SenSyr LLC

Suite 2-212
111 College Place
Syracuse, New York 13244
(315) 445-8701
(413) 803-3016 Fax
sales@sensyr.com

TNG-5:

The TNG-5 family of devices consists of three models: the full version (TNG-5), a mid-scale Version (TNG-5 Lite), and a minimal version (TNG-5 UltraLite). A prototype of the full version is currently being developed and tested. A comparison chart of all the TNGs, including the anticipated TNG-5 line is shown below.

	TNG-3b	TNG-4	TNG-5 Full Version	TNG-5 Lite	TNG-5 UltraLite
I/O Connectors	Phone Jacks	RJ-12	RJ-45	RJ-45	RJ-45
Analog Inputs	8	8	16	8	8
Analog Input Resolution	8-bits	8/10/12 bits[1]	10-bits	10-bits	10-bits
Digital I/O Lines	8 (input only)	16 User-defined I/O	16User-defined I/O	16 User-defined I/O	16 User-defined I/O
Analog Outputs	None	4 8-bit (0-4 V)	None	None	None
SPI Ports	none	1 Port; 8 software-selectable enable lines	2 Ports; 8 software-selectable enable lines	2 Ports; 8 software-selectable enable lines	1 Port; 8 software-selectable enable lines
RS-232 Ports	none	none	1 Port - either TTL or true RS-232 levels dependent on how the board is populated.	none	none
User-definable ports	none	none	2 Ports. Each port supports up to 4 lines. Function is defined by the installed mezzanine function board.	none	none
Interface	RS-232 19.2 kbps	Bidirectional RS-232 19.2 kbps[2]	Bidirectional LAN / USB / RS-232 128 kbps[3]	Bidirectional LAN / USB / RS-232 128 kbps[3]	Bidirectional RS-232 128 kbps
Streaming Mode Sample Rate	192 samples/s	~190 samples/s	~400 samples/s	~800 samples/s	~800 samples/s
Power	Serial Port Powered Only	RS-232 Port / External Power	USB or RS-232 Powered / External Power	USB or RS-232 Powered / External Power	RS-232 Port / External Power
Operating Modes	Streaming	Streaming or Command: Depends on installed firmware.	Streaming or Command: Depends on switch setting.	Streaming or Command: Depends on switch setting.	Streaming or Command: Depends on switch setting.
ICSP	No	No	Yes	Yes	Yes
Downloadable Firmware	No	No	Yes[4]	Yes[4]	Yes[4]
Electrical Isolation	No	Yes	Yes	Yes	No

[1] Depends on installed processor.
[2] Speeds up to 57.6 kbps optional.
[3] Only one interface can be implemented on any one board.
[4] If firmware supported

Fig. 1. TNG Comparison Table

TNG-5 dramatically increases the capabilities of TNG. A limited model, TNG-5 “lite” effectively replaces TNG-4 with a similar set of I/O; however, the 4-channel DAC will be expunged in favor of a second SPI expansion port. The full TNG-5 has 16 analog input channels, 16 digital I/O lines, 2 SPI expansion ports, an RS-232 port (RJ-12 connector), two special-purpose ports, and an optional connector for the insertion of a Multi-Media Card (MMC) memory module.

Both the full and lite versions of TNG-5 support 10-bit ADC resolution, external power options, electrical isolation of the I/O ports from the computer interface, two independent voltage regulators, USB communications, and faster data rates. Board population options allow continued support for legacy RS-232 applications. Either TNG-5 will have about 60 milliamps (mA) of available power when connected as a bus-powered USB device without an external power source. The addition of an external power source allows for up to 500 mA of power for TNG-5 operations.

The TNG-5 "UltraLite" version only supports an RS-232 interface, has eight 10-bit resolution analog inputs, a digital I/O lines, and one SPI port. Electrical isolation is not supported. This version is intended for use with handheld computers.

TNG-5 connectivity changes yet again. Instead of RJ-12 modular connector, all TNG-5 versions use 8-conductor, RJ-45 modular connectors. The RJ-45 connectors should prove to be more robust than the RJ-12 modular connectors. Additionally, the RJ-45 connectors possess two more pins—one is used as an additional ground conductor, and the other used to provide either the principal TNG-5 operating voltage (Vcc) or the voltage from the secondary regulator (jumper selectable).

Both the full and lite versions of TNG-5 provide for an optional secondary regulated power bus that can be made available by jumper selection on all the RJ-45 ports. The secondary regulated power bus is down-regulated from the primary regulated power. This secondary voltage is resistor-programmable (as is the primary supply), and can be set in the range of 1.2 through (Vcc-0.4) volts. Now it becomes possible to operate sensors and peripherals at 3.3 V without down-regulating at the peripheral, or operating the entire TNG at 3.3 volts.

Normally, TNG-5 is configured such that it is powered off until DTR is asserted. If powered only through the external power jack, TNG-5 can be wired to be always on for any interface. The USB interface supports power up on enumeration as a population option.

