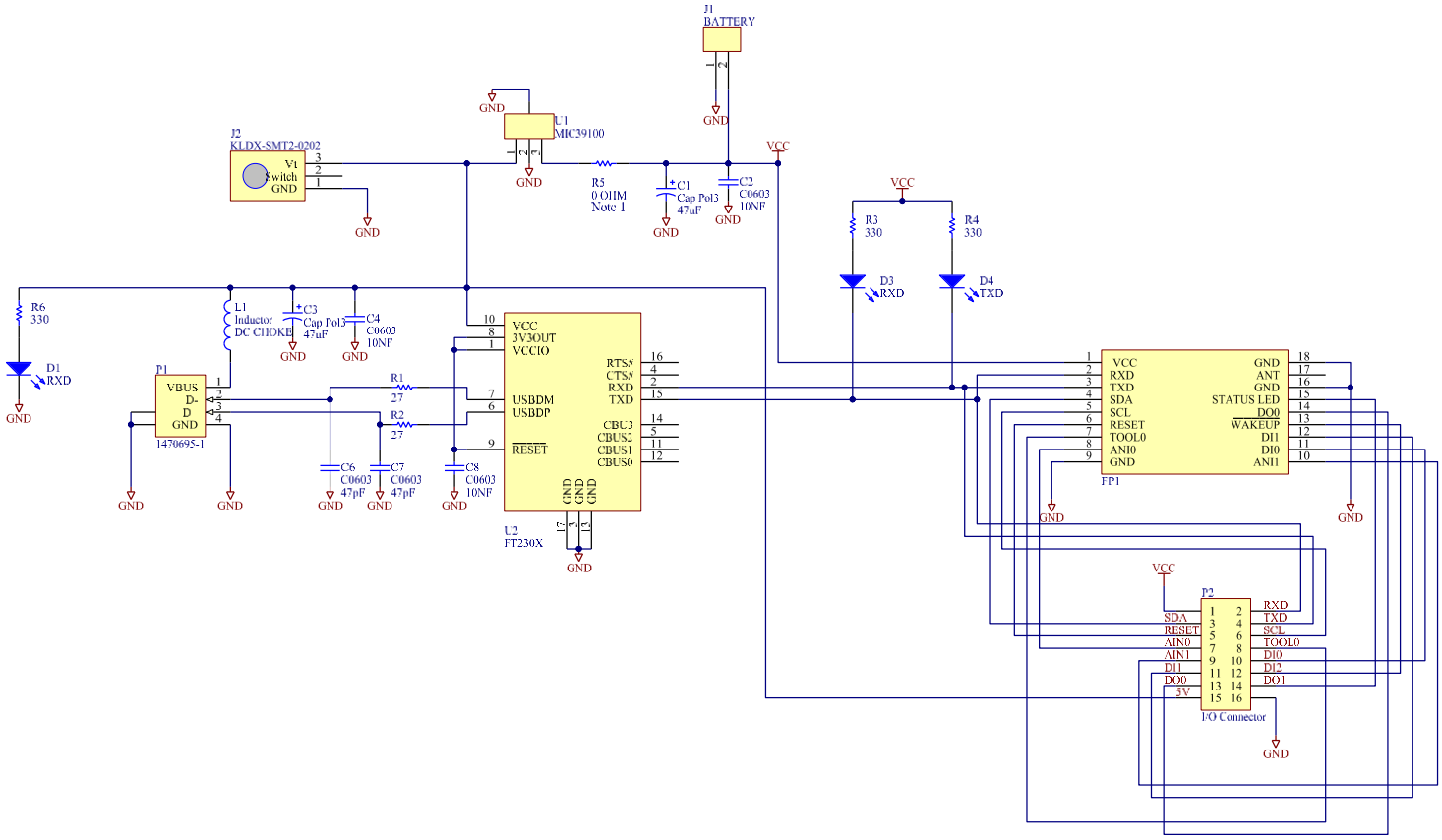


Pin	Description	Pin	Description
1	VCC: 2.4-3.6V DC (1.8 – 3.6V w/o adc)	10	AIN1: Analog input 1
2	RXD: UART data receive	11	DI0: Digital input 0 (Wakeup input in battery mode)
3	TXD: UART data transmit	12	DI1: Digital input 1
4	SDA: I2C IO Expansion	13	*WAKEUP
5	SCL: I2C IO Expansion (Serial mode)	14	DO0: Digital output 0
6	RESET: Programming line	15	STATUS LED
7	TOOL0: Programming line	16	GND*
8	AIN0: Analog input 0	17	ANT* (**)
9	GND	18	GND*

* These pins are not available on versions with a wire antenna or antenna connector. They are only used when the antenna is to be connected on the host PCB.

5 Typical Applications

5.1 USB Interface – Basic Master

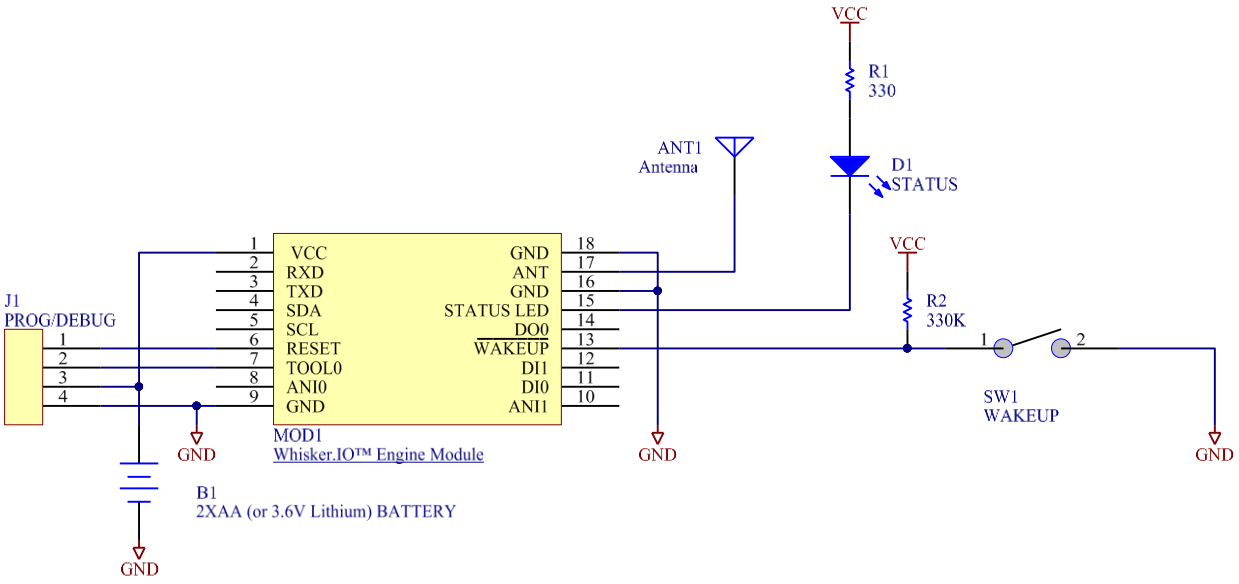


The easiest way to build a quick sensor network is to connect one Whisker.IO Engine to a PC via USB to act as a master or gateway.

