ANT Serial/Analog Converter

The ANT Serial/Analog Converter consists of an ANT+ radio configured as a receiver connected to an FTDI chip that resends the ANT data over a USB serial interface. The FTDI chip acts as a COM port on the computer it is attached to so it's straight forward to connect to software such as Matlab for collecting the serial data. For channels 1-4, it also outputs an analog 0 - 3.3V, 12-bit signal representing the value of the ANT+ parameter.

The system is like a repeater in that it just resends the ANT data over the serial port.

Setup:

The ANT to Serial Converter does not have any internal memory so it needs to be setup every time it is reconnected. The device supports up to 15 simultaneous channels and many various ANT+ profiles.

The user can set the profile and sensor number via text commands over the serial port. This can be done manually with a terminal emulator or through the Matlab program.

The format of the command is as shown below.

profile [channel] [profile number]

The "profile" command sets the profile for each channel according to the following table.

ANT+ Profile	Abbreviation	Profile Number
NONE – Channel Disabled	NONE	0
Muscle Oxygen	MO2	1
Heart Rate	HR	2
Bicycle Speed and Cadence	BSC	3
Bicycle Cadence Only	CAD	4
Bicycle Speed Only	SPD	5
Stride Based Speed and Distance	SDM	6
Fitness Equipment Profile	FEC	7
Cycling Power	PWR	8

The "sensor" command sets the ANT+ sensor number for each channel. For a Moxy sensor, the ANT+ sensor number is the number written on the top of the sensor. For other sensor types like heart rate straps, the sensor number can be identified by connecting the sensor to another device like a Garmin watch or the PerfPro Software and noting the sensor number.

sensor [channel] [sensor number]

The "show" command will show a table of all 15 channels with their assigned profiles and sensor numbers.

The “analog” command is used to enable or disable the analog output for channels 1-4. Only these 4 channels have analog output capability. The following table show the output for each of the ANT+ profiles that are supported.

analog [channel] [enable (1/0)]

ANT+ Profile	Profile Number	Value Range	Output
Muscle Oxygen	1	0% to 100%	0 to 3.3 V
Heart Rate	2	0 to 256 BPM	0 to 3.3 V
Bike Speed and Cadence (Cadence)	3	0 to 256 RPM	0 to 3.3 V
Bike Cadence Only	4	0 to 256 RPM	0 to 3.3 V
Bike Speed Only (wheel RPM)	5	0 to 128 RPM	0 to 3.3 V

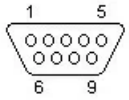
The “dac” command allows the analog output to momentarily be set to a fixed value. The purpose of this command is to set values for calibration. The dac is a 12-bit output so the dac values are 0 to 4095.

dac [channel] [value 0-4095]

Analog Outputs:

DB9 Connector Pinout

PIN Number	Function
1	Ground
2	Ground
3	Ground
4	Ground
5	Ground
6	Analog Channel 1
7	Analog Channel 4
8	Analog Channel 2
9	Analog Channel 3



DB-9 Male

ANT+ Data

ANT+ sends data in 8 Byte (64 bit) Pages. The format of the pages is defined by each of the various ANT+ profiles. The first byte on the page identifies the page number. The remaining 7 bytes contain the data as defined by the profile. These pages are typically sent at near 4 Hz although 2 Hz and 1 Hz are also sometimes used.

Conversion to Serial Data
The ANT to Serial Converter encodes each page in a way that can be reliably transmitted as a serial port message. It used a 20-Byte serial message for this.

The first byte is always 0x21 (this is an ! as an ASCII character). 0x21 will never be another Byte in the message so this is always certainly the starting character of a message.

The second Byte is always 0x30 plus the Matlab Channel number. The ANT+ pages do not identify the channel or the ANT+ profile. The ANT to Serial Converter supports up to 15 sensors simultaneously. This is the only Byte in the message that will start with a 0x30 in Hex. This way Matlab can be certain that this Byte is the Channel Byte. It is up Matlab to keep track of which ANT+ profile is associated with each channel. The ANT+ profile is not identified in the message.

0x31 is Matlab's Channel #1
….
0x3F is Matlab's Channel #15

It should be noted that Matlab starts index arrays with 1 where C++ typically starts index arrays with 0. The channels are numbered from 1 to 15 in Matlab. Inside of the ANT to Serial Converter source code, the channels are numbered from 0 to 14. The ANT to Serial Converter handles this offset so when programming in Matlab the channels should always be 1 to 15.

