Readme File for: mrmSampleApp.c

This zip file contains all of the sample source code necessary to demonstrate how to interface to P4xx MRM module. This code uses the functions in mrm.c to:

- connect to the radio via Ethernet, USB, or serial
- get the MRM configuration from the radio and print it
- get the status/info from the radio and print it
- start MRM and receive scans from it according to user defined parameters

The sample code does not include the executable. The executable has not been provided because our C compiler might not produce code that will run on all machines. However, all files needed to compile the program have been included so that the code can be compiled with your compiler of choice for operation on your machine of choice.

The sample code has two uses. First, it can be used as a reference and thereby accelerate the users development of special purpose code. Second, the code can be used to extract raw radar scans from a P4xx at faster rates than are achievable with MRM-RET.

The balance of this document will focus on a) discussing issues and options associated with extracting raw radar scans and b) describing how to operate the sample code.

Maximizing Transfer of Raw Radar Scans

This capability can be important for users who analyze Doppler radar returns from fast moving targets. Basically, these applications can require that radar scans be generated at a higher rate than the P4xx to Host communications channel can support. This limitation manifests itself as dropped packets. In other words, when using MRM-RET, if the user selects 5000 scans at a rate that is "too fast" then packets will be randomly dropped and logged data will be missing anywhere from "a few" to "many" scans.

There are several factors which limit communications. These include:

- the speed of the Host computer
- the interface type (USB or Ethernet)
- competition with other programs which might be running concurrently with either MRM-RET or the sample app
- software and hardware bottlenecks in the P4xx.

To test the limits of data transfer rate, the transfer rate was measured for a variety of PIIs and scan sizes. Scan sizes are measured in "Quanta" where one Quanta equals approximately 5800ps. Increasing the Quanta will increase the total amount of data to be transferred. Increasing the PII will increase the integration and therefore reduce the rate at which data will be generated.

The results of the test has been captured in the two tables shown in **Figure 1**. This upper table is a matrix of PII vs Quanta. The value in each matrix cell is the fastest rate (Hz) at which scans

can be generated in the P4xx. A green cell indicates that MRM-RET was able to successfully collect and log all of the requested scans. A red cell indicates that at least one scan was dropped.

These transfer rates do NOT include overheads in the P4xx. Therefore, the effective scan generation rate is a bit less. That information is captured in the lower table and indicates that the internal overheads reduce the transfer rate by 10-20%. For example, for Quanta = 3, the lowest integration rate which resulted in no packet loss was achieved at PII 11. This corresponds to a predicted rate of 205Hz. In practice, 10,000 scans were logged in 58 seconds for an effect rate of 172 Hz. In this test, performance was the same over both the Ethernet and USB interfaces.

Highest Rep Rate (Hz)

PII	Quanta 1	2	3	4	5	6	7	8
6	19,727	9,863	6,576	4,932	3,945	3,288	2,818	2,466
7	9,863	4,932	3,288	2,466	1,973	1,644	1,409	1,233
8	4,932	2,466	1,644	1,233	986	822	705	616
9	2,466	1,233	822	616	493	411	352	308
10	1,233	616	411	308	247	205	176	154
11	616	308	205	154	123	103	88	77
12	308	154	103	77	62	51	44	39
13	154	77	51	39	31	26	22	19
14	77	39	26	19	15	13	11	10
15	39	19	13	10	8	6	6	5

Scans sent	10000	5000	10000	10000	10000	10000	10000	10000
Time (Sec)	76	38	58	142	92	110	126	143
predicted rate (hz)	154	154	205	77	123	103	88	77
actual rate (Hz)	132	132	172	70	109	97	79	70
predicted bit rate	473,088	946,176	1,889,280	946,176	1,889,280	1,898,496	1,892,352	1,892,352

Figure 1: Results of transfer rate testing.