This circuit allows the PC to talk to the module using a virtualized COM: port, making it easy to send local and remote commands using a terminal emulator. You can use any terminal emulator, but we recommend RealTerm; it is open source and free.

The COM: port settings are 115.2kbaud, 8 data bits, 1 stop bit, no parity, no flow control. Once you have the port configured that way, you can start typing AT commands and watch the incoming notification updates. The rest of the examples will assume that a master using this minimal circuit exists so that the described commands can be sent.

5.2 Basic Temperature Sensor



The Whisker.IO Engine is essentially a wireless IoT sensor in a module. Because it can measure temperature internally, all that is required to turn the module into a complete wireless temperature sensor is battery power and an antenna.

J1 is optional. It provides connection to hardware debugger and can be used to re-flash the firmware in the Whisker.IO Engine module.

Normally, a wireless sensor like this will operate in battery power mode. This mode is configured via remote AT commands. Prior to entering this mode, the module is also configured to report readings at a set interval. Once the battery mode is entered, the module will continue sleeping throughout the reporting period. When the period expires, the module will wake-up and transmit whatever readings it is configured to send, listen for 500mSec for an incoming response command, and then go back to sleep.

It might be necessary to change the configuration of the device after it has been deployed. There are two ways that this can work. First, the master or gateway can wait until the device wakes up and sends its periodic message and then immediately send the configuration command to the device within 500mSec. This requires that the gateway or master have a mechanism to queue commands (our commercial and industrial gateways can do this).

The second method is to "wake up" the device so that it is continuously listening for a period, making it easy to send new configuration commands. Our SensorBlocks are designed to work this way. The user simply swipes a magnet over the serial number label to wake up the SensorBlock and then the gateway can send commands to the SensorBlock to change its configuration. A reed switch is used to detect the magnet swipe.

In this example, SW1 is used to wake up a sleeping module; it can be a pushbutton or reed switch. When the WAKEUP pin is grounded, the module will wake-up and begin listening for commands via the LoRa radio. It will stay away for 1 minute.

The STATUS LED is used to indicate the internal status of the module. While sleeping, the LED is dark. While awake, it blinks every second. When the module is transmitting or performing some other function, the LED turns on for the duration of the event.

A very simple wireless sensor network can be built with temperature sensor nodes and a USB master as shown in 5.1. Using a terminal emulator such as Realterm, AT commands can be sent to the USB master that can configure the USB master, configure remote sensors, and read and write various I/O on the sensors.

For example, this sensor can be configured to periodically sample the temperature sensor every 5 minutes and report the reading to the master using the following AT commands.

For the purposes of this example, the sensor's MAC address (or serial number) is ***DE1A0E9C***:

Command	Local Response	Remote Response	Description
+++	OK		Puts local engine into command mode
ATTMDE1A0E9C	OK		Sets target MAC to sensor
ATMA	OK	RMOKrr	Sets sensor's master to remote WIOEN
ATAI0301	OK	RMOKrr	Enable analog channel 3 period (battery voltage)
ATAI1D01	OK	RMOKrr	Enable analog channel 29 period (internal temp)
ATSR00000005	OK	RMOKrr	Set periodic sampling to 5 minutes
ATRF	OK	RMOKrr	Save configuration to FLASH

The 'ATTM' command specifies the target MAC address for all remote AT commands that follow.

The 'ATMA' command is a remote AT command and is used to configure the target module's master MAC address. This way, when the target module needs to send an unsolicited message (like a periodic update), it knows which master module it should report to. By executing the 'ATMA' command, the target module becomes paired to the master.

The 'ATAI' commands are remote AT commands that are used to enable sampling of analog inputs. In this case, we are enabling channel 3, battery voltage, and channel 29, temperature.

'ATSR' is a remote AT command that sets the sampling period for all enabled channels, analog and digital. The sampling period is a 32-bit number measured in minutes.

'ATRF' is a remote AT command that will force the target module to copy all of its configuration into flash memory. This way, if powered is cycled on the target module, the operating configuration will be persistent. Only configuration data is persistent.

With these commands executed, the sensor will sample temperature every 5 minutes and transmit a response. For more information about AT commands and responses, please refer to 8.3.

Once every 5 minutes, the sensor module will transmit a periodic update message that contains the values of the analog channels being sampled. The USB gateway module will parse and validate the message and then it will generate a message through the UART for the host that looks like this:

RMPU0CDE1A0E9C4301315D009DDE

Section 9 describes the various responses and messages and how to parse them. This particular message is described in 9.2.3.1.

All numeric values are in human readable HEX format (without the leading 0x). Therefore a single byte requires 2 characters in the message.

'RMPU' is the message type indicator. 'RM' indicates that this is a remote response. 'PU' indicates that this is a periodic update.

The two digits directly after 'RMPU' indicate the length in bytes of the channel data field, which includes the channel indiciators and values for all channels included in this message. In this case, the value is 0x0C (or 12 decimal) because there are 2 channels that require 6 bytes each.

After the channel field length is the MAC address of the module that sent the update. In this case, it is 'DE1A0E9C'.

The channel field starts immediately after the MAC address. The channel field contains all of the channel readings and each reading contains a channel ID and value. The size of the channel value depends on the channel type. Analog inputs are always 16-bits (2 bytes).

The first reading is '430131'. The first two characters of this field are the channel ID, which is logically or'd with the channel type. The channel type is indicated by the upper 3 bits while the channel ID is indicated by the lower 5. In this case, the channel type is analog (0x40) and the channel ID is 3 (battery voltage). The value is 0x0131, or 305 decimal. The battery voltage is sent in fixed point format and must be divided by 100 to obtain the proper voltage, which is 3.05.

The second reading is '5D009D'. In this case the channel type is analog (0x40) and the channel ID is 0x1D (internal temperature). The value is 0x009D, or 235 decimal. Internal temperature is also sent in fixed point format and must be divided by 10 to obtain the proper value, which is 23.5 Celsius.

6 Electrical Specifications

Parameter	Minimum	Typical	Maximum	Units
Supply voltage – without A/D converter	1.8	3.0	3.6	VDC
Supply voltage – with A/D converter	2.4	3.0	3.6	VDC
Supply current – Transmit mode		120		mA
Supply current – Receive mode		13		mA
Average supply current – battery mode (5 min period)		60		uA
Average supply current – line power mode		13		mA
Receiver sensitivity	-125	-120		dBm
Link budget		135		dB
Minimum Demodulation SNR		-12.5		dBm (3)
RF data rate		7.031		kBaud
Analog input voltage range	0		2.5	Volts (1)
UART data rate - programmable	4800		115200	Baud
Unique MAC address length -	32		32	Bits (2)

Notes:

(1) 10-bit A/D converter (0=0V, 1023=1.45V)

(2) Each module is programmed at the factory with a unique 32-bit MAC address. This address is clearly marked on the module and is used to remotely command the functionality of the module.