The 3rd through 18th Bytes of the serial message contain the 8 Bytes from the ANT+ data page. Each of the ANT+ bytes is broken into two nibbles and placed in 2 serial message Bytes. All serial message bytes containing ANT+ data start with a 0x40 in Hex to identify the data as the ANT+ data.

The final 2 Bytes in the serial message are the carriage return and line feed characters. These are typical end of message characters and they make the data show up nicely in rows if it is collect as a text file for example.

The following tables summarize Serial Message Bytes.

Matlab Byte 1	0x21	Start of Message Character (!) Will never be used in the rest of the message
Matlab Byte 2	0x3_	Channel Number Nibble (0-15) This will be the decimal value as a character
Matlab Byte 3	0x4_	ANT Byte 0 Least Significant Nibble
Matlab Byte 4	0x4_	ANT Byte 0 Most Significant Nibble
Matlab Byte 5	0x4_	ANT Byte 1 LSN
Matlab Byte 6	0x4_	ANT Byte 1 MSN
Matlab Byte 7	0x4_	ANT Byte 2 LSN
Matlab Byte 8	0x4_	ANT Byte 2 MSN
Matlab Byte 9	0x4_	ANT Byte 3 LSN
Matlab Byte 10	0x4_	ANT Byte 3 MSN
Matlab Byte 11	0x4_	ANT Byte 4 LSN
Matlab Byte 12	0x4_	ANT Byte 4 MSN
Matlab Byte 13	0x4_	ANT Byte 5 LSN
Matlab Byte 14	0x4_	ANT Byte 5 MSN
Matlab Byte 15	0x4_	ANT Byte 6 LSN
Matlab Byte 16	0x4_	ANT Byte 6 MSN
Matlab Byte 17	0x4_	ANT Byte 7 LSN
Matlab Byte 18	0x4_	ANT Byte 7 MSN
Matlab Byte 19	0x0D	End of Message Character \r standard message terminator
Matlab Byte 20	0x0A	End of Message Character \n standard message terminator

When reading this data in to Matlab, the following verifications are recommended.

- Setup the serial port to 115200 baud with a “CR” terminator.
- Use fRead to read 20 Bytes at a time. The messages are always 20 bytes.
- Verify that the start and end Bytes are correct to be sure that the message is complete.
- Flush the serial buffer if a message is incomplete or invalid.

Decoding the Data:

The following tables are a guide for decoding the ANT Page data in Matlab or some other serial reader. The ANT data page formats are defined for each of the various ANT+ profiles. It is up to Matlab to know which ANT+ profile is associated with each channel. ANT Byte 0 identifies the page number to use for decoding.

Serial Byte 0		Serial Byte 1		Serial Byte 2		Serial Byte 3		Serial Byte 4		Serial Byte 5		Serial Byte 6		Serial Byte 7	
!		Channel		ANT Byte 0				ANT Byte 1				ANT Byte 2			
0x2	0x1	0x3	Ch	0x4	LSN	0x4	MSN	0x4	LSN	0x4	MSN	0x4	LSN	0x4	MSN
			Bit 3 Bit 2 Bit 1 Bit 0		Bit 3 Bit 2 Bit 1 Bit 0		Bit 7 Bit 6 Bit 5 Bit 4		Bit 3 Bit 2 Bit 1 Bit 0		Bit 7 Bit 6 Bit 5 Bit 4		Bit 3 Bit 2 Bit 1 Bit 0		Bit 7 Bit 6 Bit 5 Bit 4
Matlab Byte 1		Matlab Byte 2		Matlab Byte 3		Matlab Byte 4		Matlab Byte 5		Matlab Byte 6		Matlab Byte 7		Matlab Byte 8	