As indicated earlier there are other factors which can affect transfer rate. For example, one of the most significant effects is the speed of the Host computer. This experiment was repeated for a limited number of different PIIs, using different computers and comparing C-Code to MRM-RET performance over both USB to Ethernet. These results are summarized in **Figure 2**. The upper table was collected on a low cost, 1 year old machine (Alan P's) and the lower table was generated using a more recent and higher performance machine (Mike E's). The results shown in **Figure 1** were compiled using a third and slightly older machine.

On Alan P's Laptop…

	C-code Ethernet	MRM Ethernet	C-code USB	MRM USB	Time to complete
PII 10, 4 s	na	Dropped 1000	Dropped 2	Dropped 1000	0:20
PII 11, 4 s	Dropped 50	Fine	Dropped 2	Fine	0:40
PII 12, 4 s	Dropped 12	Fine	Fine	Fine	1:15
PII 13, 4 s	Fine	Fine	Fine	na	2:25

On Mike E's Laptop…

	C-code Ethernet	MRM Ethernet	C-code USB	MRM USB	Time to complete
PII 8, 4 s	Dropped 51	Fine (7 broke)	Dropped 4290		
PII 9, 4 s	Fine		Dropped 4531		
PII 10, 4 s	Fine		Fine	Dropped 1518	
PII 11, 4 s	Fine		Fine	Dropped 74	
PII 12, 4 s	Fine		Fine		

Figure 2: Comparison of performance on different machines and different interfaces for a small selection of parameters. (4 s = 4 Swaths/Quanta. This produces scans which are 23,437ps in length)

Mike E's faster machine outperformed the results shown in **Figure 1**. In particular, the best performance was achieved using MRM-RET and Ethernet (transfer rate ~1000Hz). The best performance through a USB connection was achieved using the sample C-Code (~250Hz).

One user of the equipment has reported transfer rates as high as a few KHz.

Operating the Sample Code as a standalone APP.

To run the APP, open the 150-0107C P400 MRM Host C Sample .zip file and move all of the files to a folder of your choice and then compile the program mrmSampleApp.

When you run this App be aware that it does not have a friendly GUI. It runs from the command line. To run the APP, use Windows Explorer to navigate to the directory which contains the APP, then hold the shift key down and right click the mouse. A drop down window will appear and you should click on "Open command window here". The command window shown in **Figure 3** should appear.

