THE ISAAC TECHNICAL REFERENCE MANUAL VERSION 1.0

The ISaAC Technical Reference Manual

1.0 Introduction

Welcome to ISaAC, a small form factor programmable 32-bit Experimenter! The ISaAC or, **I**deal **S**ystem **a**nd **A**pplication **C**ircuit, can be used as either a "Plug and Play" add-on board for the Raspberry Pi (Raspi) or as a standalone Arduino compatible 32-bit Microcontroller Experimenter. In either case, ISaAC is designed for rapid application development and ease of use. In this manual explain configuring and using ISaAC "Plug and Play" capability with the Raspi, and for those who do not use Raspi how to configure and operate ISaAC standalone.

Although challenging problems can be handled with the Raspi alone, adding ISaAC allows the Raspi to work with greater capabilities, in near real time, across a larger number of applications (see figure 1). No special programming is required, as the ISaAC comes pre-programmed for the Raspi with a high level Application Programming Interface (API). The API provides the Raspi new functionality: Raspi power control and external system power +5VDC on/off, an on board EEPROM for read/write of both data and ISaAC API command scripts, a 100 year Real Time clock calendar (RTCC) with multiple programmable alarm capabilities, 4 analog digital converter (ADC) conversion inputs, a programmable digital analog out (DAC) signal output, 13 digital I/O, and dual pulse width modulation (PWM) outputs for servos and sound. In this manual, we will cover how to integrate ISaAC with the Raspi, and then work through API examples using Linux minicom terminal program. Other API methods with Raspi besides minicom are to exercise the API with BASH (BASH = Bourne Again Shell) or Python. BASH is Linux shell programming to execute API scripts from the command line. Python programming with ISaAC, and using the Python libraries, can be used to work more challenging and fun applications.

In a standalone configuration (not integrated with Raspi, see figure 2) you can exercise the API through PC running TeraTerm or a similar terminal emulation capability through a COM port. You need to wire ISaAC serial communication pins to a serial to USB interface. We will show you how. Because the API is

standard across all configurations, the API exercises are similar for both standalone and Raspi configurations.

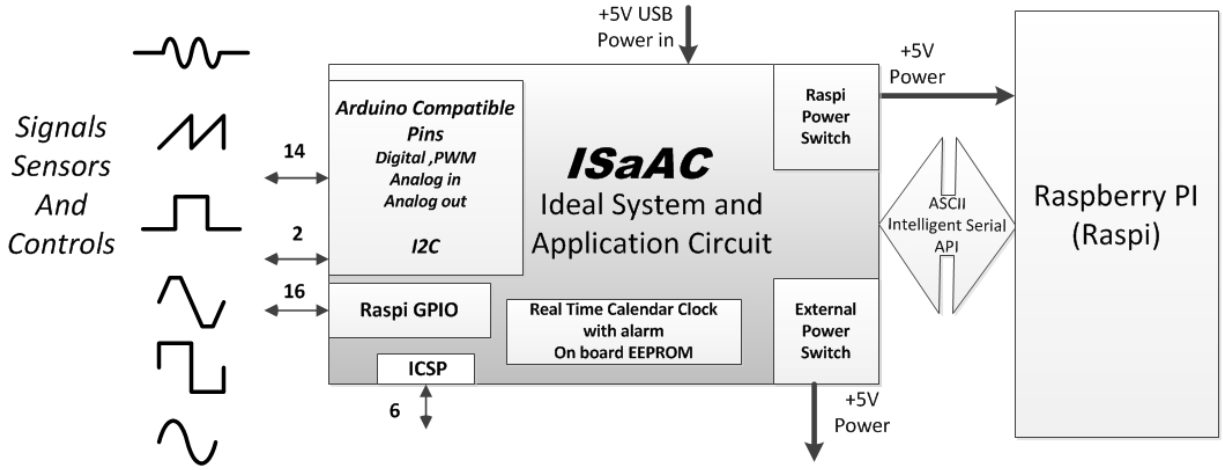


Figure 1: ISaAC with Raspi Block Diagram

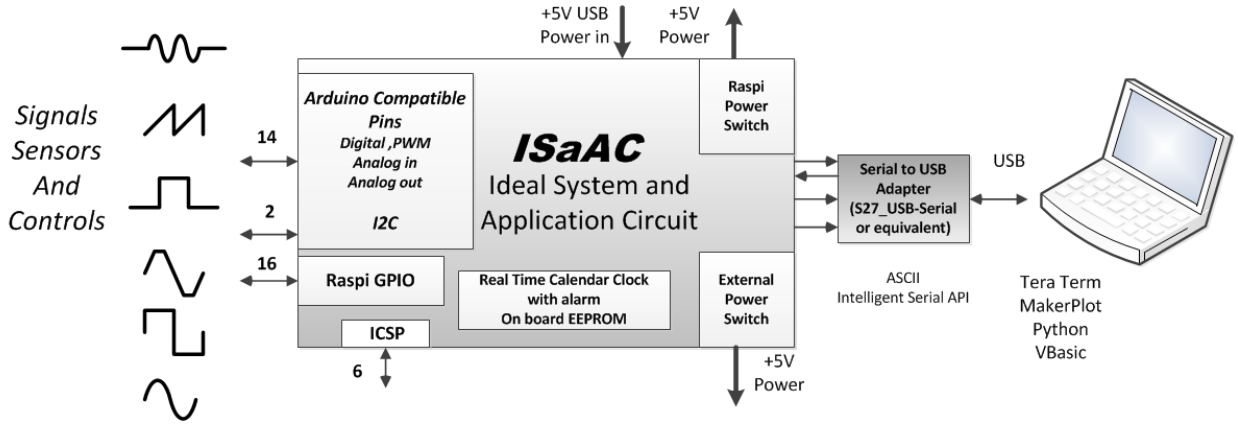


Figure 2: ISaAC Standalone with PC

The ISaAC board comes fully assembled and tested with the API. The ISaAC is shown mounted onto the Raspi in figure 3.

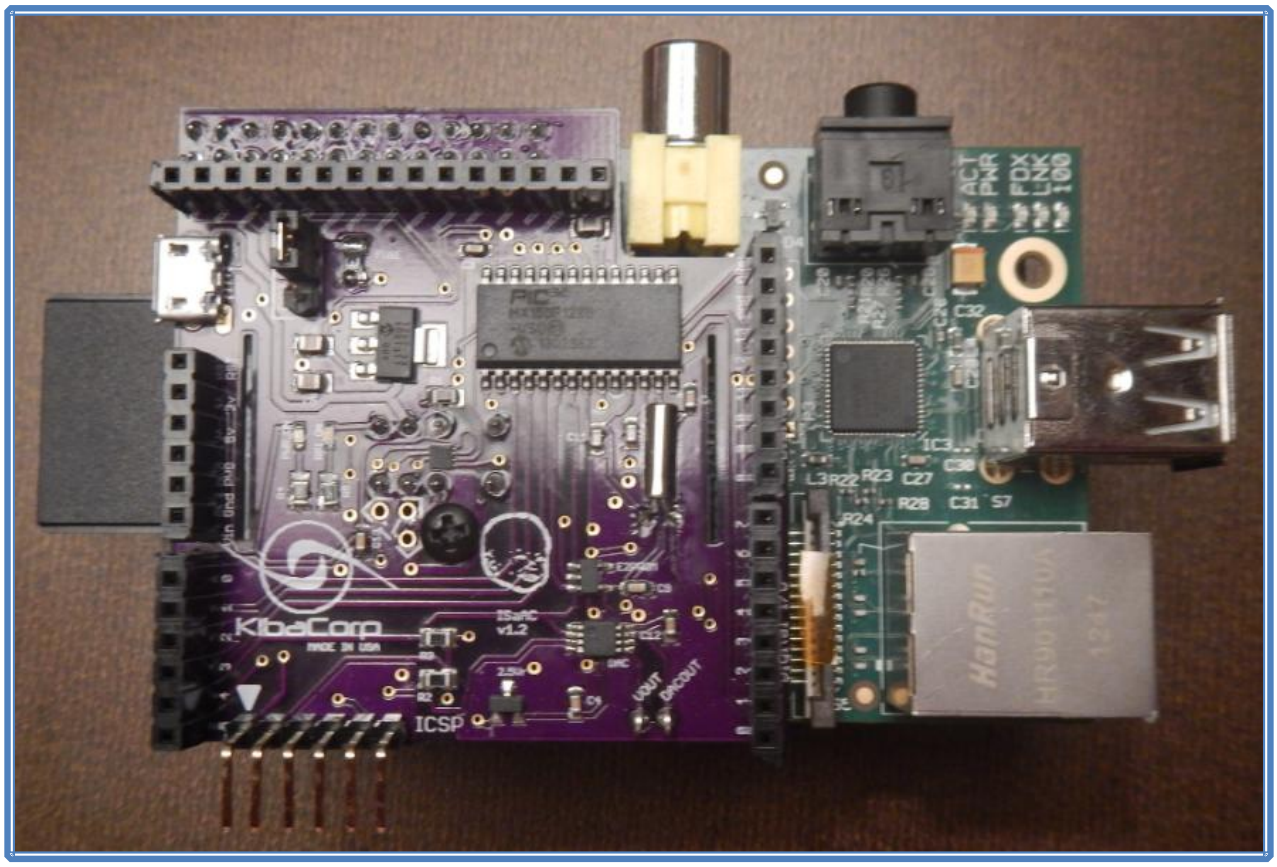


Figure 3: ISaAC and Raspi

ISaAC uses a clock controllable 40 MHz 32-bit PIC32 Microcontroller. It is a fabricated as a small SMT card that attaches to the Raspi GPIO header and can be optionally reinforced with mounting hardware. ISaAC supports an accessible Raspi GPIO pin connector as well as an add-on Arduino-style expansion interfaces (see figures 4). Additionally, ISaAC provides unobstructed physical access to the Raspi CSI cable ports through on-board slot cut outs.