The full and lite versions of TNG-5 support an ethernet interface as a population option. Such a web-enabled TNG-5 has huge potential for new applications. Now the TNG-5 and its host computer can be miles, even continents, apart. 802.11 wireless connectivity is possible with a proper application of wireless bridges and/or access points. A TNG-5 operating as an ethernet device does have some limitations: the device can only be operated with an external power source, and there would be no electrical isolation feature.

All the TNG-5 models can be programmed and reprogrammed via the in-circuit serial programming (ICSP) header. This feature allows the firmware to be updated as necessary.

Knowledgeable end-users could adapt TNG-5 devices to their own specialized applications by writing their own firmware.

Even more exciting is the potential of downloading new firmware to TNG-5 devices over the device's interface connection. This feature will not only make firmware updates virtually painless, but opens the door to not nearly instantaneous adaptation of TNG-5 for various end-user applications.

All that past TNG firmware has done is essentially pass data to and from the host computer. If the application required some closed loop control or condition monitoring, the host computer had to do it. With the advent of downloadable firmware, custom TNG-5 applications can be developed, and the loop can be closed in TNG-5. In other words, TNG-5 can be given the smarts to make its own decisions and process its own data.

For example, let's say the desired application is to monitor and maintain a water bath at a constant, set temperature. TNG-4 could be set up to perform this function, but the host computer would have to continuously run a NeatTools or LabView program. TNG-5 with customized firmware could be configured to run this application independently. The host computer wouldn't even have to be on. The host computer, then, would only have to communicate with the dedicated TNG-5 to monitor the current temperature or set a new temperature. A web-enabled TNG-5 could be programmed to perform the application function, as well as to send emergency e-mail alerts should a fault condition be detected.

These are three unique features of the full TNG-5 that vastly expand its utility over previous versions.

The full version of TNG-5 has a population option that supports a multi-media card connector. A TNG-5 with this feature could now be programmed to act as a data logger. Using a 64 MB MMC, and assuming a 64 byte storage packet, over 1 million time-stamped data packets could be stored. At the rate of one point per second, more than 10 days of data could be accumulated, or more than one day's data at 10 samples per second. A 256 MB MMC has room to store 100 64-byte data packets per second for more than 24 hours.

The second unique feature of the full version of TNG-5 is its RS-232 port. This feature allows TNG-5 to connect to peripheral devices that require an RS-232 interface. There's great flexibility in how the support can be implemented. Population options allow the support to be used as a genuine, 3-wire RS-232 serial interface using standard RS-232 voltage levels. Alternatively, the port can source inverted or non-inverted TTL-level RS-232 signals. It all depends on how the board is populated. When TNG-5 is operated with an external power source, this RS-232 port is electrically isolated from the host computer, provided the attached serial device is also not electrically connected to the host computer.

The third in each feature of the full version of TNG-5 is its two special-purpose ports. The power and ground pins of these ports are allocated the same as with all the other TNG-5 ports. The four signal pins of a special-purpose port, however, are defined by its corresponding mezzanine card. A mezzanine card is a special-purpose printed circuit board approximately 1.8 in. long by 0.8 in.

wide. The mezzanine card has two sets of pins on either end that allow it to mate with sockets on the TNG-5 printed circuit board in only one orientation, preventing erroneous insertion. The mezzanine card's interface to the TNG-5 microcontroller makes the mezzanine card appear to be another SPI port.