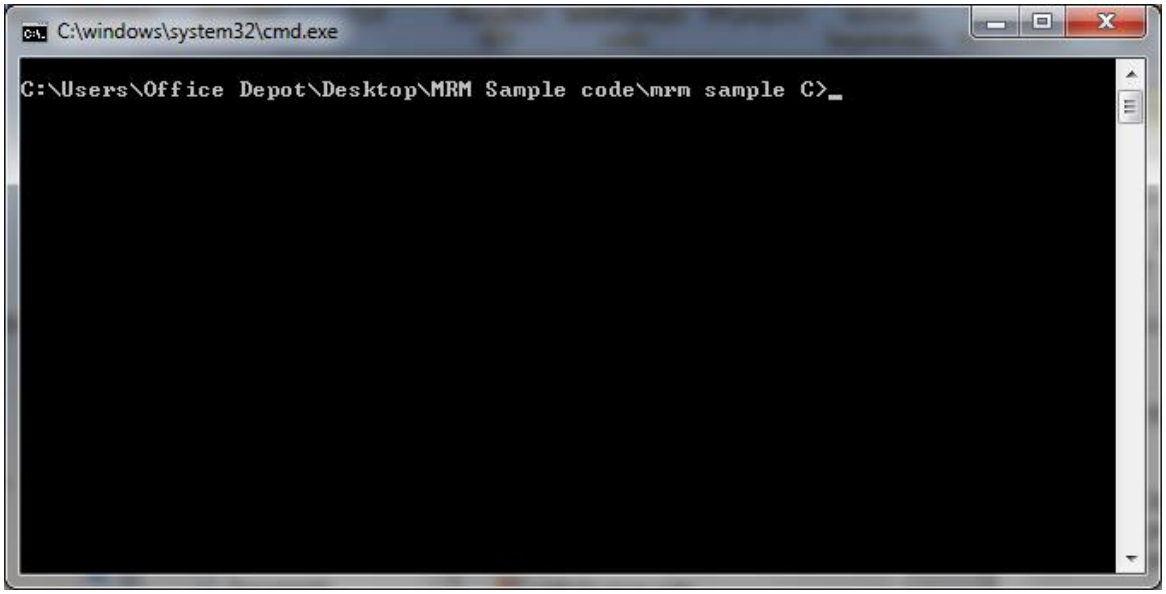

Figure 3: Command window

Entering "mrmSampleApp.exe" and a carriage return will produce the menu of options shown in **Figure 4.**

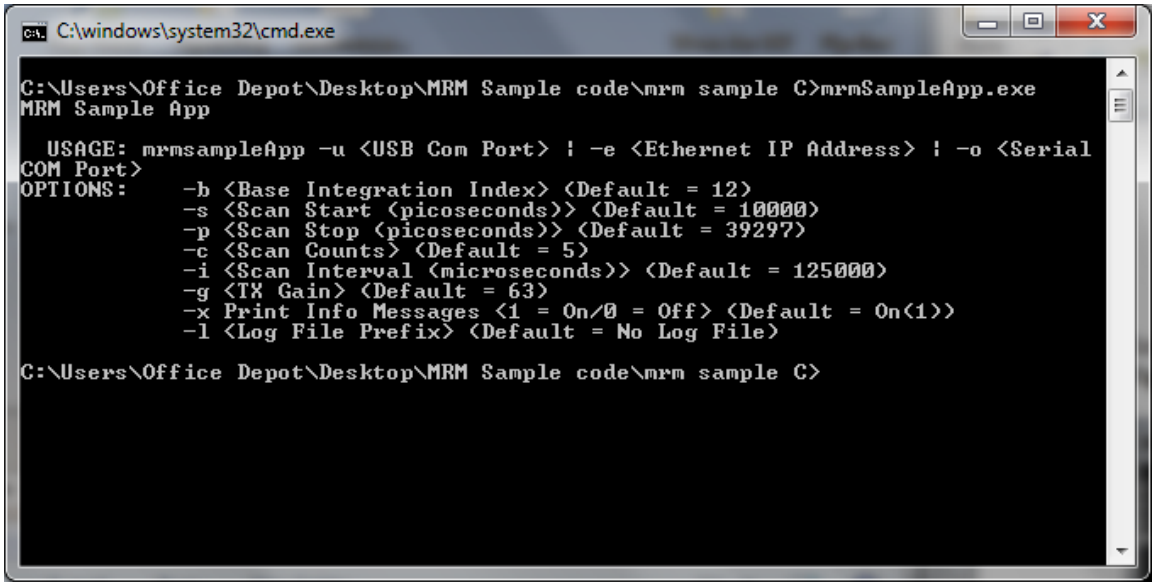

Figure 4: Menu of options

The parameters are selected using the following structure:

mrmsampleApp<space>-u<space>COM1<space>…

For example, entering the following:

mrmsampleApp –u com1 –b 10 –p 33000 –c 10000 –i 0 –g 40 –x 0 –l test_run <CR>

will cause the SampleApp to do the following:

- connect to the P4xx through USB Com Port 1(-u com 1)
- collect data using PII 10 (-b 10)
- since scan startpoint was not specified (-s option) it will be set thedefault value of 10,000
- set the scan endpoint to 33,000 (-p 33000) (this will collect 4 full swaths of ~5800*4 ps. Since the P4xx cannot collect a partial swath, it will round up the endpoint so that is actually collects and reports from 10,000 to 33437ps))
- 10,000 scans will be collected (-c 10000)
- the interval between readings will be as fast as possible (-i 0)
- the transmit power will be set to a gain of 40
- debug messages will be not be printed to the command line (-x 0)
- and the scans will be stored to the file "test_run.csv".