ISaAC Accessible interfaces

- (9) Digital I/O programmable
  - DO,D1,D2,D8,D9,D10,D11 or 0,1,2,8,9,10.11,12,13
  - D9 and D10 can be configured as PWM
  - API uses pin numbers 00, 01,02,03,04,05,06,07,08,09,10,11,12,13
- (4) Analog In (10 bit resolution) or Digital I/O programmable

  - A0,A1,A2,A3
  - API uses pin numbers 14,15,16,17

- (1) Analog Out (DAC) (10 bit resolution)
  - VOUT (buffered), DACOUT (un-buffered)
- An accessible I2C Bus: pins A4 (data) A5 (clock)
  - Bus contains EEPROM and DAC

- Raspi GPIO pin breakout
- Micro USB for +5V power
- Raspi Power Select Jumper for switchable control of power to Raspi
- Separate external +5V power switch for external circuits.
- An ICSP (In Circuit Serial Programmer) interface
  - Custom flash programming and debug

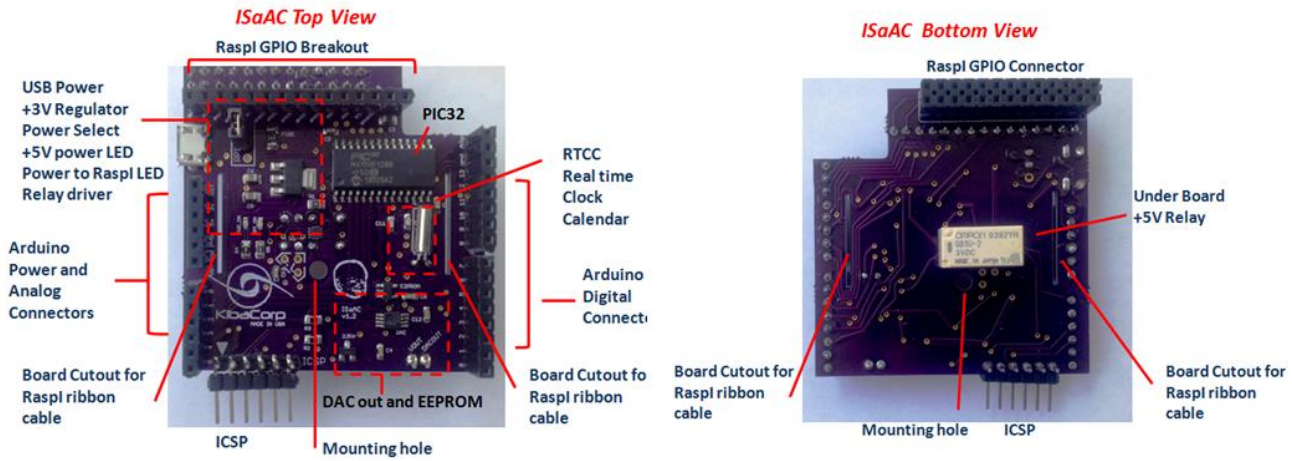


Figure 4: ISaAC board

2.0 API Overview

ISaAC uses the Raspi GPIO serial port (19200 8N1) for API communications. API commands are ASCII upper case characters and are echoed back for easy debugging. Command verification from ISaAC is

simply 'A' (acknowledge) or 'N' (not acknowledge). There are 19 unique API commands. Detailed descriptions are contained in Appendix A.

API Commands	Function
?	query state of digital input pin
A	set pin to analog input/read
CR	Read RTCC clock
CS	Set RTCC clock
D	output to DAC
ER	single EEPROM read
EW	single EEPROM write
H	set output pin high
I	Configure pin for digital input
L	set output pin low
M	Help Menu
O	Configure pin for digital output
P	set pin to PWM output
Q	build script
T	Set alarm
U	PI power cycle on alarm
V	view EEPROM
X	execute script
Z	Z – precision pulse

Figure 3: API

3.0 Setting up ISaAC with the Pi

The first thing to do is prepare the Raspi for the ISaAC board. We have to reconfigure the Raspi serial port (ttyAMA0) as it is set by default to be an administrative terminal. There are two ways to do this:

- Method 1: Edit the /boot/cmdline.txt file.
  - Remove the following: "console=ttyAMA0, 115200 kgdboc=ttyAMA0, 115200"
  - Save and close the file

- Method 2: Insert the Raspi SD card into your computer
    - edit cmdline.txt file as above

The next step will prevent the Raspi from outputting boot information over the serial line. If this is left on, the serial buffer can get overwhelmed with this data and it might be difficult to successfully send commands to the ISaAC initially.

- Open /etc/inittab for editing
- Place a hash (#) in front of the line with: TO:23:respawn:/sbin/getty –L ttyAMA0115200 vt100

At this point, initiate shutdown of the Raspi using the command "sudo shutdown now -h". After shutdown, attach the ISaAC board to the Raspi. Place the jumper onto the "RPI PWR" spot on the ISaAC and attach the micro USB power cable to the ISaAC. The ISaAC uses a red led for +3.3V power indication and a blue led for relay on/off indication. Both boards should now power up and these leds should be on. Ensure your Raspi is functioning normally via your method of choice (i.e. SSH or HDMI port).

One final configuration step is required before we proceed. Open minicom "sudo minicom -s". You need to request superuser permissions to make changes to the configuration and to access the onboard serial port. If you don't have minicom installed, simply type "sudo apt-get install minicom".

Once minicom is open, chose the "Serial port setup" menu item, and the press "A" to change "Serial Device" to "/dev/ttyAMA0". Press "E" to adjust the serial port settings. Scroll through the menu with "A" and "B" until you find a baud rate of 19200. Ensure the "Current:" line at the top reads "19200 8N1". Press <Enter> to leave this screen then press <Enter> again to get to the main menu. Choose the menu item "Save setup as dfl" to make this the default setting, now choose the menu item "Exit", which will dump you into the active minicom terminal. Now it's time to have some fun.

4.0 Running ISaAC with Pi

A few precautions before proceeding:

- If you are bread boarding your circuits with a separate power supply, make sure to link the power supply ground and ISaAC ground pins. For most simple experiments, the ISaAC should be able to supply power to the breadboard.
- For routine purposes use only the ISaAC board to supply power to the Raspi. If independent cables are used to power the Raspi, make sure the "RPI PWR" jumper is removed and both boards use the same power source. Failure to do so risks permanent damage to both boards.

5.0 Configuring ISaAC standalone

5.1 Setup Equipment Requirements

1. Acroname S27-USB-SERIAL- adapter for ISaAC API communication with PC USB using TeraTerm Software Emulator
2. USB A-to-mini-B extension cable for interfacing PC to a UART S27 device
3. (4) jumpers to connect S27-USB-SERIAL pins to ISaAC
4. ISaAC board with Micro USB to standard type A USB to supply power to ISaAC
5. TeraTerm Terminal Emulator (see Appendix C software installation instructions)

Figure 4: Acroname S27-USB-SERIAL

5.2 Powering up ISaAC

The ISaAC receives +5VDC from its micro USB and converts this to +3.3VDC for on board use. Any USB port can work with ISaAC as it requires less that 100 ma to operate. Plug in micro USB cable in to ISaAC and the other side into your PC. Both the Red (“power on” indicator) and Blue LED (“relay on” indicator) should be turned on.

5.3 Set up ISaAC Standalone configuration

Once TeraTerm software has been correctly installed (see Appendix C for instructions) and software driver for S27-USB Serial interface is installed (again see Appendix C for details), you ready to begin to exercise the ISaAC using the API contained in the Flash code.