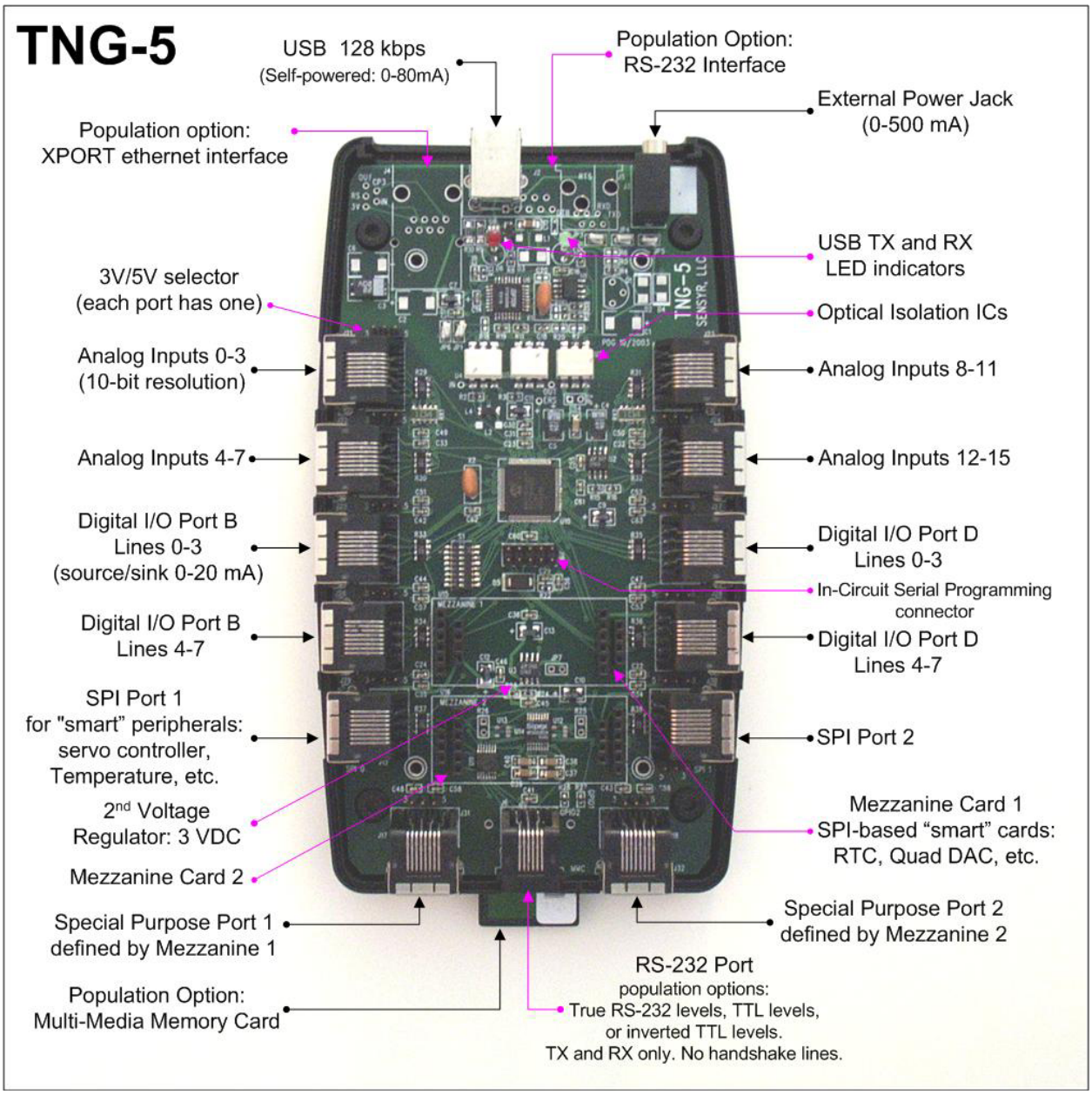
This feature conveys immense flexibility to TNG-5. Mezzanine cards can be made to perform a wide variety of functions, and don't even have to define the pins on the special-purpose port. An example of this might be a real-time clock module to assist in data logging applications. Just some of the possible mezzanine card candidates include 4-channel DACs of varying resolutions, a high-resolution 4-channel ADC, a precision strain gauge interface, and a multichannel thermocouple interface. Another choice would be a generic mezzanine card for prototyping application-specific circuits. The list is almost endless.

The real payoff, however, is that the use of all of these multifunction cards would not be restricted to the full version of TNG-5. A carrier board can be made such that the TNG-5 mezzanine cards will also plug into the carrier board, and because the TNG-5/mezzanine card interface is based on SPI, the combined carrier board in mezzanine card can be attached to any SPI port on either of the other two TNG-5 versions. This eliminates having to develop separate printed circuit boards for mezzanine use and external use.

So there it is: TNG-5 packs quite a wallop! TNG-5 has different versions for different application environments. TNG-5 has new, faster host interfaces. ICSP and downloadable firmware, make theTNG-5 family of devices more adaptable to applications, and open the door to stand-alone control or data logging applications. The full version of TNG-5 sports new interfaces and ports, and eight more analog inputs.

TNG-5 Feature Bullets:

- USB Interface: acts as a virtual COM port at 128 kbps (max).
- Asserting DTR powers up TNG-5.
- 16 ADC inputs, 10-bit resolution, 0-5 VDC input range.
- 16 Individually-configurable digital I/O lines (source/sink 20 mA).
- 2 Built-in SPI expansion port connectors.
- 2 Mezzanine-defined SPI-like port connectors.
- 1 RS-232 port (true RS-232 or TTL levels (inverted/non-inverted); TX and RX only).
- Optional MMC interface.
- Each port connector (except the RS-232 port) has 2 ground and 2 power pins where one of the power pins can be selected as either the primary (5V) or secondary (3V) power source via a board jumper.



Software Considerations:

The addition of new features and interfaces present a host of software issues relative to previous TNG versions that need addressing:

- **ADC resolution:** Previous command mode versions allowed either 8- or extended- bit resolution. Streaming software had 8-bit versions and a special 10-bit version for the TNG-4X. I see no reason to be able to switch resolution with this faster device. If all you want is 8-bit resolution, throw away the LSB (see next section).
- **ADC result justification:** Previous command mode versions supported left- or right-justfied bits in the result. The TNG-4X module used left-justified results. Again, I see no reason to

support both justification schemes. The left-justified version has its advantages: ADC results from all ADC resolutions (8, 10, and 12-bits) can be treated the same.