Serial Byte 8		Serial Byte 9		Serial Byte 10		Serial Byte 11		Serial Byte 12		Serial Byte 13		Serial Byte 14		Serial Byte 15	
ANT Byte 3				ANT Byte 4				ANT Byte 5				ANT Byte 6			
0x4	LSN	0x4	MSN	0x4	LSN	0x4	MSN	0x4	LSN	0x4	MSN	0x4	LSN	0x4	MSN
	Bit 3 Bit 2 Bit 1 Bit 0		Bit 7 Bit 6 Bit 5 Bit 4		Bit 3 Bit 2 Bit 1 Bit 0		Bit 7 Bit 6 Bit 5 Bit 4		Bit 3 Bit 2 Bit 1 Bit 0		Bit 7 Bit 6 Bit 5 Bit 4		Bit 3 Bit 2 Bit 1 Bit 0		Bit 7 Bit 6 Bit 5 Bit 4
Matlab Byte 9		Matlab Byte 10		Matlab Byte 11		Matlab Byte 12		Matlab Byte 13		Matlab Byte 14		Matlab Byte 15		Matlab Byte 16	

Serial Byte 16		Serial Byte 17		Serial Byte 18		Serial Byte 19									
ANT Byte 7				Terminators											
0x4	LSN	0x4	MSN	0x0D		0x0A									
	Bit 3 Bit 2 Bit 1 Bit 0		Bit 7 Bit 6 Bit 5 Bit 4												
Matlab Byte 17		Matlab Byte 18		Matlab Byte 19		Matlab Byte 20									

When reconstructing ANT numbers across multiple ANT Bytes, the higher numbered ANT Byte contains the Most Significant Bits. The higher numbered bits are most significant.

Example #1

In the MO2 profile, the THb reading is 12 bits consisting of ANT Byte 4 and bits 0:3 of ANT Byte 5.

Bits 0:3 of Byte 5 are the Most significant followed by Byte 4 so the 12 bit number would be:

Byte 5	Byte 5	Byte 5	Byte 5	Byte 4	Byte 4	Byte 4	Byte 4	Byte 4	Byte 4	Byte 4	Byte 4
Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Matlab Byte 13 && 0x0F << 8				Matlab Byte 12 && 0x0F << 4				Matlab Byte 11 && 0x0F			

Example #2

For a second example, in the MO2 profile, the current SmO2 reading is 10 bits consisting of ANT Byte 6, bits 6 and 7, and ANT Byte 7 so the 10 bit number would be:

Byte 7	Byte 7	Byte 7	Byte 7	Byte 7	Byte 7	Byte 7	Byte 7	Byte 6	Byte 6
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6
Matlab Byte 18 && 0x0F << 6				Matlab Byte 17 && 0x0F << 2				Matlab Byte 16 && 0x0C >> 2	