(3) Typical GFSK radio (e.g. Bluetooth or ZWave) require at least +9dB SNR.

7 Ordering information

The WhiskerRF module can be configured with a number of antenna options:

1. PCB antenna – the antenna is connected to a pin on the module, allowing a PCB trace antenna to be used on the customer's board.
2. uFL connector – the module will be populated with a uFL connector. This configuration is normally used when the module's location requires a cable to go from the module to the enclosure wall. We offer uFL-RPSMA cables (purchased separately) for this configuration.

Ordering Code	Antenna Configuration
OEM-900-SC	902-928MHz surface mount package with uFL connector
OEM-900-SE	902-928MHz surface mount package – PCB antenna
OEM-900-TC	902-928MHz through hole package with uFL connector
OEM-900-TE	902-928MHz through hole package – PCB antenna

8 Command Mode

8.1 Entering Command Mode

The engine will enter command mode when three plus characters are received.

If the engine was not already in command mode, it will respond with 'OK'. At this point, all of the AT command described in this section should work.

If the engine is already in command mode when the three plus characters are received, it will stay in command mode but will respond with 'ER' the next time CRLF is sent. This is because the engine is expecting AT commands and +++ is not part of a properly formatted AT command.

8.2 Command Format

All commands are terminated with a CRLF (0xd 0xa).

Values sent in commands and responses must be in HEX format.

A hex encoded value uses two ASCII characters to describe the value of a byte. For example, a byte with a value of 0x20 (ASCII space) will be represented in the command as a 2 followed by a 0.

8.3 Command Table

AT commands are divided into two groups: local and remote.

Local commands are executed on the module that receives the commands directly through the module's serial port. Remote commands are sent to a target module via the LoRa wireless connection.

The following table shows the available AT commands and describes their function.

Command	Local/Remote	Description
ATCN	Local	Exit command mode
ATTMaaaaaaaa	Local	Sets the MAC address of the target module. All subsequent commands will be sent to the addressed module.
ATMA	Remote	Tell the target that we are master. All subsequent responses will be directed to this module and sent to the host via the UART interface.
ATSDxx	Remote	Sets the digital output (x) to high (1).
ATCDxx	Remote	Clears the digital output (x) to low (0).
ATRDxx	Remote	Reads the digital input(x). This only works with digital inputs. This command cannot be used to determine the state of a digital output. See 'Response' section for more information.
ATRAxx	Remote	Reads the analog input(x). This command cannot be used to read the value of an analog output. See 'Response' section for more information.
ATNDxx	Remote	Sets a digital input trigger on Dix. A response will be sent by the target whenever Dix changes once this trigger has been set.
ATAIxx	Remote	Sets analog input 'xx' to be sampled at the sample rate.
ATDIxx	Remote	Sets digital input 'xx' to be sampled at the sample rate.
ATSRxxxxxxxx	Remote	Sets the sample period for analog and digital inputs. 'xxxxxxxx' is a 32-bit number representing the sample period measured in seconds. If the period is 0, no samples are taken. The value of the sample period will affect battery life: please see the detailed explanation of this command for more information
ATNSbbtt	Remote	Sets the threshold of received bytes (bb) or maximum time between bytes (tt) on the target UART that will cause a response to be sent. Valid values for 'bb' are 0 through 31. 'tt' is the maximum time in mSec between bytes before a response is sent. So, even if 'bb' bytes are not received, if 1 byte is received and 'tt' byte times has expired, a response will still be generated.
ATLSbbtt	Local	Sets threshold of received bytes(bb) or maximum time between bytes (tt) on the local UART (serial port) that will cause a response to be sent.
ATBRbb	Local	Change serial port baud rate of local module. 'bb' indicates the desired baud rate.
ATSEllb0..bn	Remote	Send serial bytes to remote module. 'll' is the length in bytes. 'b0' thru 'bn' are the bytes to send. Each byte is encoded as two hex digits. Up to 32 bytes can be sent per message.
ATRBbb	Remote	Set's baud rate of remote module. 'bb' indicates the desired baud rate.
ATBP/ATLP	Remote	Set's target power mode. ATBP sets mode to battery power, ATLP sets mode to line power.

ATEKkkkkkkkkkkkkkkkkkkkkkkkkkkkkk	Local	Sets the module's encryption key. The same key must be used on all modules that are to talk to each other.
ATSCxx	Local	Sets the operating channel. See command details for more information
ATCN	Local	Exit command mode
ATWC	Local	Causes the local module to write its current configuration to NV memory. This will cause the configuration to persist across reset events.
ATRF	Remote	Causes the target (remote) module to write its configuration to NV memory. This will cause the configuration of the target module to perist across reset events.
ATPOmm	Remote	Enables tracking mode on modules connected to a GPS receiver via the UART. When the module receives this command, it will send the proper configuration commands to the GPS module. 'mm' indicates the mode. If 'mm' is zero, the GPS module is disabled. If 'mm' is greater than zero, the GPS module is enabled. If the target module is in battery mode with the GPS enabled, the GPS module will be placed into standby mode while the module sleeps. When the module wakes up, it will make the GPS module active and wait until a fix is available. It will then transmit the fix via an RMPO notification message.
ATDMxxyy	Remote	Sets the mode of a digital input. 'xx' indicates the digital input channel. 'yy' indicates the mode: 00 – normal input 01 – digital counter (32-bit)
ATSFxx	Local	Sets spreading factor on local module. Acceptable values for 'xx' are 7 through 11. NOTE: MODULES MUST HAVE THE SAME SF SETTING TO BE ABLE TO COMMUNICATE. INCREASING SF WILL INCREASE RANGE WHILE DECREASING DATA RATE.
ATRExx	Remote	Sets spreading factor on remote module. Acceptable values for 'xx' are 7 through 11. NOTE: MODULES MUST HAVE THE SAME SF SETTING TO BE ABLE TO COMMUNICATE. INCREASING SF WILL INCREASE RANGE WHILE DECREASING DATA RATE.
ATRCxx	Remote	Sets operating channel on remote module. See command details for ATSC command for more information
ATRM	Local	Get MAC address of local module

8.4 Command Details

8.4.1 ATCN – Exit command mode

Use this AT command to exit command mode.

8.4.2 ATTM – Set target MAC address

This command must be used before any other commands that operate on a remote module (target).

Example:

ATTMF21133E4

In this example, we are setting the target MAC address to F21133E4. Once this command is sent, all subsequent remote commands will be sent to the module whose MAC address is F21133E4.

It is up to the host to keep track of what Whiskers are operating in "battery power" mode. If a message is sent to a battery powered Whisker when it is sleeping, the message will be lost. The proper way to send messages to a battery powered Whisker is to queue the messages until an incoming message is received from the Whisker in question. When that message is received, the host can then send messages to the Whisker provided that the first message starts within 500mSec and each message is sent within 500mSec of the last message. The battery powered Whisker will go back to sleep automatically when it has no radio activity for a given period, with the actual time period depending on the manner in which the Whisker "woke up".