- **ADC channels:** Previous versions only supported 8 channels. New software needs to accommodate 16 channels for the full TNG-5. The number of ADC channels should be a module parameter.
- **SPI Ports:** Previous software versions supported a single SPI port. Port expansion was accomplished by using Port D bits to create extra chip select lines. The full and lite versions of TNG-5 have two built-in SPI ports (Just wait. It gets even better) while the ultralite version has just one. The question is whether to treat the ports separately or the same.

  The problem with treating the ports as separate is that they really aren't. The clock and data lines are shared by all SPI and Mezzanine ports. Data cannot be exchanged with multiple SPI devices simultaneously.

  On the other hand, the select lines used by default for each port are physically different and are not really usable by the others. Moreover, the physical layout would make it better to associate Port B select lines with SPI port 1 and Port D select lines with SPI port 2. I'm inclined toward supporting separate SPI ports, while recognizing that this might confuse an end user who might have a hard time understanding why messages can't be sent to the two SPI ports simultaneously.

  Another issue is speed. The internal speed of the TNG-5 microcontroller is 32 MHz. The TNG-4 SPI clock was 1 MHz in its default configuration. Most existing SPI-compatible devices in our inventory cannot be clocked faster than 10 MHz. Of greater concern is the effect of rates above 1 MHz on useful cable lengths, particularly when those cables are constructed from ribbon cable. Therefore, SPI clock speed has to be intelligently addressed for TNG-5.
- **Mezzanine Ports:** These two ports only exist on the full version of TNG-5. The interface with the microcontroller is effectively another SPI port. Each mezzanine card has two chip select lines, and two general purpose digital I/O lines. Again, the clock and data lines are shared with each other and the SPI ports. Again, I support the idea of treating them as separate ports.
- **RS-232 Port:** This port only exists on the full version. It is my intent to treat it substantially like an SPI port. The baud rate should be software selectable. There are two TLL digital I/O lines associated with this port. I don't think that this port would really support continuous bidirectional asynchronous serial communications at any real speed. The intent was to be able to be able to talk to legacy peripherals (like a blood pressure monitor)—the kind of communications where a command string is sent and maybe a response returned (like a command mode TNG-4).
- **MMC:** The MMC device looks like (guess what?) another SPI port. This feature was envisioned for situations where TNG-5 is working as a data logger in a stand-alone capacity. Moreover, the data needs to be written in a compatible format. I'm not going to provide support for this feature right now.

- **USB interface:** The USB-to-serial IC used in TNG-5 is the FT232BM from FTDI. Two software drivers are available: a virtual COM port (www.ftdichip.com/FTWinDriver.htm), and a DLL (www.ftdichip.com/FTD2XXDriver.htm). Initially, I intend to use the virtual COM port driver. Eventually, we may want to use the DLL driver. Also, support for USB 2.0 descriptor strings and settings are provided in the form of an on-board EEPROM. Using and programming this EEPROM needs to be investigated.
- **Data Rate:** The interface rate is going to be nominally 128 kbps. The fastest possible rate using optical isolation is about 200 kbps. Without the optical isolation 1 Mbps is possible. These speeds are going to be difficult to sustain (and obtain) in certain applications. Baud rate should be a parameter. I expect to use the on-board configuration switch to select initial baud rate. There should be a baud rate command. On initialization, though, everyone has to agree to start at the same rate.
- **Sample Rate:** Sample rates and data rate are intertwined. The rate that samples are sent can not exceed the data rate. However, it can be much slower. A mechanism for altering sample rate needs to be implemented.
- **Command Mode vs. Streaming Mode:** With TNG-3b we had little choice but to implement a streaming mode as there was no way to send commands. TNG-4 saw the advent of command mode, but even the streaming mode protocol recognized that output and configuration only needed to be sent when changed. Both TNG-3b and TNG-4 used a moderately fast RS-232 connection. Now that the TNG-5 family is targeted a host of operational environments which include sharing bandwidth with other devices on a data bus (USB and Ethernet), it should be better behaved. We've discussed implementing both the command and streaming modes within the same device, selected by switch setting. I think we need to meld the two modes into a pseudo-streaming mode. TNG-5 could be commanded to repeatedly transmit a block of data with a fixed interval between transmissions. This stream could be suspended for certain other commands, and then reinstated. Also, a switch selection could enable a default block transmission after initialization subject to modification by command.

TNG-5 Commands:

TNG-5 commands are ASCII bytes (To send the "9D" command, transmit a single byte equal to that value, not the ASCII characters "9" and "D"). Command byte values are in hexadecimal. The following commands are supported:

90: Read first bank memory byte. Reads any memory location (0-255). The memory location to read is specified in the byte sent immediately following the command. The command returns one byte that is the value of that location.