1. Connect S27-USB-Serial interface to PC USB using a Mini-B to A cable.
2. Bring up TeraTerm and select interface for serial port identified for S27-USB- Serial Interface. Configure for 192008N1. Go into terminal mode. Use CAPS LOCK on when communicating with ISaAC.
3. Connect four wires from S27-USB-Serial Interface as shown. See figure figures 7, 8, 9 for descriptions and pictures of setups.
    a. RX Arduino Interface pin 6 to S27-USB-Serial SRX (data out)
    b. TX pin of 10 GPIO to S27-USB-Serial STX (data in)
    c. +3VDC Arduino to S27-USB-Serial VCC
    d. GND Arduino to S27-USB-Serial GND

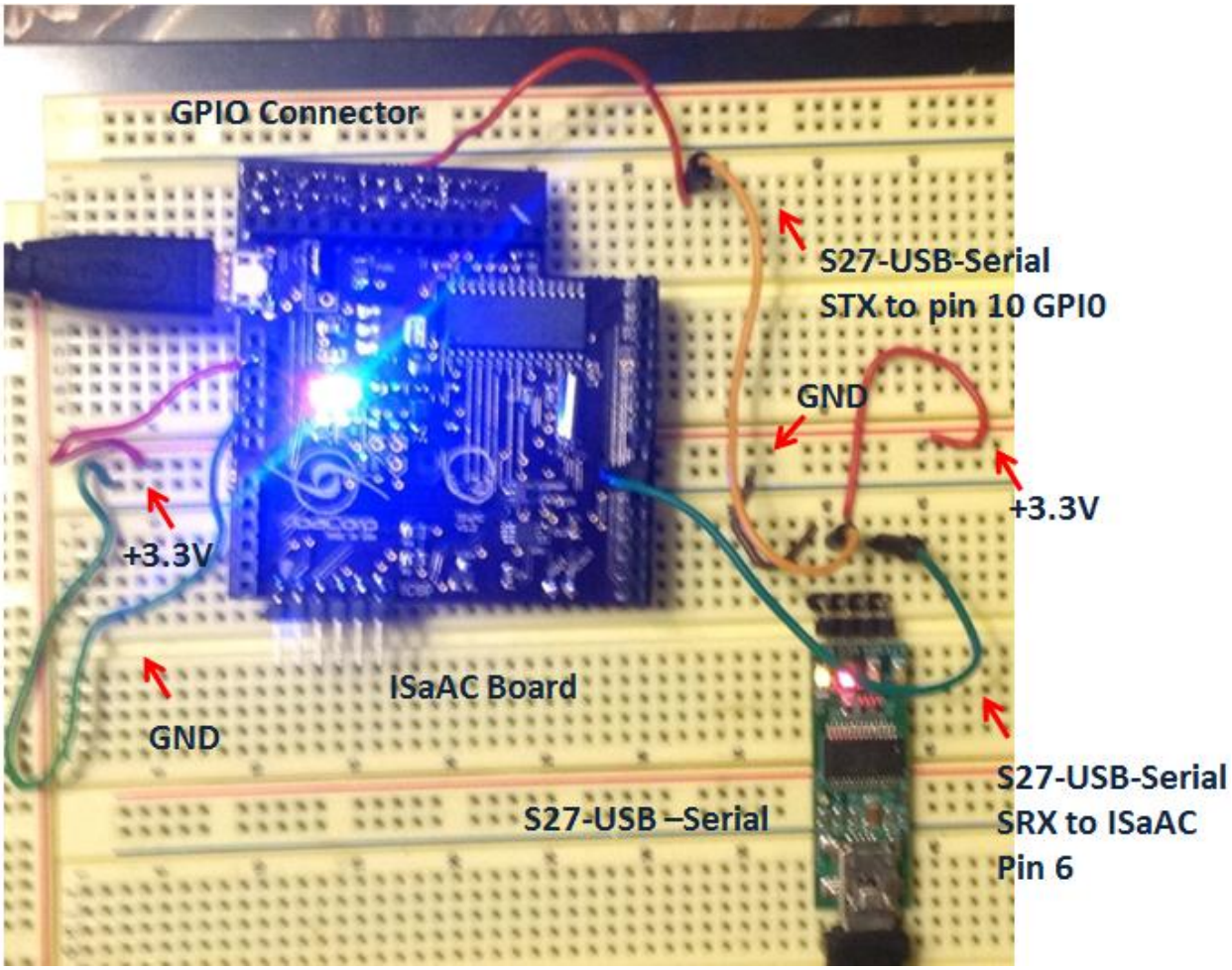


Figure 7: Prototype Setup

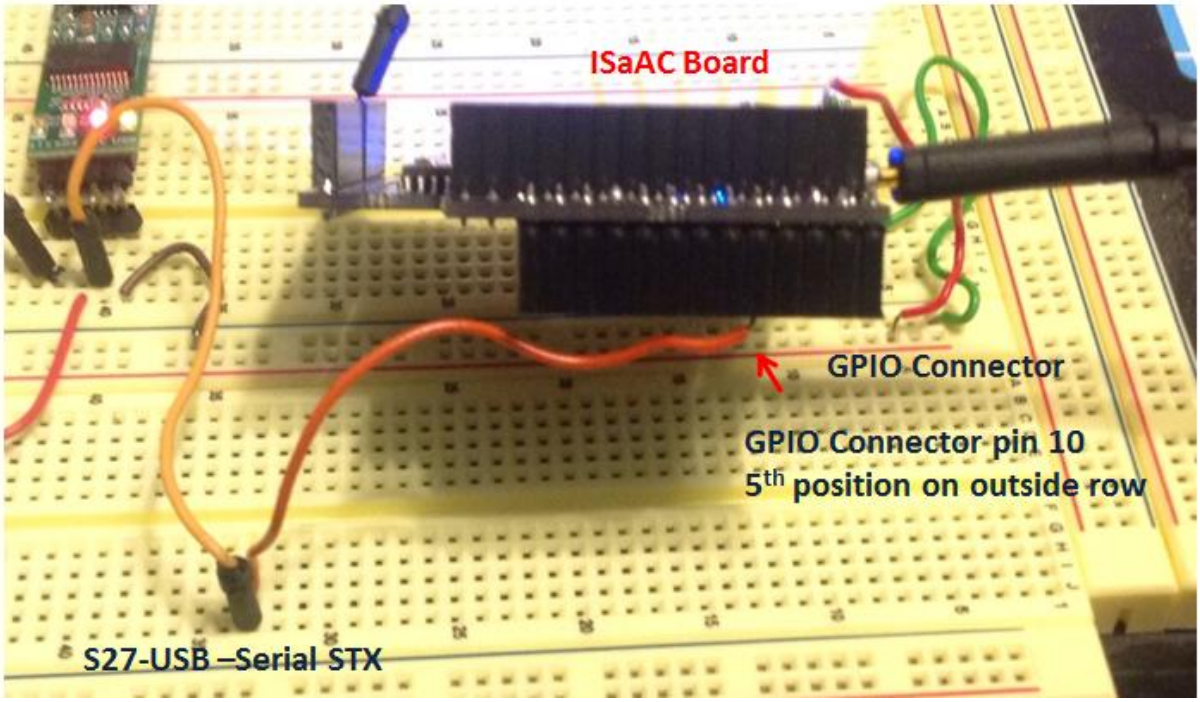


Figure 8: Prototype Continued

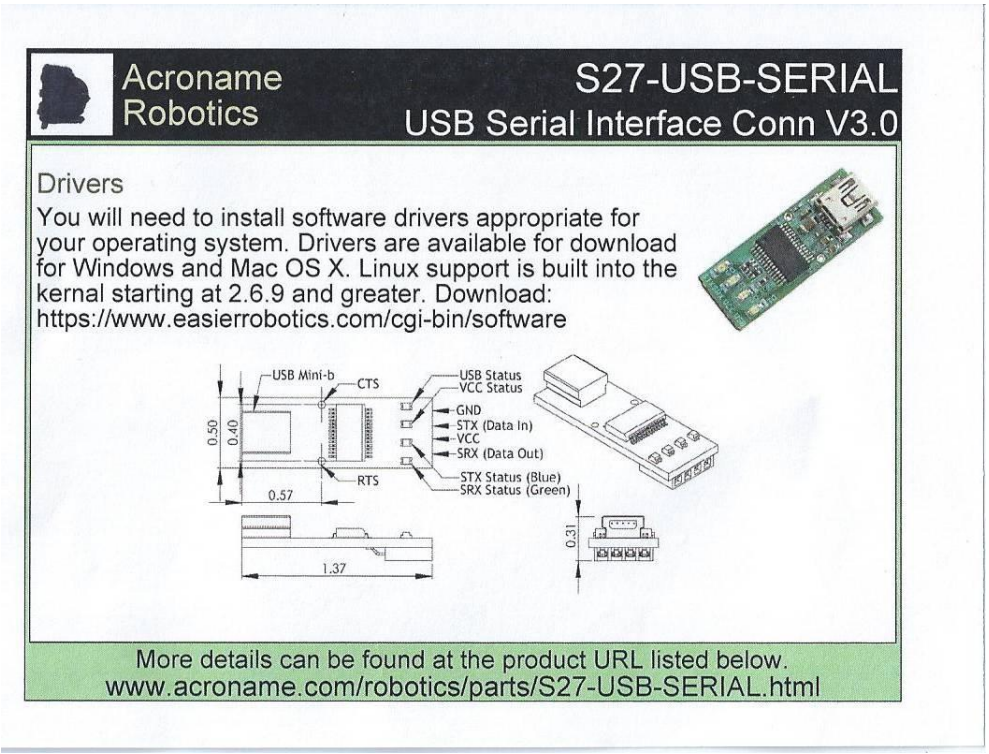


Figure 9: USB to UART adapter

4. Connect power via USB to ISaAC using micro USB cable as described above. You are now ready to conduct the next series of API tests using ISaAC API.