A battery powered Whisker can wake up two ways.

The two digital inputs will cause an interrupt when their state changes. This is the first wake-up method. If the module is configured with a trigger on the input that generates the interrupt, a notification message will be transmitted to the master. The module will stay awake for 500mSec after this message is sent so that queued messages can be transmitted by the master.

If the module is in battery mode, a timer will also generate an interrupt every minute which will wake up the module. If periodic sampling is enabled and the sample period has expired, any channels configured for periodic sampling will be sampled and a periodic update notification message will be transmitted to the master. The module will stay awake for 500mSec after this message is sent so that queued messages can be transmitted by the master.

In battery powered mode, the WAKEUP pin is dedicated to monitoring a switch that will "wake up" the module. This allows the user to purposely wake up the Whisker for configuration purposes. With this wake up mode, the "stay awake" period is 1 minute. If there is no radio activity in this time the Whisker goes back to sleep. Receive messages not addressed to the Whisker in question are ignored for this purpose.

See section 8.4.10 for more information about battery mode operation.

8.4.3 ATMA – Notify target of master

Once the target's MAC address is set, this command is used to notify the target that this module is the master. Once the target knows the master's MAC address, all responses are sent to that MAC address. This effectively links the target and the master.

A module can only be attached to one master at a time. Any module with the proper encryption key and on the same channel can execute ATMA and capture any other module. Therefore it is important that all modules be configured with a common secret encryption key prior to deployment.

8.4.4 ATSD, ATCD – Set and clear digital outputs

Once the target's MAC address is set, this command is used to set or clear digital outputs on the target module.

Example:

ATSD01– Sets digital output #0 to high on module F21133E4
ATCD01– Clears digital output #0 to low on module F21133E4

This example assumes that ATTM has already been used to set the target MAC address to 0xF21133E4

Since there is only 1 digital output on the module,1 is the only valid parameter for this command. The values are validated on the target side. If they are invalid, a remote error response is generated.

8.4.5 ATRD, ATRA – Read digital and analog inputs

Once the target's MAC address is set, this command is used to tell the target that we want to read the specified analog or digital port.

Example:

ATRD01 – Reads digital input #0 on module F21133E4. See 'Responses' for information on response
ATRA02 – Reads the power supply voltage of F21133E4. See 'Responses' for information on response

This example assumes that ATTM has already been used to set the target MAC address to 0xF21133E4

See section 10.1 for a complete listing of supported digital input channels.

See section 10.3 for a complete listing of supported analog channels.

Any other values will generate an error response. The values are validated on the target side. If they are invalid, a remote error response is generated.

8.4.6 ATDI/ATAI – Enable/disable periodic reporting for digital/analog input

This command will tell the target module to periodically send an appropriate response with the current value of the specified digital or analog input.

Example:

ATDI0101 (This sent the DIO which is indicated by 21 in the notification)

This will cause the target module to transmit the current value of DI0 every sample period. Substituting ATAI in the same command would cause the target module to transmit the current value of AI0 every sample period.

All channels are sampled at the same time as defined by the sample period set with the ATSR command.

When the sample period is reached, the Whisker will sample each digital and analog input specified by ATAI or ATDI commands and generate an update message with each channel. Up to 6 channels can be transmitted per update.

8.4.7 ATSR

This command sets the sample period of the target. The sample period is a 32-bit unsigned number measured in minutes.

Example:

ATAI0401
ATAI0501
ATSR0000000f

This example enables the temperature and relative humidity channels and sets the target's sample period to 15 minutes.

Setting the sample period to zero will disable all channels and stop the period update message.

8.4.8 ATND – Set notification on change in digital input

This command will tell the target module that a response should be sent whenever the specified digital input changes.

Example:

ATND01

This example will cause a response to be sent whenever DI0 on the target module changes state. 00 thru 0A are the only valid parameter values for ATND. The values are validated on the target side. If they are invalid, a remote error response is generated.

Notifications are transmitted instantaneously regardless of power mode.

8.4.9 ATNS/ATLS – Set serial notification threshold on remote/local module

These commands will tell the target/local module to generate a response whenever a certain number of bytes are received. A response is also sent whenever at least one byte has been received, but a specified number of byte times have expired since the last byte was received.

Example:

ATNS0A02↵↓

This example will cause a response to be sent when 10 bytes (0x0a) have been received –OR- at least 1 byte has been received and 2 mSec have expired since the last byte. These thresholds only apply in transparent serial mode; in other words, these thresholds apply when the module is not in command mode.

8.4.10 ATBR/ATRB – Set serial port baud remote on local/remote module

These commands are used to set the baud rate of the local/remote module. Valid baud rate settings are:

Setting	Baud Rate
0	2400
1	4800
2	9600
3	19200
4	38400
5	57600*
6	115200*

**** NOTE: the module is not capable of sustaining these baud rates unless a pause of 1mSec is inserted between bytes.***

The following example will set the local baud rate to 57.6kbit/second.

Example:

ATBR05↵↓

8.4.11 ATSE – Send serial bytes to target

Serial bytes can be sent to a target module in two ways. If the local module is not in command mode, it will automatically send all bytes received on the UART to the target module. This is called "transparent serial mode". If the local module is in command mode, the ATSE command is used to send serial bytes.

Example:

ATSE0B48656C6C6F20576F726C64

This example sends 11 bytes on the TXD pin of the target's UART:

Hello World

When bytes are received on the RXD pin of a target once ATMA has been executed on that target, those bytes will be sent as a response. See 'Response' for more information.

8.4.12 ATBM/ATLP – Set target's power mode

This command will configure the power mode of the target. In line power mode, the target will continuously listen for incoming packets. The ATLP command is used to set this mode.

In battery mode, the target will sleep until an I/O trigger wakes it up. A target that is operating in battery mode is capable of receiving commands from the gateway in two ways. First, every time the target sends a notification update

message to the gateway, the target stays awake for 500mSec after the transmission to listen for incoming messages. The master application (running on the host connected to the master) must queue messages for the target and begin transmitting them immediately after a notification update message is received. If the Whisker Network Manager is used, this will happen automatically.

In this mode, the target will reset the 500mSec timer each time a message is received from the gateway, allowing the gateway to send a chain of messages if required. If the timer expires without a radio event (TX or RX) the target will go back to sleep.

User intervention can also be used to wake up a battery powered target, allowing it to receive messages. For battery operated devices, DO0 becomes a wakeup input. An external switch is connected to this pin so that when a user presses the switch, the radio wakes. In this case, the sleep timer will be set to 5 seconds instead of 500mSec

This supports a manual provisioning mechanism using the following procedure:

1. Commands are queued at the master's host that changes the configuration of the target. Normally, the user would use a client application on a PC or mobile device to edit the configuration and send the new information to the master's host application. In the Whisker.IO IoT system, this is done with the Whisker.IO app. At this point, the host application on the master side is watching for an incoming message from the target, which will act as a trigger to send out the new configuration commands.
2. The user presses the wake up button on the target. The target will then send a status update to the master, which will act as a trigger to start sending down all of the information.
3. The master will then send the configuration commands to the target and get confirmation of the changes.