9D: Return ID/version. This command returns 30 bytes: "TNG-5 V1.0 ©2004 SenSyr, LLC<CR><LF>"

BB: Set interface baud rate. Allows change of baud rate by command. The command expects one subsequent byte that serves as a baud rate divisor for the new baud rate. The formula is: 32 MHz/ (64(X+1)). The default baud rate is determined by configuration switches. The first three configuration switches (SW0-SW2) determine the baud rate at startup. The baud rates corresponding to the values (0–7) of these switches are: 2400, 4800, 9600, 19200, 38400, 57600, 115200, and 125000 (128K).

CC: Get packet count. Returns a two-byte packet count (high, low).

F0: Reset packet number. In block mode each packet sent increments a 16-bit number that can be optionally sent as part of the packet. This command resets the packet number to zero.

FF: Does nothing really. If you send it 3-6 times, then you'll be sure that the next byte will be interpreted as a command byte.

ADC Commands:

A0: Read ADC Channel 0. This command returns 2 bytes: MSB, LSB. The most significant 8 bits are returned in the first byte. The second byte contains the least significant 2 bits left-justified in the byte (the remaining bits = 0). To convert this number to an integer right shift the combined word by 4-bits. Some care must be exercised in using the single channel read commands. There are limits to the time resolution of the samples. The ADC sampling process is independent of the data interface. At 128 kbps you could theoretically obtain 6400 samples per second using a single channel command; however, the channel's data is not updated that fast. The maximum usable sample rate is 1250 Hz.

A1: Read ADC Channel 1.
A2: Read ADC Channel 2.
A3: Read ADC Channel 3.
A4: Read ADC Channel 4.
A5: Read ADC Channel 5.
A6: Read ADC Channel 6.
A7: Read ADC Channel 7.
A8: Read ADC Channel 8.

A9: Read ADC Channel 9.
AA: Read ADC Channel 10.
AB: Read ADC Channel 11.
AC: Read ADC Channel 12.
AD: Read ADC Channel 13.
AE: Read ADC Channel 14.
AF: Read ADC Channel 15.

C0: Read the first N ADC channels where N ranges from 1 to 16. The number of consecutive ADC channels to read is sent in a byte immediately following the command. The total number of bytes returned is dependent on N (N+((N+1)/2)). The most significant bytes are returned first. The least significant nibbles are returned packed into bytes. Odd ADC channel data is in the high nibble. This scheme is common to all the multiple channel commands.

C1: Read ADC channels 0-3.
C2: Read ADC channels 4-7.
C3: Read ADC channels 8-11.
C4: Read ADC channels 12-15.
C8: Read ADC channels 0-7.
CF: Read ADC channels 8-15.

CA: Read all ADC channels. This command returns the data for all 16 ADC channels (24 bytes).

Digital I/O Commands:

BC: Set Port-B configuration. This command takes the next byte received as the Port-B configuration data byte. Each bit of the configuration byte corresponds to a port bit. Set configuration bits to 1 to enable input, 0 for output. This is similar for Port D.

DC: Set Port-D configuration. This command takes the next byte received as the Port-D configuration data byte.

BF: Read Port-B configuration. Returns one byte where bits = 1 are inputs.

DF: Read Port-D configuration. Returns one configuration byte.

Port C cannot be configured. Port C bits are pre-defined. The least significant 3 bits are defined as outputs, and can be written. This has proved useful on at least one occasion when I was implementing a specific SPI protocol.

BD: Write Port-B data. This command takes the next received byte as data to write to Port-B.

CD: Write Port-C data. This command takes the next received byte as data to write to Port-C.

DD: Write Port-D data. This command takes the next received byte as data to write to Port-D.

FB: Read Port-B data. This command sends one byte read from Port-B.

FC: Read Port-C data. This command sends one byte read from Port-C (just to be complete).

FD: Read Port-D data. This command sends one byte read from Port-D.

FA: Read Port B and D together. This command returns 2 bytes—Port-B, and Port-D.

BF: Read Port B configuration data. This command returns the one-byte Port B mask.

DF: Read Port D configuration data. This command returns the one-byte Port D mask.

DB: Write B and D digital I/O ports. This command accepts the next two bytes sent as data for Ports B, and D, respectively.

Alarm Commands:

Each ADC channel can have a high and/or low level alarm condition associated with it. A high-level alarm occurs if the value of the associated ADC channel exceeds the alarm limit. A low-level alarm occurs if the value of the associated ADC channel is below the alarm limit. The alarm values are single-byte values corresponding to the most-significant bits of the ADC value for an analog input channel.

A 16-bit mask is stored for each ADC channel and alarm type. Each bit of the alarm mask corresponds to a bit in Port B and Port D. When the alarm condition occurs, the bits (assumed to be configured as outputs) specified by the nonzero bits in the mask are inverted. That is, if the bit was 0 it is set to 1 and *vice versa*. Once set, alarms need to be reset.

F1: Get High Alarm Status. Returns 2 bytes: ADC alarms 15-8 and ADC alarms 7-0. Each bit=1 indicates an alarm condition in that ADC channel.