5. Make sure CAP ON is locked. Type character ‘M’ followed by return into TeraTerm active window. ‘M’ stands for MENU; you should something similar to the following response (figure 10). This confirms communications is working between ISaAC and PC TeraTerm. If unsuccessful see debugging detailed suggestions below:

    a. Check hardware connection between ISaAC and S27-USB. Try a loop back by shorting RX/TX on S27-USB-SERIAL to check

connectivity. In this configuration whatever characters you type into TeraTerm active window should echo back.

b. Check TeraTerm serial port setting to confirm 192008N1 and correct COM port is used for ISaAC

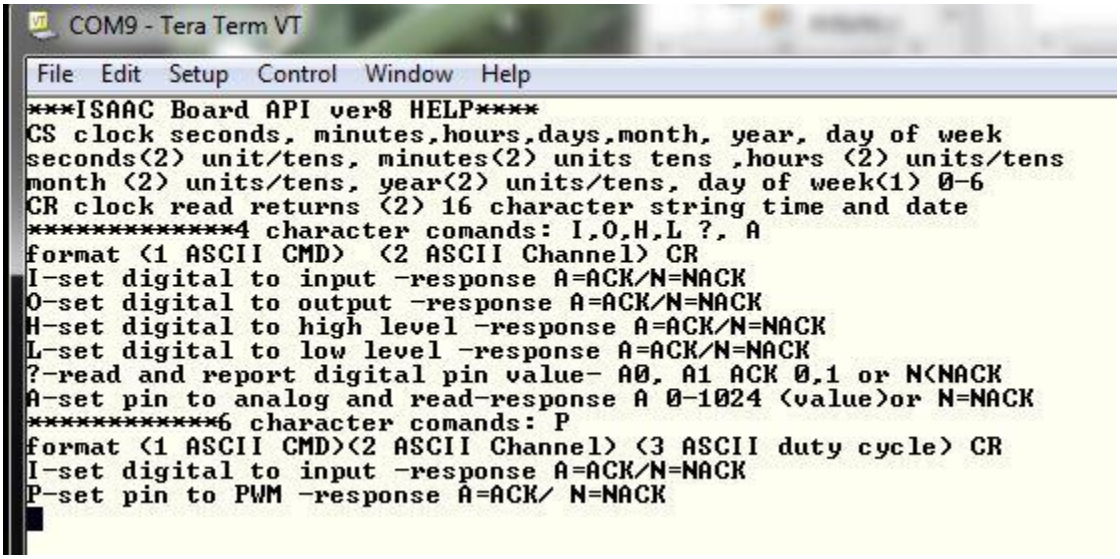


Figure 10: ISaAC 'M' Menu Response

6.0 ISaAC API Examples

These API examples are shown using Raspi configuration with minicom, but apply ISaAC standalone.

6.1 HELLO WORLD

This little board provides a lot of additional features and we will step through basic examples for each. First, we have the classic “HELLO WORLD” example. Plug in an LED, using the appropriate resistor to any

of peripheral pins on the ISaAC (D0-D13 or A0-A3). Refer to the circuit diagram below.

Open the minicom terminal as a superuser.

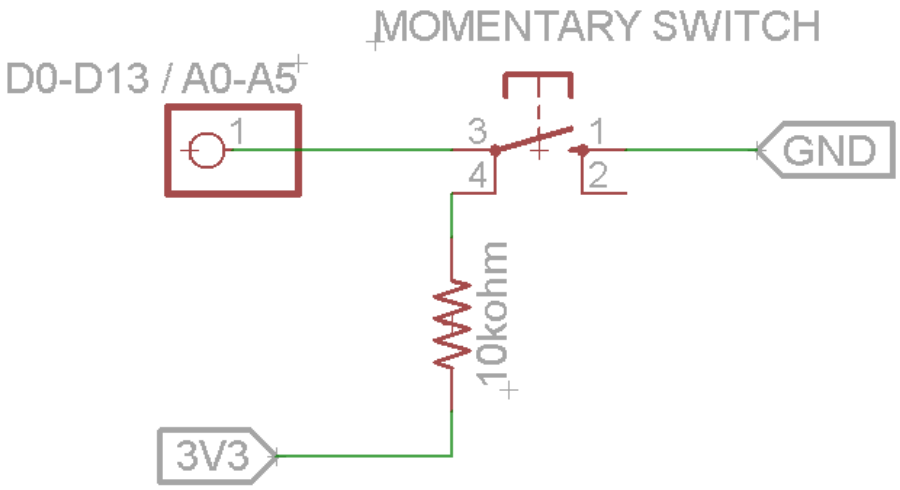
Let's assume we have connected to pin 1 on our ISaAC board. First, we need to set the pin to an output by typing O01 into the terminal; you should receive an "A" as a response. Now type H01, the light should turn on now if it hasn't already. Type L01 and the light will turn off. We have set pin 1 to an output with the first command, then we set that output to high (on), and then we set it to low (off).

These pins can assume 3 states, high-output, low-output, or input; otherwise known as tri-state general purpose input output (GPIO). All of the pins on the ISaAC can be commanded in this way. A few of the pins have more special functions as well, which we will cover later.

- Set pin to output – "O" then 2 digits for pin
- Set pin to high – "H" then 2 digits for pin
- Set pin to low – "L" then 2 digits for pin

6.2 USER INPUT

Let's work another example using another pin as an input. With a momentary push button and wire up your breadboard according to the following circuit.

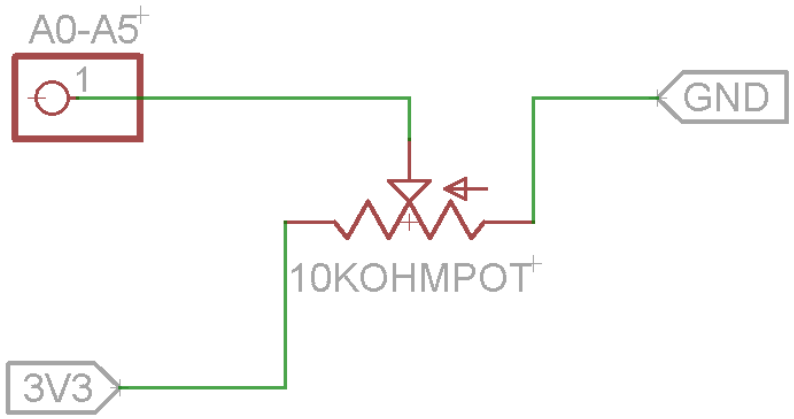


Setup pin 2, as an input, using the API commands I02. Now we can query the state of the pin using the? 02 command. If the switch is not pressed your will get a response of A1, meaning "ACKNOWLEDGE" and "1" for high. If the switch is pressed you will get a response of A0. As in the previous example this functionality works on all of the ISaAC GPIO pins.

- Set pin to input – "I" two digits for pin
- Read input state – "?"two digits for pin

6.3 ANALOG MEASUREMENT

Measuring digital states is very useful, but having the ability to measure analog signals adds another dimension of utility. The ISaAC has 4 pins that have 10-bit analog digital conversion. These pins are labeled A0 to A3. An example circuit can be created with a thumb potentiometer following the schematic below.



Pins A0 to A3 are designated as pins 14, 15, 16, and 17 in the API. We will use pin A0 for this example. Set the pin to an input using the command I14. Next, set the pin to an analog state using the command A14. This will set the pin and return the current value as a 10-bit integer. Play around with the potentiometer and send the A14 command again to get a feel for the different readings. You should see a range from 0 to 1023. This function is limited to the pins listed above.

- Set analog pin to input – "I" two digits for pin (14 to 17)
- Set pin to analog mode, read state – "A" two digits for pin (14 to 17)

6.4 PULSE WIDTH MODULATION

ISaAC also provides a pair of pins (9 and 10) that can provide pulse width modulation (PWM). This signal can be provided to power circuits to drive motors, high powered LEDs, speakers, etc. it can also be used directly to power a low-power LED. PWM comes in two flavors with the ISaAC:

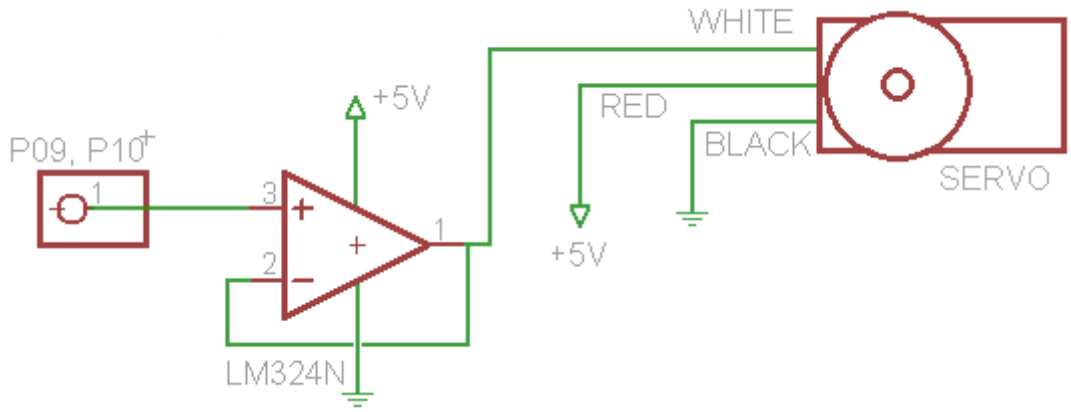
- 100Hz fixed-frequency with programmable duty cycle
- Variable frequency (50Hz to 20KHz) with programmable duty cycle

Go back to the circuit we wired up for the "HELLO WORLD" experiment. Move your pin connection from pin 1 to pin 10. Set pin 10 to an output ("O10"). Choose an 8-bit integer duty cycle for PWM (0 = 0%, 255 = 100%). Send the command P10127 for a half lit LED, P10000 to turn the LED off, and P10255 to turn the LED all the way on.