Example Matlab Code:

```
clc
clear all

mask4 = uint8(15);
mask2 = uint8(12);

dataPeriod = 1/24/3600; %1 second update
graphPeriod = 1/24/3600; %1 second update
savePeriod = 60/24/3600; %60 second update

profile = zeros(1,15);
profile(1) = 5;
profile(2) = 0;

sensorID = zeros(1,15);
sensorID(1) = 1;
sensorID(2) = 0;

dataParsed = zeros(15,3);  %15 channels and 3 parameters per channel
dataArray = zeros(1,5);

previousSpeedEventTime = 0; %Sets variables for bike speed calculation
previousSpeedRevolutionCount = 0;

s = serial('COM5', 'TimeOut', .1, 'BaudRate', 115200, 'Terminator', 'LF');
%s = serial('COM5', 'TimeOut', 1, 'BaudRate', 115200, 'Terminator', 'LF');
%s = serial('COM5', 'BaudRate', 115200, 'Terminator', 'CR');
%s = serial('COM5', 'TimeOut', 1, 'BaudRate', 115200);
%s.InputBufferSize = 500;  % set to 500 to have a max of 25 readings backed
up
starttime = 0;

try
     fopen(s);
    handles.s = s;
catch e
    errordlg(e.message);
end

for i = 1:15
    fprintf(s,'profile %u %u',[i profile(i)]);
    fprintf(s,char(13));
    pause(.05);
    fprintf(s,'sensor %u %u',[i sensorID(i)]);
    fprintf(s,char(13));
    pause(.05);
```

```
end
flushinput(s);

arrayCount = 1;

startTime = now;
prevDataTime = startTime;
prevGraphTime = startTime;
prevSaveTime = startTime;
prevSaveCount = 0;

fprintf(s,'start');
fprintf(s,char(13));

running = 1;
h=msgbox('Click here to Stop');

while running==1
    if s.bytesavailable > 19
%         try
            dataNibbles=fread(s,20,'uint8');
            if dataNibbles(1)== 33 && dataNibbles(19)== 13 && dataNibbles(20)
== 10
                dataChannel = bitand(dataNibbles(2),mask4);
                switch profile(dataChannel)
                    case 0
                        %Profile not set for this channel - Shouldn't get
here
                    case 1 %MO2 Profile
                        page = bitshift(bitand(dataNibbles(4),mask4),4) +
bitand(dataNibbles(3),mask4);
                        if page == 1                    %Decode Page 1 Data
                            nibble1 = uint32(bitand(dataNibbles(13),mask4));
                            nibble2 = uint32(bitand(dataNibbles(12),mask4));
                            nibble3 = uint32(bitand(dataNibbles(11),mask4));
                            dataParsed(dataChannel,1) =
double(bitshift(nibble1,8)+ bitshift(nibble2,4) + nibble3)*.01;
                            nibble1 = uint32(bitand(dataNibbles(18),mask4));
                            nibble2 = uint32(bitand(dataNibbles(17),mask4));
                            nibble3 = uint32(bitand(dataNibbles(16),mask2));
                            dataParsed(dataChannel,2) =
double(bitshift(nibble1,6)+ bitshift(nibble2,2) + bitshift(nibble3,-2))*.1;
                        end
                    case 2 %Heart Rate
                        page = bitshift(bitand(dataNibbles(4),mask4),4) +
bitand(dataNibbles(3),mask4);
                        if page == 0 || page == 2 || page == 4 || page == 5
                            nibble1 = uint32(bitand(dataNibbles(18),mask4));
                            nibble2 = uint32(bitand(dataNibbles(17),mask4));
                            dataParsed(dataChannel,1) =
double(bitshift(nibble1, 4) + nibble2);
                        end
                    case 5 %Bike Speed Only
                        page = bitshift(bitand(dataNibbles(4),mask4),4) +
bitand(dataNibbles(3),mask4);
                        if page >= 0 && page <= 5
                           tireDiameter = .66; %Bike tire diameter in meters
```

```
                            tireCircumference = 2.2; %(tireDiameter / 1000) *
pi; %Tire circumference in kilometers
                            nibble1 = uint32(bitand(dataNibbles(14),mask4));
                            nibble2 = uint32(bitand(dataNibbles(13),mask4));
                            nibble3 = uint32(bitand(dataNibbles(12),mask4));
                            nibble4 = uint32(bitand(dataNibbles(11),mask4));
                            currentSpeedEventTime =(double(bitshift(nibble1,
12)) + double(bitshift(nibble2, 8)) + double(bitshift(nibble3, 4)) +
double(nibble4));
                            nibble1 = uint32(bitand(dataNibbles(18),mask4));
                            nibble2 = uint32(bitand(dataNibbles(17),mask4));
                            nibble3 = uint32(bitand(dataNibbles(16),mask4));
                            nibble4 = uint32(bitand(dataNibbles(15),mask4));
                            currentSpeedRevolutionCount =
(double(bitshift(nibble1, 12) + double(bitshift(nibble2, 8)) +
double(bitshift(nibble3, 4)) + double(nibble4)));
                            if currentSpeedEventTime > previousSpeedEventTime
&& (currentSpeedEventTime - previousSpeedEventTime) < 10240
                               RevolutionsPerSecond =
(currentSpeedRevolutionCount - previousSpeedRevolutionCount) /
((currentSpeedEventTime - previousSpeedEventTime) / 1024);