F2: Get Low Alarm Status Returns 2 bytes as in F1.

F3: Clear alarm. This command interprets the next byte sent as follows:

7	6	5	4	3	2	1	0
H	L	X	X	D3	D2	D1	D0

Where H = 1 if clearing high alarm.
L = 1 if clearing low alarm.
X = don't care (0).
And D0-D3 = Channel number (0-15).

F4: Alarm Set. The subsequent four bytes are interpreted as follows:

Byte 1:

7	6	5	4	3	2	1	0
H	L	X	X	D3	D2	D1	D0

Where H = 1 if setting high alarm.
L = 1 if setting low alarm (don't set both at once).
X = don't care (0).
And D0-D3 = Channel number (0-15).

Byte 2: High byte of ADC channel limit.
Byte 3: Port B mask.
Byte 4: Port D mask.

F5: Disable Alarm. Turns off the high and/or low alarm for a given ADC channel. Subsequent byte is interpreted as in F3 command. Masks and thresholds are unaffected.

F6: Enable Alarm. Turns on the high and/or low alarm for a given ADC channel. Subsequent byte is interpreted as in F3 command. Masks and thresholds are assumed to have been previously set.

F7: Disable All Alarms. All alarms are disabled. Masks and thresholds are unaffected.

F8: Enable All Alarms. Masks and thresholds are assumed to have been previously set.

RS-232 Commands:

TNG-5 has a secondary RS-232 port whose operation is dependent on how the board is populated. The port can support true RS-232 levels, TTL levels or inverted TTL levels. These operational modes are not software selectable. Two TTL-level I/O lines are associated with this port—GPIO1 and GPIO2. The port has a 32 character input buffer and a 32 character output buffer.

D0: Sets RS-232 port baud rate. The subsequent byte is used as a divisor for the baud rate generator (BRGH=0: 32 MHz/ (64(X+1)); BRGH=1: 32 MHz/(16(X+1))). X is the divisor. The default baud rate is 9600 (33h; BRGH=0).

D1: Configure RS-232 port. The subsequent byte is interpreted as follows:

7	6	5	4	3	2	1	0
X	X	X	X	X	BRGH	GPIO2	GPIO1

Where X = don't care (0), BRGH = 0 for 32 MHz/(64(X+1)) baud rate formula and BRGH =1 for 32 MHz/(16(X+1)) baud rate formula. GPIO1 and GPIO2 are set to 1 to designate the corresponding line to be an input. They are set to 0 to designate that the corresponding line should be an output.

D2: Clear input buffer. All bytes in the input buffer are discarded.

D3: Clear output buffer. All bytes in the output buffer are discarded and the port is reset.

D4: Number of characters in the input buffer. This command returns one byte (0-32) corresponding to the number of characters currently in the input buffer.

D5: Number of characters in the output buffer. This command returns one byte (0-32) corresponding to the number of characters currently in the output buffer.

D8: Write N characters to the RS-232 port. Writes N (0-32) characters to the output buffer of the RS-232 port. The byte subsequent to the command is interpreted as follows:

7	6	5	4	3	2	1	0
GPIO2	GPIO1	D5	D4	D3	D2	D1	D0

Where D0-D5 is the number of bytes to send, and GPIO1 and GPIO2 correspond to the state of the GPIO2 and GPIO1, assuming they are configured as outputs.

Additional 0-N characters are sent corresponding to the number of data bytes. If N=0, then only the state of GPIO1 and GPIO2 are affected.

D9: Read N characters from the RS-232 port. The subsequent data byte is read as N. If N is greater than the number of characters in the buffer, then only those characters are sent. The return data are N+1 bytes where the first byte is as in the D8 command. GPIO1 and GPIO2 are the state of those two data lines assuming they are configured as inputs. D0-D5 corresponds to the number of returned data bytes from the buffer to follow. If N is 0, then only the first byte is returned.

SPI Commands:

9C: SPI Configure. This command redefines all SPI operations and Mezzanine operations until the next SPI configuration byte is received or TNG-5 is power-cycled. This command gets two subsequent bytes.

Byte 1:

7	6	5	4	3	2	1	0
SMP	CKE	---	CKP	--	---	SSPM1	SSPM0

SMP = SSPSTAT:SMP bit for PIC with same meaning.
1 = Input sampled at end of output.
0 = Input sampled in middle of output bit (default).

CKE = SSPSTAT:CKE bit for PIC with same meaning.
1 = Data output on leading edge of clock (default).
0 = Data output on trailing edge of clock.
CKP = SSPCON:CKP bit for PIC with same meaning.
0 = Clock normally low (default).
1 = Clock normally high.

SSPM1 and SSPM0 set the clock rate.
00=FOSC/4 (8MHz, default)
01=FOSC/16 (2MHz).
10=FOSC/64 (500kHz).
11=TMR2/2 Rate (TMR2=[8MHz/(1-256)])

Byte 2: Is the TMR2 value (0-255) or 0 if not being set.