- Set pin to output – "O" either pin 09 or 10
- Set 100Hz duty cycle – "P" integer from 000 to 255

Running PWM at 100Hz is great for many applications; however some peripherals may require different frequencies, including motors, servos, and speakers. Standard servos usually run at 50Hz. Sending a

pulse duration between 1ms and 2ms will cause the servo to rotate fully from one direction to the other (refer to your servo's documentation). You will need an op-amp chip to drive the servo, I'm using the TI LM324N quad op-amp (if you are short this part or the servo for that matter, check the Nuts and Volts sponsors storefronts).



Notice we are using the 5V power from the ISaAC this time to power the servo. No special level shifters are required in this case because the comparator is supplying power to the servo, always keep in mind some pins on the ISaAC are not 5V tolerant. To move the servo using a 50Hz frequencies with pin 9 send the following command: Z09000500000001500. This is pin 09, a frequency of 00050Hz, a delay of 0000ms, and a pulse of 001500 micro seconds. To move the servos through its range of motion try changing the last 4 digits to 1200 and 1800. Some servos are a different, try modifying the last 4 digits to determine the range of your servo. One additional note, the frequency chosen is shared between the two PWM pins, so changes to one will affect the other.

- Set pin to output – "O" either pin 09 or 10
- Send precision pulse command:
  "Z" (2 digits pin) (5 digits frequency Hz) (4 digits delay in ms) (6 digits pulse width in µs)

6.5 ANALOG SIGNAL OUTPUTGENERATION

The ISaAC is outfitted with a 10-bit digital to analog (DAC) signal generation chip. This is connected to two special pins on the board labeled DACOUT and VOUT. Generally, the VOUT pin should be used as it provides an amplified output. Using the DAC is simple; the command is D followed by a 10-bit integer (0000 to 1023). This will create a signal of 2.5V divided by the value you provide, 1023 will be 2.5V and 0000 will be 0.0V. This is a distinct function from PWM as it provides a steady voltage at the given value instead of 3.3V pulses at the requested duty cycle. You can hook up our "HELLO WORLD" circuit to the DAC, but only about a settings from 750 to 1023 will actually illuminate a high-efficiency LED.

6.6 REALTIME CLOCK READ / WRITE

Another cool feature is the real time clock calendar. The Raspi relies on a network connection for time information. If you have a deep space exploration project, you can rely on ISaAC for accurate time over 100 years from now. First the clock must be set using the CS command. The format of the command is CS(2 digits for year)(2 digits for day)(2 digits for month)(2 digits for seconds)(2 digits for minutes)(2 digits for hours)(1 digit for day of week 0=Sunday). Reading the clock is much simpler, just send the CR command and you receive the current time and date as a 16 character string for current time including day of the week, hours, minutes, seconds, (i.e. "Fri 13:11:03 "), and then on a separate line, a 16 character string for current date to include year, month, day of month and year (i.e. "JUL 05, 2013"). If the power is removed, the date information will be used. For a low power application using a battery pack, use the ISaAC to keep time and have the Raspi sleep until the horsepower is needed. In a subsequent article we show you how to set the date on the Raspi using the "date --set" command, once you recover the time information from the ISaAC.

6.7 REALTIME CLOCK ALARM

Having the real-time is great for time stamping collected data, but ISaAC has an additional alarm feature. This allows you cycle power on the Raspi or simply receive a text alarm at a given time in the future at a specified rate. Setting the alarm will only work after the clock has been set using the CS command. Note, that power to the Raspi can be umpired manually and then the pentagonal breakout on the ISaAC can be used to hardwire a separate power toggling function. To implement the alarm function, type T, then a number for the rate, and then a number for the mode character.

Time frequency	Rate Character
Off	0
1 second	1
10 seconds	2
1 minute	3
10 minutes	4
1 hour	5
1 day	6
1 week	7
1 month	8
1 year	9

Mode	Mode Character
One Time Trigger	1
One Time Trigger / Power Cycle	2
Repeated Trigger	3

For example, a command of T33, would issue an alarm every minute. You would see a string returned of ALARM-M plus the time/date stamp. Issuing a T32 command would cycle power on the Raspi every minute; probably not the best idea. As stated above in this case an alternate device could be wired to the breakout for this sort of "high" frequency triggering. Another command is available for cycling the

power to the Raspi. The U command is issued with a time delay and a time to power back up. For example U99997, waits 9999ms to shut down the Raspi and then will power it up one week later. The first four digits are the delay in ms before shutting down the Raspi and the last digit follows the time frequency convention above for when to power back on.

6.8 EEPROM READ/WRITE/EXECUTE

The ISaAC also sports a 128 Kb EEPROM on board. This allows the user to store API scripts up to 32 ASCII characters or data in the onboard nonvolatile memory. The scripting is composed in byte-wise write commands. Send EW then 000 to 127 for the memory address and 000 to 255 for the ASCII character code to write. For example to write the command A14, send the commands EW000065 (address 000, character A), EW001049 (address 001, character 1), EW002052 (address 2, character 4). You can find an ASCII table at http://www.asciitable.com. Use the numbers from the DEC column. To read the currently stored script, simply type ER. To execute the currently stored script type X and two digits (00 to 31) for the number of characters to execute. To execute the command above, simply type X03. You will get feedback the script is executing, plus the normal command feedback. Note, you only get 32 characters of execution; the rest of the memory is intended for nonvolatile storage of data.

If you get lost while you are playing around, simply type M (for MENU), to get a list of commands with a brief description.

7.0 ISaAC API Screen Shots

To facilitate API understand several screen shot of API operation are shown using a Tera Term Serial Emulator. Detailed API command descriptions are contained in Appendix A. Figure is the result of 'M' command (Help Screen)

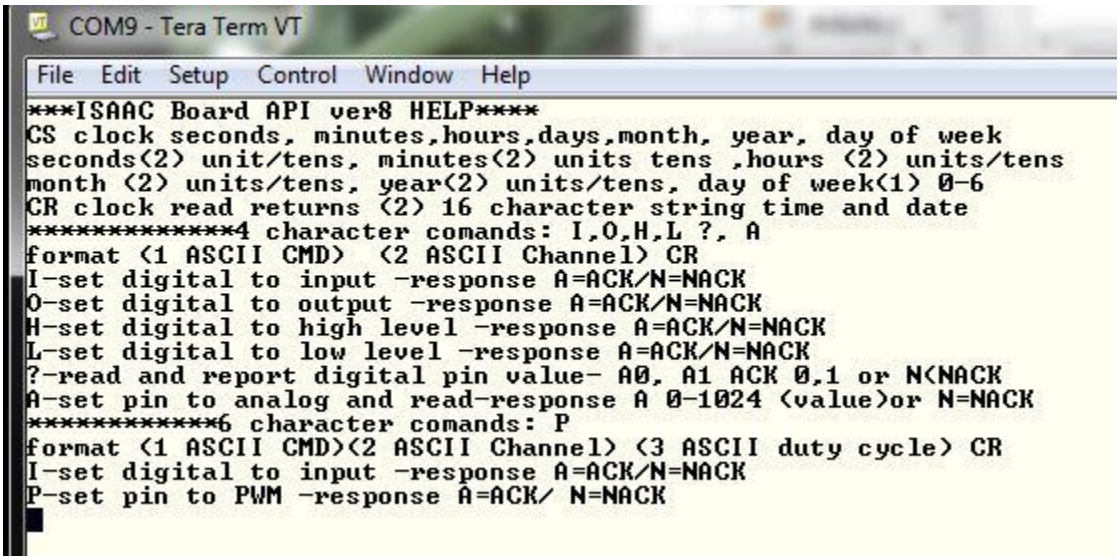


Figure 5: (M) command help screen

Figure 6 shows the use of the 'EW' EEPROM write to location 127 data 64, 'ER' EEPROM read from location 127 with 'A' and 64. The 'D' or DAC output command is also showed. A 'D'1023 causes a +2.5V level to appear on output of DAC. The 'D'511 changes the level to +1.25V. The 'D'255 changes the level to +.625 V, while 'D'0000 zeroes the DAC output.