In both modes, notifications due to digital and analog triggers will be sent immediately by the target to the master.

8.4.13 ATEK – Set encryption key

Every module is shipped with a default encryption key. However, to make encryption effective for a given application, this key needs to be set by the user to a secret value. Only then can the module's communications be truly protected. Whisker engine uses 128-bit AES encryption, so a 16 byte key must be specified.

Example:

ATEK00112233445566778899aabbccddeeff

This example sets the encryption key to 0x00112233445566778899aabbccddeeff.

Modules should be configured with a encryption key prior to deployment. Keys can only be changed locally; no remote command can be used to change the encryption key.

8.4.14 ATSC/ATRC – Set channel

This command will set the channel used to send and receive data. ATSC sets the channel on the local module and ATRC sets the channel on the remote module.

Example:

ATSC0B – Sets channel to 11

The channel calculation is:

Frequency = 902.5 + 0.5 * channel (in mHz)

Therefore, valid values for the channel are 0 thru 50. Any other value will result in an error response.

If these commands are used to change the channel of one or more remote modules, then the ATRC command should be used first to change the remote channels and then ATSC should be used to channel the local channel.

8.4.15 ATDC – Display Configuration

This command will send all configuration information to the serial port in name value pairs.

TM=0x11223344
MA=0x44332211
EK=0x112233445566778899aabbccddeeff00
CH=0x0a
OK plus CRLF

Where TM is target MAC, MA is local MAC, EK is encryption key, and CH is channel.

8.4.16 ATWC / ARTF – Write Configuration

This command will save all configuration information to non-volatile memory. When the module comes out of reset, it will automatically load these variables, putting the module in the same state it was the last time the command was executed.

ATWC writes configuration to the local module and ATRF writes configuration to the remote module.

8.4.17 ATPO – Tracking mode

This command will turn tracking mode on and off, depending on the mode value. When tracking mode is enabled, commands are sent via the UART to an attached GPS module. If the module is there and responds properly, a RMOK response is sent to the master. If not, a RMER response is sent to the master.

To support battery operation, the GPS module is configured to sleep while the module sleeps. When the sampling period expires, the GPS module is set to active mode and the module waits for a fix and then transmits a position update notification message to the master.

Several tracking modes are supported:

Mode Value	Description
00	Tracking is turned off
01	Tracking on, response contains lat and long in human readable form

Human readable responses show the actual latitude and longitude as follows:

RMPOffeeddcc,35.5109,-97.7499,1,173,175

This response is formatted differently than the other formats so that the latitude and longitude can be represented in human readable form.

'RMPO' identifies this message as a periodic position notification.

'ffeeddcc' is the MAC address of the module sending the message.

The two decimal numbers following the MAC address are the latitude and longitude, respectively.

The next number, '1' in this case, is the speed over ground.

The next number, '173' in this case, is the angle.

The final number is the RSSI.

All values are in decimal form. Commas are placed between values to make parsing easy.

After reset, the GPS module will be active until the first fix is acquired. After first fix, the module is placed in a low power mode until the next sample period as set by sample rate (ATSR). When the next sample period is reached, the GPS module is taken active again until a fix is achieved and the result is reported to the master via the RMPO response, then the module is returned to the low power state.

As the sample rate is increased (in minutes/sample), battery life improves. If the sample period is less than 10 minutes, the GPS module should be able to hot start and obtain a fix in a few seconds. If the sample period is more than 10 minutes, it is possible for the GPS module to lose the timing reference meaning it will require a warm start, which means that it can take up to 32 second to get a fix.

Tracking mode is designed specifically for the MTK3339 GPS module from GlobalTop technologies. With this module connected, current consumption for the GPS module and the Whisker.IO Engine are as follows:

Mode	Current Consumption (Typical)
Sleep	500uA
Acquisition	35mA
Transmit	150mA

At our facility, we tested GPS operation with a 3 minute sleep period, ensuring that the GPS module had clear line of sight of the sky. Our tests showed an average fix time of a few seconds once first fix was acquired and a maximum current consumption of 2.5mA (average). Using an 18 A-H D-Cell LiThCl- 3.6 battery, estimated battery life is around 1 year.

Customers interested in using Whisker.IO with GPS should perform their own tests in the target environment and with the target configuration to determine suitability for any particular application.

8.4.18 ATDM – Set digital input mode for remote module

This command is used to configure a digital input as a counter on the target module.

When an input is in counter mode, it will increment a counter each time the module transitions from high to low, making it perfect for counting pulses in applications like water metering. The counter will start at 0 when the module is powered up.

The following example will configure DI1 as a counter on the target module.

Example:

ATDM0101

To restore DI1 to normal mode:

ATDM0100

8.4.19 ATRM – Read local MAC address

This command is used to get the MAC address of the local module.

Example:

ATRM

Response:

MA=FE112233

This response indicates that the local MAC address is 0xFE112233.

9 Responses

9.1 Local responses

When a command is sent by a host to a Whisker engine module, one of two responses is sent immediately via the serial port.

If the command was valid, the response will be:

OK followed by CRLF

If the command invalid, the response will be:

ER followed by CRLF

Every command will generate a local response.

9.2 Remote responses

All remote responses are prepended with 'RM' and are terminated with a "\n\r". These responses are actually generated by the remote module and are used to validate the command on the remote end and to format the response if information is required.

It is possible for a command to get an 'OK' local response and a 'RMER' remote response. This will usually happen when an invalid IO pin is specified in the command. Since not all Whisker engine modules are equipped with the same IO, it is possible for a command to be valid on one module and invalid on another. The local response does not take into account whether a command specifies a valid value, only that the value is properly formatted. All range checking is done on the remote end.

9.2.1 Decoding SNR and RSSI in remote responses

Every remote response is appended with the SNR and RSSI values measured during the message reception; the last 4 characters prior to the "\n\r".

Example:

RMOK03FE

In this example, the SNR is encoded as 03 and the RSSI is encoded as FE.

SNR and RSSI are both 2's compliment values.

SNR is calculated as follows:

```
if((snr & 0x80)==0x80)
        snr-=256;
```

RSSI is calculated as follows:

```
if( (rssi&0x80)==0x80)
        rssi-=256;
```

9.2.2 Command generated responses

When a remote module (target) receives a command, it will generate one of several responses. If the command does not require any data in the response (such as with ATSD and ATCD) and the command is valid, the response will be:

RMOKssrr followed by CRLF

If the command is errant, then the response will be:

RMERssrr followed by CRLF

Responses are generated automatically by the remote as soon as it executes the command. The sending module will continually listen, regardless, until either a valid response is received or the listen timeout expires.