98: SPI 1 write/read. This command requires a subsequent SPI flag byte and 1-6 or 8-256 data bytes. The SPI flag byte is detailed below. If the read bit is set, TNG-5 will return the number of bytes sent.

7	6	5	4	3	2	1	0
R/W	S2	S1	S0	--	D2	D1	D0

R/W: This bit = 0 when data is output only.
This bit = 1 when reading data.

S2-S0: These bits specify which SPI enable line to use.
000 = Port C bit 1
001 = Port B bit 7

010 = Port B bit 6
011 = Port B bit 5
100 = Port B bit 4
101 = Port B bit 3
110 = Port B bit 2
111 = Port B bit 1

D2-D0: These bits specify the number of data bytes to send/receive (1-6). When D2-D0 equals 7, the next byte sent will be interpreted as the number of subsequent bytes in the message. If this next byte is zero, no bytes are sent or received.

99: SPI 2 write/read. This command requires a subsequent SPI flag byte and 1-6 or 8-256 data bytes. The SPI flag byte is detailed below. If the read bit is set, TNG-5 will return the number of bytes sent.

7	6	5	4	3	2	1	0
R/W	S2	S1	S0	--	D2	D1	D0

R/W: This bit = 0 when data is output only.
This bit = 1 when reading data.

S2-S0: These bits specify which SPI enable line to use.
000 = Port C bit 2
001 = Port D bit 7
010 = Port D bit 6
011 = Port D bit 5
100 = Port D bit 4
101 = Port D bit 3
110 = Port D bit 2
111 = Port D bit 1

D2-D0: These bits specify the number of data bytes to send/receive (1-6). When D2-D0 equals 7, the next byte sent will be interpreted as the number of subsequent bytes in the message. If this next byte is zero, no bytes are sent or received.

Mezzanine commands:

There are two mezzanine ports. Each port has two possible select lines and two general-purpose lines.

9A: Mezzanine 1 write/read. This command requires a subsequent flag byte and 1-6 or 8-256 data bytes. The Mezzanine flag byte is detailed below. If the read bit is set, TNG-5 will return the number of bytes sent.

7	6	5	4	3	2	1	0
R/W	--	S1	S0	--	D2	D1	D0

R/W: This bit = 0 when data is output only.
This bit = 1 when reading data.

S0-S1: These bits specify which SPI enable line to use.
00 = Use mezzanine select line 1
01 = Use mezzanine select line 2
10 = Use general purpose I/O line 1
11 = Use general purpose I/O line 2

D2-D0: These bits specify the number of data bytes to send/receive (1-6). When D2-D0 equals 7, the next byte sent will be interpreted as the number of subsequent bytes in the message. If this next byte is zero, no bytes are sent or received.

9B: Mezzanine 2 write/read. This command requires a subsequent flag byte and 1-6 or 8-256 data bytes. The Mezzanine flag byte is detailed below. If the read bit is set, TNG-5 will return the number of bytes sent.

7	6	5	4	3	2	1	0
R/W	--	S1	S0	--	D2	D1	D0

R/W: This bit = 0 when data is output only.
This bit = 1 when reading data.

S0-S1: These bits specify which SPI enable line to use.
00 = Use mezzanine select line 1
01 = Use mezzanine select line 2
10 = Use general purpose I/O line 1
11 = Use general purpose I/O line 2

D2-D0: These bits specify the number of data bytes to send/receive (1-6). When D2-D0 equals 7, the next byte sent will be interpreted as the number of subsequent bytes in the message. If this next byte is zero, no bytes are sent or received.

E0: Mezzanine status. Returns the status of both sets of mezzanine I/O lines assuming that the lines have been configured as inputs as follows:

7	6	5	4	3	2	1	0
--	--	M2.1	M2.0	--	--	M1.1	M1.0

ED: Write mezzanine I/O bits. Takes subsequent byte as state of mezzanine I/O bits assuming that the lines have been configured as outputs. Refer to E0 command for data byte structure.

EC: Configure mezzanine ports. Takes one subsequent byte to set the general-purpose I/O lines as inputs (bit=1) or outputs (bit=0). Refer to E0 command for data byte structure.

Block data commands:

In the interest of maintaining some sort of streaming mode, I am creating the block data commands. The SW7 configuration switches can be used to set this mode as the default (125 blocks/second).

A data block (packet) consists of a start byte (alternating 55/AA) followed by a flag byte and one or more data bytes. The data bytes sent are defined by settable parameters. ADC data is sent packed.

B0: Block send off. Turns off block data transmission.

B1: Block send on. Turns on block data transmission as per B4, B8, and B9.

B4: Set sample interval. Two subsequent bytes (HB,LB) set a divisor that determines an interval between transmissions. The minimum sample interval is very much affected by the current baud rate. Minimum interval = 1/[(baud rate/10)/packet size)]. Each count corresponds to one millisecond.

B8: Set number of ADC Channels to send. Subsequent byte is number of channels starting at 0.