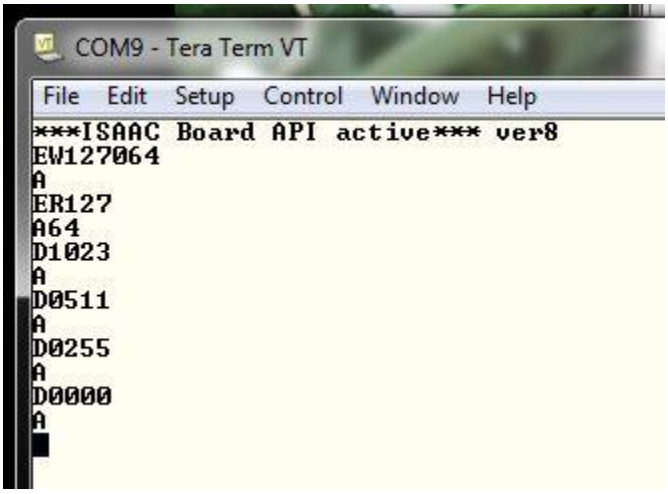


Figure 6: ER EW and DAC

Figure 7 shows the use of the 'O' 00 causing pin D0 to be an output. The 'H'00 and 'L'00 toggle DO from high to low. The 'I'01 causes pin D1 to be an input. The '?'01 reads the value of this digital input and returns A1 meaning 'ACK' and a high level. The 'A'14 causes A0 to be an analog input, the results are immediately returned as A1023 or 'ACK' with a ADC count of 1023. The next couples of commands are

reading the same analog input as the voltage level is changing

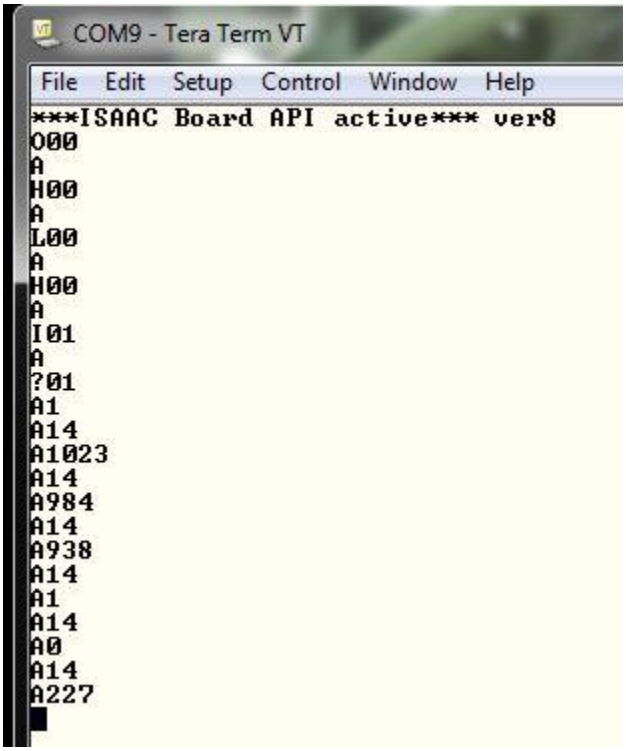


Figure 7 I/O Operations

Figure 8 shows the use of 'CS' RTCC clock setting, 'CR' RTCC clock read. Once the RTCC has been set, the 'T' command for alarm trigger can be used. The 'T'20 causes a ten second alarm single shot. The results in "ALARM-S" report. The 'T'21 causes a ten second alarm continuous. This results in "ALARM-M" reports every 10 seconds.

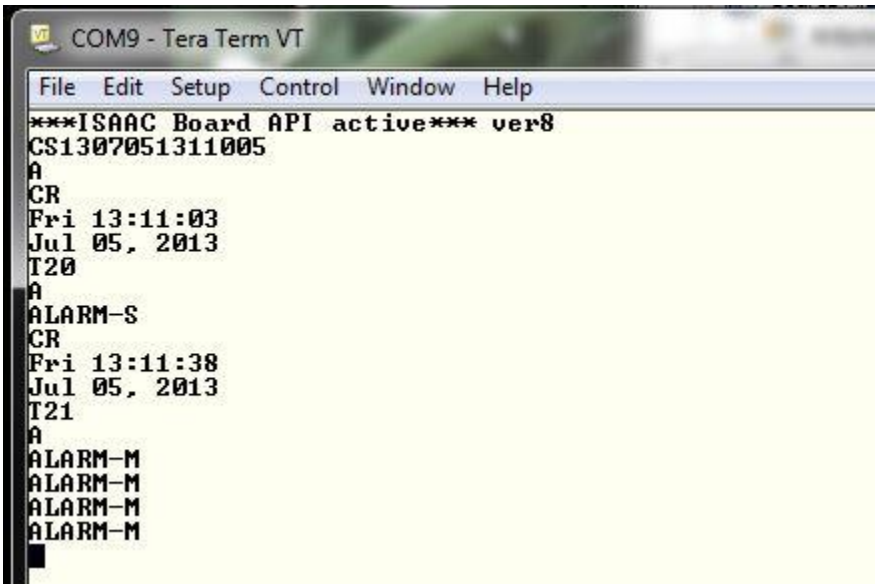


Figure 8 Clock and Alarm operations

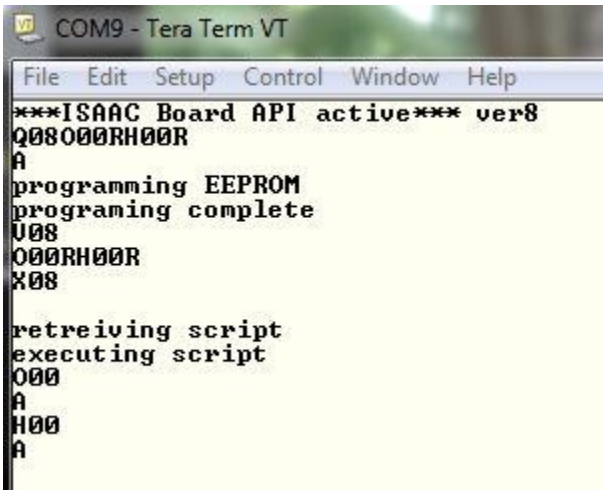


Figure 9 script build, view, and execute

In figure 9 we build an 8 character script, ,using the 'Q' command, which the first four character are 'O00R' setting D0 to output, 'R' is the 'echoed 'delimiter, and the second four are 'H00R' setting D0 output to high level. The 'Q' command takes the eight and programs them in EEPROM script location starting at address 0. Only one script is allowed and a script can be up to 32 characters long. This new script will 'clobber' any older scripts. The next command 'V' is used to viewed the stored script, again the value 8 is used for size. Finally the command 'X' is used to execute the script, again 8 is used for size.

As the script is executed you see the 'O' command executed and 'ACKed' and then the 'H' command then proceeds it.

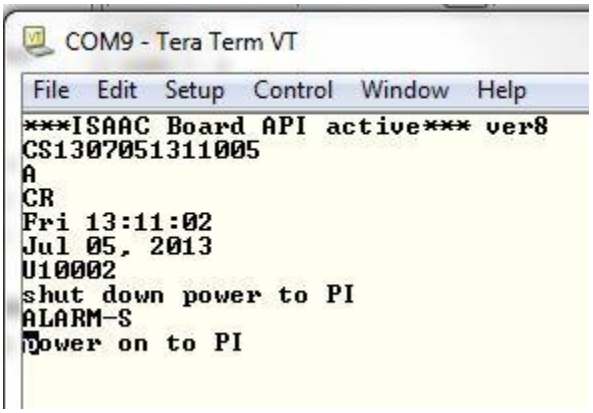


Figure 10: U command for power cycle of PI

Figure 10 shows the 'CS' and 'CR' command being executed to set and read the RTCC. The next command is 'U'. The string 'U10002' delays the PI shut down by 1000 milliseconds, alerts the system a shutdown is pending. The +5V power relay is opened. The ISaAC is still running and has a type 2 rate alarm trigger set. There are many types of rate available for minutes, hours, days , months, even year. Type 2 is just several minutes. Once the alarm trigger occur the ISaAC will close the +5V relay to the Raspberry PI , turning its power back on. This is a one-time event. ISaAC outputs a message once this happens.

8.0 Updating the API version

The ISaAC API will be periodically updated with new functionality. This API will be issued under new version numbers. All updated API will be published and downloadable via web site http://www.kibacorp.com. The updates will be posted as PIC32 MPLABX object code and can be used to immediately program ISaAC using a Microchip PICKIT3 programmer / debugger and the ISaAC provided In Circuit Serial Programmer Interface (ICSP). As the ISaAC comes preconfigure the PICKIT3 is not required for board operation. A PIC32MX boot loader capability for the ISaAC is under development. This will allow ISaAC to be programmed automatically by Raspberry Pi.

Appendix A –Detailed API Command Syntax

1. The 2 Character Command Set (M or Menu Help)

The format for 2 character command is:

- Command (single character)='M'
- Delimiter (1 character) 0x0d – this is the ASCII Carriage Return

This command response issues an 'A' (ACK) or 'N' (NACK). If 'A' then ISaAC will clear and home screen and the issue current version number and a list of all available commands.

2. The 3 Character Command Set ('CR') Clock read for RTCC

The format for 3 character command is show below. Prior to using the special command 'CR' (CLOCK READ), 'CS' (CLOCK SET) must be issued.

- Command (2 character '*CR*')
- Delimiter (1 character) 0x0d

This command causes the ISaAC to read the on-board ISaAC Real Time Clock Calendar (RTCC) and then issues a 16 character string response to the PI for current time including day of the week, hours, minutes, seconds, (i.e. "Fri 13:11:03 "), and then on a separate line, a 16 character string for current date to include year, month, day of month and year (i.e. "JUL 05, 2013"). This data can be captured and used within your Raspberry Pi's application.

3. The 4 Character Command Set ('T') for RTCC Alarm Trigger

The format for 4 character command is show ,prior to using 'T' a special command the 'CS' (CLOCK SET) must have been be issued.

- Command (1 character 'T')
- Rate Character: 0 = off, 1 = every 1 sec, 2= every 10 sec, 3= every 1 minute, 4=every 10 minute, 5 = every 1 hour, 6= every 1 day, 7 =every 1 week, 8 = every 1 month, 9 = every 1 year.
- Mode Character: 1 =one time trigger, 2 = one time trigger with power up of Raspberry PI, 3= continuous trigger
- Delimiter (1 character) 0x0d

This command causes the ISaAC to turn on or turn off (Rate = 0) the Real Time Clock Calendar (RTCC) alarm. A one-time trigger will output to the PI upon alarm a string "ALARM-S". A continuous trigger will output the string "ALARM-M" upon each alarm condition being met.