9.2.3 Event generated responses

Events on a remote module (target) will also generate responses. The sources for event generated responses are:

1. Digital input trigger (via ATND command)
2. Analog input trigger (via ATNA command)
3. Serial bytes received trigger (via ATNS command)
4. Periodic sampling of analog and digital inputs

The responses are:

RMNDxxyyaaaaaaaassrr - digital input (x) changed on target module with MAC address(aaaaaaaa). Current value is 'yy'

Example: RMND0100F21133E4 – digital input 1 of module F21133E4 changed to '0'

RMNSaaaaaaaaccb0…bnssrr – 'cc' serial bytes were received on RXD pin on target module with MAC address (aaaaaaaa). Bytes (b0…bn) are hex encoded.

Example: RMNSF21133E40B48656C6C6F20576F726C64 – 'Hello World' was received on RXD pin of module F21133E4.

RMPO….. – the content length and format of this response will be determined by the tracking mode. See the ATPO command for more information

RMPUxxaaaaaaaav0…vnssrr – this is a complex, detailed response. Please see subsection 9.2.3.1 for more information

9.2.3.1 Periodic Notification

When the sample period (command ATSR) is set greater than zero, any analog or digital input channels that have been enabled for periodic sampling report their values at the set interval. Up to 4 channels can be set for periodic sampling, and each type of channel reports different value types. So, this response (notification) message is complicated.

The periodic notification response starts with RMPU and is followed by two digits that represent the length of the data portion and are followed by 8 digits that identify the MAC address of the sensor node that is sending the response. The remainder of the message contains the sample data for each included channel.

The various channels are reported as follows. All numerical values are hexadecimal.

Analog Input:
xxyyyy - where xx is the channel number (00 through 1F) and yyyy is the actual reading. The value is always the raw 16-bit reading. If the resolution is less than 16-bits, the value is shifted left an appropriate number of times to make it a 16-bit value. Analog channels will always have bit 6 set in 'xx'. Therefore, analog input channel 0 would be indicated as 'xx' = 41

Digital Input:
xxyy - where xx is the channel number and yy is 00 for off and 01 for on. A digital input in normal mode will have bit 5 of 'xx' set.

Digital Counter:
xxyyyyyyyy – where xx is the channel number is yyyyyyyy is the counter for the digital input. Any digital input can be configured as a counting channel using the ATDM command. A digital input in counter mode will have bits 5 and 6 of 'xx' set.

As an example, consider the following periodic notification response:

RMPU14DE1A0E9C5D00FB2101610000000F039A

Where (in order):

Command:
RMPU indicates that this is a periodic notification

Data count:
14 – is the data byte count and indicates that there are 20 bytes in that section. This does not include the MAC or rssi

Mac Address
DE1A0E9C – is the **MAC address** of the remote Whisker.IO Engine sending the notification

Channel Data
5D00FB – 5D indicates that the channel is analog input 1D (Bit 6 is set to indicate analog) and 00FB is the raw value, which in this case is the fixed point temperature. FB is 251 in decimal. The temperature is always stored multiplied by 10, therefore the temperature is 25.1 degrees C.

2101 – 21 indicates that the channel is digital input 0 in normal mode (Bit 5 alone is set to indicate a normal digital input). 01 indicates that the input is high or on.

610000000F – 61 indicates that the channel is digital input 0 in counter mode (bits 5 and 6 are set). 0000000F indicates that the counter value is 15 decimal

SNR
03 – 3dB signal to noise ratio

Rssi
9A – is the RSSI value (154 decimal). To calculate the actual RSSI, subtract 256 from this value to get the signal strength in negative dBm. In this case, the actual RSSI is -102dBm.

This example shows a typical periodic notification response message, including each type of channel, so that it can be used as a reference for how to parse these messages.

10 I/O Channel Maps

NOTE: These I/O channel maps are preliminary and are subject to change until the initial full release of Whisker firmware. In the code the channel 0 starts at the number 1 not zero. Therefore the digital input channel 0 would be represented by 01 in the code and 21 in the data response. This is the same for analog and digital representations.

10.1 Digital Inputs

CID	Name	Description
1	DIN0	Whisker digital input channel 0
2	DIN1	Whisker digital input channel 1
3	DIN2	Whisker digital input channel 2
4	MMA8652 FF	Free fall detection.
5	MMA8652 Tap	Tap detection
6	MMA8652 Motion	Motion is detected
7	RESERVED	Reserved
8	RESERVED	Reserved
9	RESERVED	Reserved
10	RESERVED	Reserved
11	RESERVED	Reserved
12	RESERVED	Reserved
13	RESERVED	Reserved
14	RESERVED	Reserved
15	RESERVED	Reserved
16	RESERVED	Reserved
17	SX1508_0	Digital input 0
18	SX1508_1	Digital input 1
19	SX1508_2	Digital input 2
20	SX1508_3	Digital input 3
21	SX1508_4	Digital input 4
22	SX1508_5	Digital input 5
23	SX1508_6	Digital input 6
24	SX1508_7	Digital input 7

* - These devices are not supported in the current version of firmware (version 1.4) but will be supported in the final version of firmware.

10.2 Digital Outputs

CID	Name	Description
1	DO0	Whisker digital output channel 0
2	NA	NA
3	SX1508_0	Digital output 0
4	SX1508_1	Digital output 1
5	SX1508_2	Digital output 2
6	SX1508_3	Digital output 3
7	SX1508_4	Digital output 4
8	SX1508_5	Digital output 5
9	SX1508_6	Digital output 6
10	SX1508_7	Digital output 7

* - These devices are not supported in the current version of firmware (version 1.4) but will be supported in the final version of firmware.

10.3 Analog Inputs

CID	Name	Description
1	AIN0	Whisker analog output channel 0. 10-bit resolution. Referenced to Vdd.
2	AIN1	Whisker analog output channel 1. 10-bit resolution. Referenced to Vdd.
3	Battery	Battery voltage. Fixed precision: 300 = 3.00V, 298 = 2.98V
4	Si7021 - Temperature	Temperature reading. +/- 0.3°C. Fixed precision: -4000 = -40.00°C, 2520 = 25.2°C
5	Si7021 – Humidity	Relative humidity. +/-2% humidity. Values 0-100.
6	RESERVED	Reserved
7	RESERVED	Reserved
8	RESERVED	Reserved
9	RESERVED	Reserved
10	RESERVED	Reserved
11	RESERVED	Reserved
12	RESERVED	Reserved
13	ADS1015_0 – AIN0	TI ADS1015/1115 12-bit A/D channel 0, addr=0x48
14	ADS1015_0 – AIN1	TI ADS1015/1115 12-bit A/D channel 1, addr = 0x48
15	ADS1015_0 – AIN2	TI ADS1015/1115 12-bit A/D channel 2, addr=0x48
16	ADS1015_0 – AIN3	TI ADS1015/1115 12-bit A/D channel 3, addr = 0x48
17	RESERVED	Reserved
18	RESERVED	Reserved
19	MPL3115 Pressure	Pressure in Pascals. 16 bit resolution. MSB
20	MPL3115 Altitude	Altitude in meters bit resolution. LSB
21	MPL3115 Temperature	Temperature – 16 bit Q12.4 format in ºC
22	RESERVED	Reserved
23	iAQ-core	Air quality – CO2
24	RESERVED	Reserved