                               dataParsed(dataChannel,1) = tireCircumference *
(RevolutionsPerSecond * 3600);
                               revolutioncount = currentSpeedRevolutionCount -
previousSpeedRevolutionCount;
                               eventtime = currentSpeedEventTime -
previousSpeedEventTime
                               variable = dataParsed(dataChannel,1)
                            elseif currentSpeedEventTime -
previousSpeedEventTime < 0
                               rollover = ((65536 - previousSpeedEventTime) +
currentSpeedEventTime) / 1024
                               RevolutionsPerSecond =
(currentSpeedRevolutionCount - previousSpeedRevolutionCount) / rollover;
                               dataParsed(dataChannel,1) = tireCircumference *
(RevolutionsPerSecond * 3600);
                               variable = dataParsed(dataChannel,1)
                            end
                            previousSpeedRevolutionCount =
currentSpeedRevolutionCount;
                            previousSpeedEventTime = currentSpeedEventTime;
                        end
                    case 6 %SDM
                        page = bitshift(bitand(dataNibbles(4),mask4),4) +
bitand(dataNibbles(3),mask4);
                        if page == 1
                            nibble1 = uint32(bitand(dataNibbles(9),mask4));
                            nibble2 = uint32(bitand(dataNibbles(10),mask4));
                            nibble3 = uint32(bitand(dataNibbles(12),mask4));
                            dataParsed(dataChannel,1) =
double(bitshift(nibble2,4)+ nibble1);
                            dataParsed(dataChannel,1) =
dataParsed(dataChannel,1) + (double(nibble3) / 16);
                            nibble1 = uint32(bitand(dataNibbles(11),mask4));
                            nibble2 = uint32(bitand(dataNibbles(13),mask4));
                            nibble3 = uint32(bitand(dataNibbles(14),mask4));
                            dataParsed(dataChannel,2) = double(nibble1);
```

```
                            dataParsed(dataChannel,2) =
dataParsed(dataChannel,2) + ((double(bitshift(nibble3,4) + nibble2)) / 256);
                        elseif page == 2
                            nibble1 = uint32(bitand(dataNibbles(9),mask4));
                            nibble2 = uint32(bitand(dataNibbles(10),mask4));
                            nibble3 = uint32(bitand(dataNibbles(12),mask4));
                            dataParsed(dataChannel,3) =
double(bitshift(nibble2,4)+ nibble1);
                            dataParsed(dataChannel,3) =
dataParsed(dataChannel,1) + (double(nibble3) / 16);
                            nibble1 = uint32(bitand(dataNibbles(11),mask4));
                            nibble2 = uint32(bitand(dataNibbles(13),mask4));
                            nibble3 = uint32(bitand(dataNibbles(14),mask4));
                            dataParsed(dataChannel,2) = double(nibble1);
                            dataParsed(dataChannel,2) =
dataParsed(dataChannel,2) + ((double(bitshift(nibble3,4) + nibble2)) / 256);
                        end
                    end
                else
                  flushinput(s);
                end
%             catch e
%                 errordlg(e.message);
%             end
        end

        if now - prevDataTime > dataPeriod
            %Fill the data Array once every time period
            prevDataTime = prevDataTime + dataPeriod;
            dataArray(arrayCount,1) = (prevDataTime-startTime)*24*3600;     %Time
Stamp in seconds
            dataArray(arrayCount,2) = dataParsed(1,2);  %Device 1 SmO2
            dataArray(arrayCount,3) = dataParsed(1,1);  %Device 1 THb
            dataArray(arrayCount,4) = dataParsed(2,2);  %Device 2 SmO2
            dataArray(arrayCount,5) = dataParsed(2,1);  %Device 2 THb
            dataArray(arrayCount,6) = dataParsed(3,1);  %Device 3 Speed
            arrayCount = arrayCount + 1;
            dataParsed = zeros(15,3);
        end

        if ishandle(h) == 0
            running = 0;
        end

        if (now - prevGraphTime > graphPeriod) | (running == 0)
            %Update the graph
            prevGraphTime = prevGraphTime + graphPeriod;
            plot(dataArray(:,1),dataArray(:,2),'g*-');
            hold on;
            plot(dataArray(:,1),dataArray(:,4),'g*:');
            ylim([0 100]);
            hold off;
            drawnow;
        end

        if (now - prevSaveTime > savePeriod) | (running == 0)
            %Save Data to .csv file
```

```
        prevSaveTime = prevSaveTime + savePeriod;

        fid = fopen('dataArray.csv','a');
        for i = prevSaveCount+1:arrayCount-1

fprintf(fid,'%f,%f,%f,%f,%f,%f\n',dataArray(i,1),dataArray(i,2),dataArray(i,3
),dataArray(i,4),dataArray(i,5),dataArray(i,5));
        end
        fclose('all');
        prevSaveCount = arrayCount-1;
    end

end

fprintf(s,'stop');
fprintf(s,char(13));
fclose(s);
delete(instrfind);
```