4. The 4 Character Command Set (I, O, H, L,'?' A) for ISaAC analog and digital I/O

The format for 4 character command is:

- Command (single character)
- Channel or pin (2 characters)
- Delimiter (1 character) 0x0d

- 4 character Commands and responses:
    - I or O-set digital to input or output, response ACK= 'A' or NACK ='N'
        - Available D0-D13 pin designators are 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, and 13. You can convert A0 to A5 to digital by using pin designators 14, 15, 16, 17, 18, and 19
    - H-set digital out high, response ACK ='A' or NACK ='N'
        - Write a high digital value to designated pin.
    - L-set digital out low , response ACK='A'/NACK= 'N'
        - Write a low digital value to designated pin.
    - ? -read and report digital value on pin- response ACK-'A' (1,0) or NACK- 'N'
        - Reads a digital value (0 or 1) for the associated pin and returns it as integer value.
    - A- set pin to analog input and read current binary value on that pin (works only for channels 14-19, or A0 to A5) response ACK-'A' (0-1024) or NACK- 'N'
        - Executes a 10 bit ADC conversion on selected channel. The pin is configured for analog. The function returns an integer value representing the ADC value from 0 to 1024. Only one analog channel is enabled at a time, available pins are 14, 15, 16, 17, 18 ,and 19
    - Any channel or pin designation uses two ASCII char '1' '9' for channel 19
        - Example -'0' and '1' or '01' for channel 1 and '0' and '0' or '00' for channel 0

5. The 4 Character Command Set (V, X) V-for EEPROM Script viewing and X for Script Execution

The format for 4 character commands for 'V', 'X' is:

- Command (single character) ='V' for viewing, 'X' for execution
- Number of Command Characters (2 characters) 00 to 31 (ASCII)
- Delimiter (1 character) 0x0d

The 'V' reads the designated number of characters stored as a script in EEPROM. Only one script can be stored in EEPROM it starts at address 0 and goes to address 31 for a total script size of 32 command bytes. The rest of EEPROM is available for user data storage using 'EW' command. .Delimiters appear during view as 'R'.

The 'X' command proceeds with retrieving the designated number of characters from EEPROM storage and executing them one at a time as a normal system entry over the serial port. It puts out two messages during the process "retrieving script" and then "executing script". During execution the system responds normally with echo of characters and 'A' responses.

6. The 6 Character Command Set (D) for ISaAC DAC Output

The format for 6 character command for Digital Analog Converter (DAC) is:

- Command (single character) ='D' for DAC Out
- DAC Value (4 characters) 0000 to 1023 (ASCII)
- Delimiter (1 character) 0x0d

The 'D' or DAC command, outputs a 10 bit count (0 to 1023 decimal) as 4 ASCII character equivalents to DAC located on ISaAC I2C bus. The output voltage is set to 2.5 volts/1024 for LSB *count.

7. The 6 Character Command Set ('ER') for ISaAC EEPROM Read

The format for 6 character command for EEPROM Read 'ER' is:

- Command (dual character) ='ER' for EPPROM Read
- EEPROM address (3 characters)- 000 to 127
- Delimiter (1 character) 0x0d

The 'ER' or EEPROM command, reads the 128X8 EEPROM on board the ISaAC. It uses the 3 ASCII characters to form an 8 bit binary value for the EEPROM address. If the command is acknowledged an 'A' and a three ASCII character (or eight bit result) is returned, otherwise an 'N' or NACK is returned.

8. The 7 Character Command Set (U) for ISaAC Raspberry PI power cycle

The format for 7 character command for 'U' is:

- Command (single character) ='U'
- Delay before shutdown in milliseconds (4 characters) – 0000 to 9999
- Trigger time to power on (1 = every 1 sec, 2= every 10 sec, 3= every 1 minute, 4=every 10 minute, 5 = every 1 hour, 6= every 1 day, 7 =every 1 week, 8 = every 1 month, 9 = every 1 year)
- Delimiter (1 character) 0x0d

The 'U' or command performs a controlled power cycle on Raspberry PI. A 'CS' command must be issued prior to a 'U' command. The 'U' waits for the specified delay before shutting down power to Raspberry PI (turning off +5V relay). Before it does this it sends out message "shutting down power". The power up cycle occurs within the ISaAC RTCC alarm established by the rate parameter. This is a one-time trigger that then turns on the relay for Raspberry PI power. An "ALARM-S" message string is sent to PI once the power has returned.

9. The 7 Character Command Set (P) for ISaAC PWM

The format for 7 character command for PWM 'P' is:

- Command (single character) ='P' for PWM
- Channel or pin (2 characters) – 09 (D9) or 10 (D10)
- Duty Cycle (3 characters) – 000 to 255
- Delimiter (1 character) 0x0d

The 'P' or PWM command, with current API accepts pins D9 and D10. The duty cycle can range from 0 (off) to 255 (on all the time). The command uses the PIC32 programmable peripheral system or PPS (D9) to output compare 1 OC1 using Timer 2 and D10 to output compare 3 OC3 using Timer 3. The PWM frequency is set to approximately 100 Hz to be compatible with Arduino.

10. The 9 Character Command Set (EW) for ISaAC EEPROM Write

The format for 6 character command for PWM 'P' is:

- Command (dual character) ='EW' for EEPROM Write
- EEPROM address (3 characters)- 000 to 127
- EEPROM Data (3 characters) = 000 to 255
- Delimiter (1 character) 0x0d

The "EW" or EEPROM write command, writes a byte of data (represented by 3 ASCII character for decimal equivalent 000 to 255) to a EEPROM address (represented by 3 ASCII character for hex equivalent 000 to 127). It returns an 'A' for ACK or 'N' for NACK.

11. The 16 Character Command Set ('CS') for ISaAC RTCC Clock Initialization and Set

The format for 16 character command is:

- Command (2 character) ='CS' for RTCC Clock set
- Date Setting (6 characters) - years(tens, units), days (tens, units), months (tens, units),
- Time Setting (6 characters) -Seconds (units, tens), Minutes (units, tens), Hours (units, tens)
- Day of week (1 character) 0 to 6 (0 =Sunday, 6 = Saturday)
- Delimiter (1 character) 0x0d

This command response issues an 'A' or 'N'. If 'A' then ISaAC will initialize and set the ISaAC RTCC. If 'N' no RTCC action will occur.

12. The Up to 32 Character Command Set ('Q') for Script build

The format for 16 character command is:

- Command (1 character) ='Q'
- Script size (two characters) 00 to 31
- Actual Script commands including 0x0d to build out script size.

This command immediately collects user type scripts to the character amount specified. Once the character count is reached the script is programmed into EEPROM starting at location 0. Only one script can be stored at a time. Once stored the script can be view and executed using 'V' and 'Q' commands.

Appendix B –ISaAC Hardware Description

A schematic for ISaAC is shown in figure 11. The +5V relay is mounted to the bottom of the board. In addition the GPIO mating connector for the Raspberry PI is on the bottom of the board. This facilitates connecting to the top of the Raspberry PI. In addition there is a mounting screw, space and nut for connecting the ISaAC mounting hole aligned with the Raspberry PI mounting hole. Male headers can be used top of board for the Arduino and GPIO connection accessible. The DAC has two small header pin on top side of board labeled 'output' and 'DAC'. The 'output' is already buffered output from the DAC. The 'DAC' is not buffered but allows the user to provide his own amplifier buffer, if that is deemed necessary. The ISaAC is configured on power up to turn on the +5V relay automatically. This can be optionally used to supply power to the Raspberry PI if the header jumper for PI power is selected on the top of the board. The ISaAC receives its power form the USB. Make sure a +5V @ 1 amp regulated wall transformer is feeding the ISaAC so it in turn can supply power to the Raspberry PI.