25	RESERVED	Reserved
26	RESERVED	Reserved
27	RESERVED	Reserved
28	RESERVED	Reserved
29	Internal Temp	Internal temperature.
30	SI1133 Lux	Lux level. See datasheet for info
31	SI1133 UV Index	UV Index. See datasheet for info

10.4 Analog Outputs

CID	Name	Description
1	ADS1015_0 PGA	PGA setting, addr=0x48. See ADS1015 datasheet for information
2	ADS1015_1 PGA	PGA setting, addr=0x49. See ADS1015 datasheet for information
3	ADS1015_2 PGA	PGA setting, addr=0x4a. See ADS1015 datasheet for information
4	ADS1015_3 PGA	PGA setting, addr=0x4b. See ADS1015 datasheet for information
9	DAC121_0	12-bit D/A, addr = 0xc0
10	DAC121_1	12-bit D/A addr = 0xc1
11	DAC121_2	12-bit D/A addr = 0xc2
12	DAC121_3	12-bit D/A addr = 0xc3
13	RESERVED	Reserved
14	RESERVED	Reserved
15	RESERVED	Reserved
16	RESERVED	Reserved
17	RESERVED	Reserved
18	RESERVED	Reserved
19	RESERVED	Reserved
20	RESERVED	Reserved
21	SX1508_0	8-bit PWM value. Setting this value sets pin to PWM output.
22	SX1508_1	8-bit PWM value. Setting this value sets pin to PWM output.
23	SX1508_2	8-bit PWM value. Setting this value sets pin to PWM output.
24	SX1508_3	8-bit PWM value. Setting this value sets pin to PWM output.
25	SX1508_4	8-bit PWM value. Setting this value sets pin to PWM output.
26	SX1508_5	8-bit PWM value. Setting this value sets pin to PWM output.
27	SX1508_6	8-bit PWM value. Setting this value sets pin to PWM output.
28	SX1508_7	8-bit PWM value. Setting this value sets pin to PWM output.

10.5 Device Specific Information

Digital Six Laboratories provides several application notes describing how to use the module with the various I/O peripherals listed in the channel I/O table. Please refer to those application notes for more information.

11 Applications

11.1 Transparent serial mode

The Whisker.IO Engine is capable of operating as a mostly-transparent serial modem. It will pass any bytes received on the local UART to a remote module (specified by the ATMA command from the remote module) and will pass any bytes received on the remote UART to the local module, providing a full duplex link.

When a module is in command mode, the transparent serial capability is disabled on that module, though it can receive serial data from remote modules via the RMNS response. **When it is not in command mode, it is automatically in transparent serial mode.**

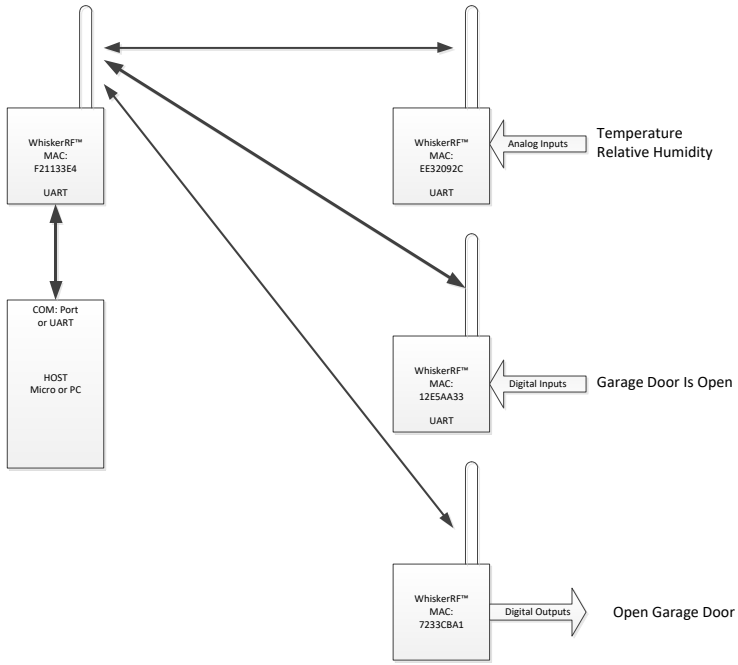
Transparent serial mode converts a Whisker engine module into a transparent wireless serial modem. It transmits all information it receives in the RXD pin over the air and all data it receives over the air is in turn transmitted over the TXD pin.

For more information about using the Whisker.IO Engine in a serial modem application, see ***Application Note: AN100 Using Whisker.IO As A Serial Modem***

11.2 IoT Wireless Sensor Network – AT Command Interface

With Whisker engine building two-way wireless sensor networks is very easy, especially when the Whisker Network Manager is used. However, there might be some applications where the WNM is not suitable, forcing the developer to use the underlying AT command interface to control the network. In this section, we will examine a simple application and how the AT commands can be used to build it. It is good to understand how the AT commands work even if you are using the WNM.

The following diagram shows a simple application.



In this example, a master (on the left) is connected to a host (microcontroller or Windows/Linux PC). Three additional modules operate as stand-alone wireless I/O devices. The top right module is connected to analog sensors, the middle right module is using its digital outputs to monitor the state of a garage door, and the lower right module is using its digital output to actually control the garage door.

Here are the commands that the master module would send to configure the network (CRLF omitted for space):

ATTM01EE32092C
ATMA
ATTM12E5AA33
ATMA
ATTM7233CBA1
ATMA

With this basic configuration complete, each of the sensor modules are now configured to communicate with the master module.

Now, to read the temperature value on module EE32092C, we would execute the following command:

ATTM01EE32092C
    Response: OK
ATRA03
    Response: OK
    Response: RMRA03014ac0

These commands configure the master to communicate with EE32092C and then request a reading from analog channel 3(battery level). On the second command, two responses are received. The first comes from the local module and acknowledges the proper execution of the command. The second comes from the remote module and contains the requested reading. In this case the reading is 0x014a (decimal 330). This indicates a battery voltage of 3.3V
Likewise, we could use the following commands to read the garage door open status:

ATTM0112E5AA33
Response: OK
ATRD01
Response: OK
Response: RMRD01

In this case, the remote module indicated that the garage door is open sensor is active (value=1). So we might want to go ahead and automatically close the garage door to make sure that none of our awesome NASCAR memorabilia gets stolen:

ATTM7233CBA1
Response: OK
ATSD01
Response: OK
Response: RMOK

The command ATSD01 sets the digital output 0 to a logic level 1, which will close the garage door.

Obviously, we wouldn't want to continuous send the ATSD command over and over to check the state of the door. There are two methods we can use that allow the remote module to automatically send the state to us.