B9: Set DIO mask. One subsequent data byte determines what digital data is sent, as follows:

7	6	5	4	3	2	1	0
--	--	--	--	--	PN	PD	PB

Where PN = 1 to send packet number.
PD = 1 to send Port D data.
And PB = 1 to send Port B data.

Fully tricked-out packet:

		7	6	5	4	3	2	1	0
Byte 1	Separator Byte:	55 or AA							
Byte 2	Flag Byte	**PN**	**PD**	**PB**	**ADC4**	**ADC3**	**ADC2**	**ADC1**	**ADC0**
Byte 3	A0 MSB	**A0**							
Byte 4	A1 MSB	**A1**							
Byte 5	A2 MSB	**A2**							
Byte 6	A3 MSB	**A3**							
Byte 7	A4 MSB	**A4**							
Byte 8	A5 MSB	**A5**							
Byte 9	A6MSB	**A6**							
Byte 10	A7MSB	**A7**							
Byte 11	A8 MSB	**A8**							
Byte 12	A9 MSB	**A9**							
Byte 13	A10 MSB	**A10**							
Byte 14	A11 MSB	**A11**							
Byte 15	A12 MSB	**A12**							
Byte 16	A13 MSB	**A13**							
Byte 17	A14 MSB	**A14**							
Byte 18	A15 MSB	**A15**							
Byte 19	A0/A1 LSB	**A1**				**A0 (xx00)**			
Byte 20	A2/A3 LSB	**A3**				**A2**			
Byte 21	A4/A5 LSB	**A5**				**A4**			
Byte 22	A6/A7 LSB	**A7**				**A6**			
Byte 23	A8/A9 LSB	**A9**				**A8**			
Byte 24	A10/A11 LSB	**A11**				**A10**			
Byte 25	A12/A13 LSB	**A13**				**A12**			
Byte 26	A14/A15 LSB	**A15**				**A14**			
Byte 27	Port B	Port B							
Byte 28	Port D	Port D							
Byte 29	Packet # (MSB)	Packet # (MSB)							
Byte 30	Packet # (LSB)	Packet # (LSB)							

Flag Byte: PN = 1 if packet number is included**.**
PD = 1 if Port D data sent.
PB = 1 if Port B data sent.
and ADC0-4 = number of ADC channels sent starting with ADC 0.

Command Table:

90	Send RAM byte	B0	Block Send Off	D0	RS-232 Baud	F0	Clear Packet #
91		B1	Block Send On	D1	RS-232 Config.	F1	Get High Alarm
92		B2		D2	Clear Input Buf.	F2	Get Low Alarm
93		B3		D3	Clear Out. Buf.	F3	Clear Alarms
94		B4	Set Sample Int.	D4	Inp. Buf. Char.	F4	Alarm Set
95		B5		D5	Out.Buf. Char.	F5	Disable Alarm
96		B6		D6		F6	Enable Alarm
97		B7		D7		F7	All Alarms Off
98	SPI 1	B8	Set ADC Mask	D8	RS-232 Write N	F8	All Alarms On
99	SPI 2	B9	Set DIO Mask	D9	RS-232 Read N	F9	
9A	Mezzanine 1	BA		DA		FA	Read B and D
9B	Mezzanine 2	BB	Int. Baud Rate	DB	Write B and D	FB	Read Port B
9C	SPI Configure	BC	Config. Port B	DC	Config. Port D	FC	Read Port C
9D	ID command	BD	Write Port B	DD	Write Port D	FD	Read Port D
9E		BE		DE		FE	
9F		BF	Get Port B Conf.	DF	Get Port D Conf.	FF	Sync
A0	ADC Ch. 0	C0	ADC Ch. 0-N	E0	Mezz. Status		
A1	ADC Ch. 1	C1	ADC Ch, 0-3	E1			
A2	ADC Ch. 2	C2	ADC Ch. 4-7	E2			
A3	ADC Ch. 3	C3	ADC Ch. 8-11	E3			
A4	ADC Ch. 4	C4	ADC Ch. 12-15	E4			
A5	ADC Ch. 5	C5		E5			
A6	ADC Ch. 6	C6		E6			
A7	ADC Ch. 7	C7		E7			
A8	ADC Ch. 8	C8	ADC Ch. 0-7	E8			
A9	ADC Ch. 9	C9		E9			
AA	ADC Ch. 10	CA		EA			
AB	ADC Ch. 11	CB		EB			
AC	ADC Ch. 12	CC	Get Packet #	EC	Conf. Mezz.		
AD	ADC Ch. 13	CD	Write Port C	ED	Write Mezz. I/O		
AE	ADC Ch. 14	CE		EE			
AF	ADC Ch. 15	CF	ADC Ch. 8-15	EF			

Configuration Switch:

Switch	Definition
SW0	Default Baud 0
SW1	Default Baud 1
SW2	Default Baud 2
SW3	undefined
SW4	undefined
SW5	undefined
SW6	undefined
SW7	Send Blocks on Start up