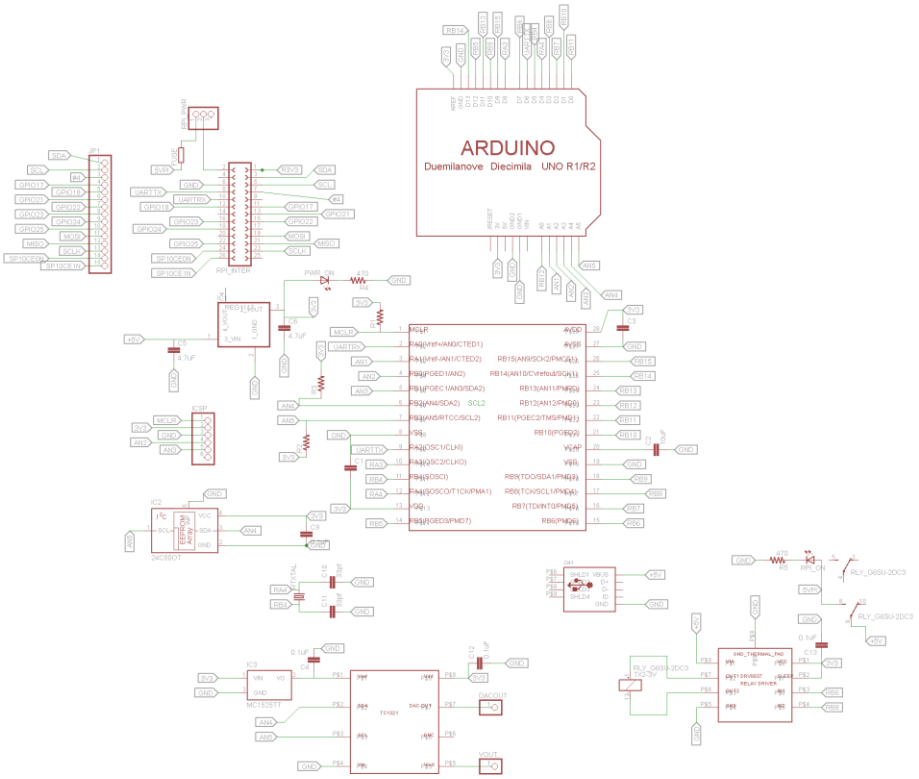


Figure 11; ISaAC Schematic

ISaAC Pin Table Designation

The ISaAC follows the ARDUINO pin format designation where D0 to D13 and A0 to A5 are identified with unique integers. Not all ARDUINO pins are available in this initial ISaAC release, five of them (D3, D4, D5, D6, D7) are special functions reserved for Raspberry Pi / ISaAC board communication and control.D3 and D& are used for relay control driver, D4 and D5 are used with the RTCC XTAL, D6 is used with PI serial communications. See chart below for all ISaAC pins; special function pins indicated in bold.

Designator (pin)	**ARDUINO**	**PIC32 pin**	**PIC32MX250F128B**
D0 (0)	RXD/PD0	22	RBP11\UART2RX
D1 (1)	TXD/PD1	21	RBP10\UART2TX
D2 (2)	INT0/PD2	16	INT0\RPB7
D3(3)	INT1/PD3	**17**	**RPB8-Relay Control**
D4 (4)	T0/PD4	12	T1CLK\RPA4
D5 (5)	T1/PD5	11	T2CLK\RPB4
D6 (6)	AIN0/PD6	**9**	**RPA2 U1RX (rs232 to PI TX)**
D7 (7)	AIN1/PD7	**15**	**RPB6 –relay control**
Internal to		**2**	**U1TX (rs232 to PI RX)**

board			
A0 (14)	ADC0/PC0	23	AN0\RA0 (AN12\RB12)
A1 (15)	ADC1/PC1	3	AN1\RA1
A2 (16)	ADC2/PC2	4	AN2\RB0\PGED1
A3 (17)	ADC3/PC3	5	AN3\RB1\PGEC1
A4 (18)	ADC4/SDA/ PC4	6	AN4\SDA2\RB2
A5 (19)	ADC5/SCL/ PC5	7	AN5\SCL2\RRB3
D8 (8)	ICP/PB0	10	IC2\RPA3
D9 (9)	OC1A/PB1	26	OC1\RB15
D10 (10)	OC1B/SS/P B2	18	OC3\RB9
D11 (11)	MOSI/OC2/ PB3	24	SDO1\OC4\RBP13
D12 (12)	MISO/PB4	14	SDI1\RPB5
D13 (13)	SCK/PB5	25	SCK1\RB14

Appendix C Installation of TeraTerm and USB drivers

Installing Tera Term

Down load the Tera Term application. It is open source and available for free from a number of internet sites. You can find Tera Term at this URL

http://logmett.com/index.php?/download/tera-term-480-freeware.html

When you first connect the USB-to-UART S27 ACRONAME interface via USB, Windows will prompt you for a driver. Navigate to the ARCONAME supplied drivers that are supplied with the product and download these to your computer. You can find drivers at this URL, or let Windows install automatically.

https://www.easierrobotics.com/cgi-bin/software

Once installed it will essentially represent a new COM. Use this new com port and then use the serial port configuration to configure Tera Term for 192008N1 with no hardware handshake.

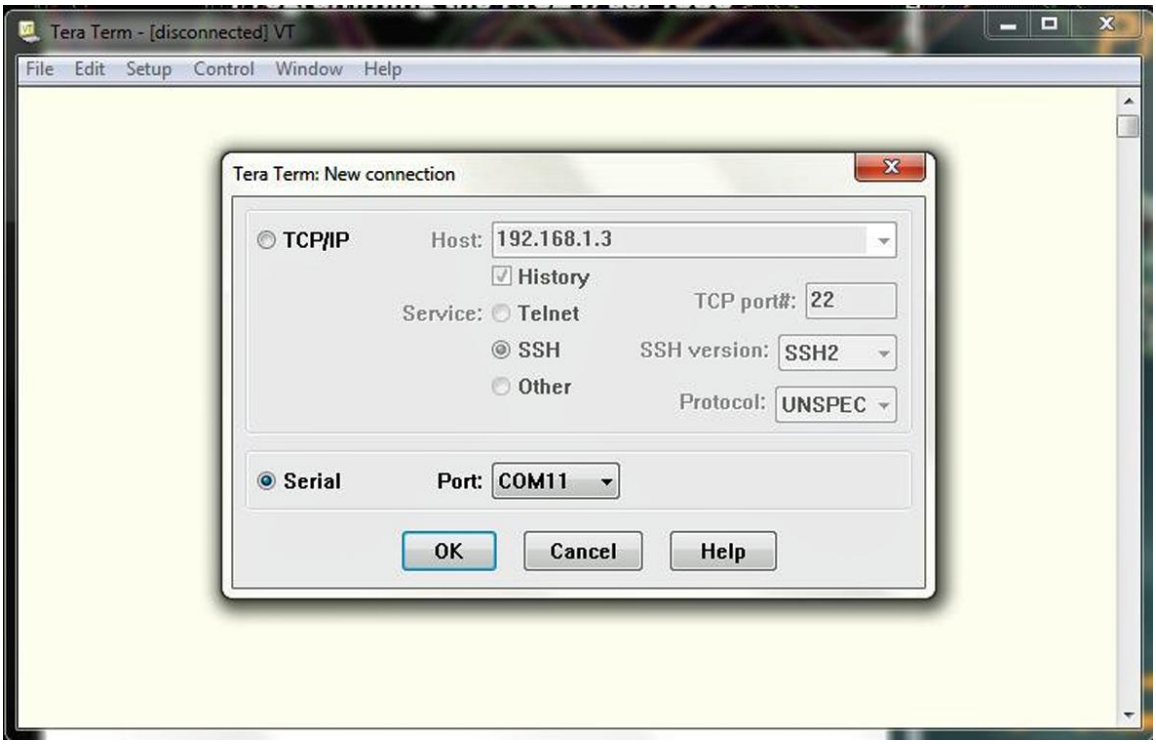


Figure 18: Tera Term Port selection

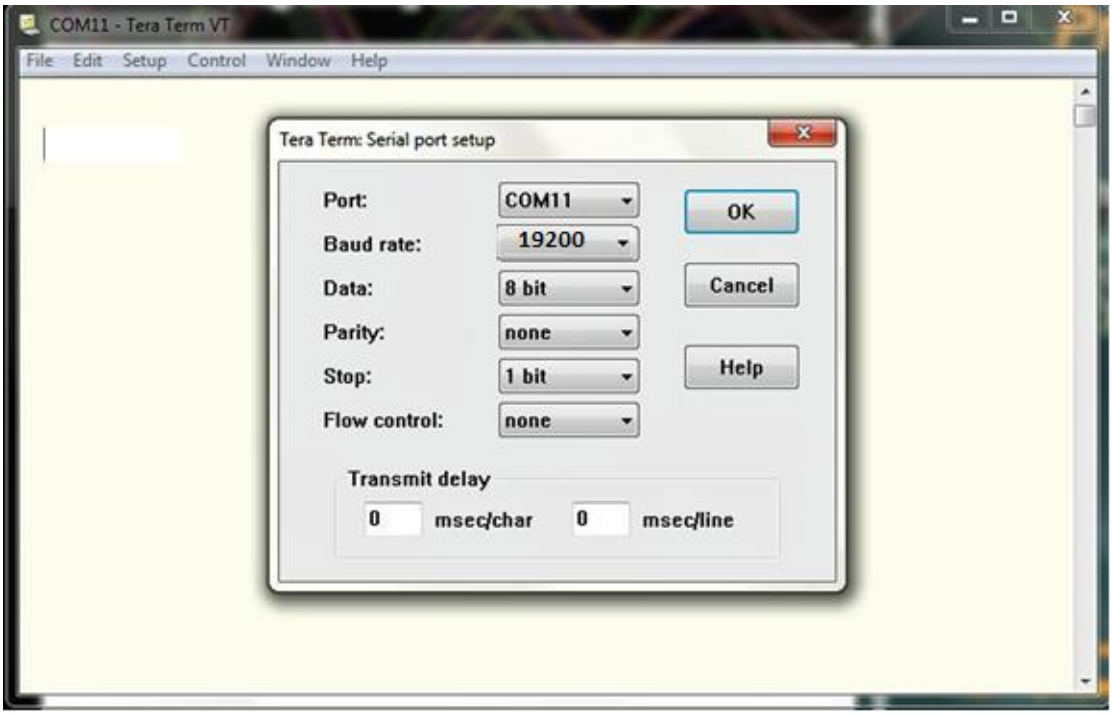


Figure 19: Tera Term Serial Port Setup