First, we could set a report interval and have the remote module automatically send the state of the door at regular intervals. This method is great for building a periodic historical record of values to build a trend. However, in this case, we care more about the event of the door changing state than the trend. So we would choose the second method which is to set an event trigger to notify us when the state changes:

ATTM12E5AA33
Response: OK
ATND01
Response: OK
Response: RMOK

With this trigger in place, the remote module will send an event response whenever the garage door sensor changes state:

Response: RMND000112E5AA33

This response tells us that digital output #0 of 12E5AA33 has changed and the current value is 1.

The beauty of event based responses like this is that the remote module only transmits when something changes state. So long as the output remains the same, no transmissions are made, saving battery life and bandwidth.

If the master ever needs to reset a remote module to its default state and remove any triggers, it simply needs to execute the ATMA command for that module again.

NOTE: HYPOTHETICAL APPLICATIONS ARE FOR DISCUSSION PURPOSES ONLY. IT IS THE RESPONSIBILITY OF THE USER TO DETERMINE WHETHER WHISKER ENGINE MODULES ARE SUITABLE FOR A GIVEN APPLICATION. THESE MODULES SHOULD NOT BE USED IN MISSION CRITICAL CONTROL SITUATIONS AS THEY OPERATE IN AN UNLICENSED BAND AND

MUST, BY LAW, ACCEPT INCOMING INTERFERENCE. THEREFORE, NO APPLICATION SHOULD RELY ON ANY WHISKER ENGINE MODULE FOR ANY MONITORING OR CONTROL FUNCTION RELATED TO SAFETY OR PREVENTION OF DAMAGE TO EQUIPMENT.

11.3 Notifications: Periodic and Event Driven

There are two ways to obtain the value of an analog or digital input on a remote module. First, the master could send an ATRD or ATRA command to request the current value. This method is good for infrequent requests where an immediate response is required. Using this method is inefficient; however, because it requires two complete transmit cycles: one when the master sends the command and the other when the remote module responds.

The second method is to use the notification response system of Whisker engine. Remote modules can be configured so that they automatically transmit the value of a given output based on some condition, without the need for a request from the master.

Two types of notifications are supported: Periodic and Event Driven.

Periodic notifications are used in applications where trending needs be tracked using periodic samples. A good example would be an application where the ambient temperature needs to be tracked. While these notifications are sent in real time, they are not intended for fast real time systems. They are meant to track relatively slow processes where the tracked variable changes slowly over time.

Event notifications are used in applications where only the change in state needs to be reported. A good example of this kind of notification would be an application where some action is required when the state of a switch changes. Once the condition is met, a notification is sent immediately. Therefore, these notifications provide faster response than periodic notifications.

Periodic notifications are configured using the ATSR, ATAI and ATDI commands. Event notifications are configured using the ATNA and ATND commands.

Periodic and event notifications can be used together. For example, consider an application where AI0—which is represented by 01 in the command below—is connected to a tank level transducer. The tank's level changes relatively slowly, so we would record the tank level trend by using the following command:

```
ATAI0101
ATSR00000001
```

This would configure the remote module to report the tank's level every minute.

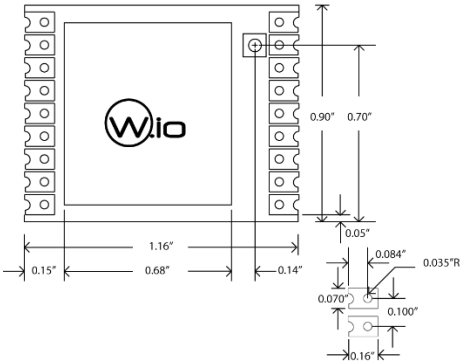
To make sure that the tank doesn't overflow, we also want an instant notification when the tank level reaches 85% of full scale. To accomplish that, we use the following command:

```
ATNA0000000366
```

This configures the remote module's AI0 channel to generate a response whenever the value reaches 0x0366 (85% of full scale). The lower limit of the window is set to 0x0000, effectively making the trigger a high level only trigger. If a minimum tank level also needed to be reported, that value would be changed accordingly.

NOTE: HYPOTHETICAL APPLICATIONS ARE FOR DISCUSSION PURPOSES ONLY. IT IS THE RESPONSIBILITY OF THE USER TO DETERMINE WHETHER WHISKER ENGINE MODULES ARE SUITABLE FOR A GIVEN APPLICATION. THESE MODULES SHOULD NOT BE USED IN MISSION CRITICAL CONTROL SITUATIONS AS THEY OPERATE IN AN UNLICENSED BAND AND MUST, BY LAW, ACCEPT INCOMING INTERFERENCE. THEREFORE, NO APPLICATION SHOULD RELY ON ANY WHISKER ENGINE MODULE FOR ANY MONITORING OR CONTROL FUNCTION RELATED TO SAFETY OR PREVENTION OF DAMAGE TO EQUIPMENT.

12 Mechanical



* Units are in inches.

13 Agency Certifications

13.1 United States FCC

The Whisker.IO Engine module complies with Part 15 of the FCC rules and regulations. Compliance with the labeling requirements, FCC notices and antenna usage guidelines is required.

To fulfill FCC Certification, the OEM must comply with the following regulations:

1. The system integrator must ensure that the text on the external label provided with this device is placed on the outside of the final product.
2. Whisker.IO Engine module may only be used with antennas that have been tested and approved for use with this module.

13.1.1 OEM Labeling Requirements

The Original Equipment Manufacturer (OEM) must ensure that FCC labeling requirements are met. This includes a clearly visible label on the outside of the final product enclosure that displays the contents shown in the figure below.

> Contains FCC ID: 2AFTI-WEN9*
>
> The enclosed device complies with part 15 of the FCC rules. Operation is subject to the following 2 conditions: (i). this device may not cause harmful interference and (ii). This device must accept any interference received, including interference that may cause undesired operation.

13.1.2 FCC Notices

Important: The Whisker.IO Engine module has been certified by the FCC for use with other products without any further certification required (as per FCC section 2.1091). Modifications not expressly approved by Digital Six Laboratories could void the user's authority to operate the equipment.

Important: OEMs must test final product to comply with unintentional radiators (FCC section 15.107 and 15.109) before declaring compliance of their final product to Part 15 of the FCC Rules.

Important: The RF module has been certified for remote and base radio applications. If the module will be used for portable applications, the device may require additional SAR testing. Consult Part 15 of FCC rules for more information.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures: Re-orient or relocate the receiving antenna, Increase the separation between the equipment and receiver, Connect equipment and receiver to outlets on different circuits, or Consult the dealer or an experienced radio/TV technician for help.

13.1.3 RF Exposure

CAUTION: To satisfy FCC RF exposure requirements for mobile transmitting devices, a separation distance of 20 cm or more should be maintained between the antenna of this device and persons during device operation. To ensure compliance, operations at closer than this distance are not recommended. The antenna used for this transmitter must not be co-located in conjunction with any other antenna or transmitter.

The preceding statement must be included as a CAUTION statement in OEM product manuals in order to alert users of FCC RF Exposure compliance.