**Rebuilding the Educational Radar Real Time Program[[1]](#footnote-1)**

**Frank Robey, MIT Lincoln Laboratory, fcr@ll.mit.edu**

## Setting up the Radar build environment

## The RTP has been built and run under both Windows and Linux.

## Building the radar RTP for Windows relies on Microsoft Visual Studio. Original development was accomplished using VS2010 and more recently the build uses VS2017. Compiling works well under either the community or the professional edition. The operating system does not appear to be critical. The original radar RTP was developed and run under Windows 7, 32 bit. The program is now being developed and run under Windows 10, 64 bit.

1. Create directories c:\data, c:\lib, c:\include, and c:\src. If you do not care to compile the programs yourself and run the program with the pre-built binary and libraries, skip to step 9.

The data directory default is c:\data. Ideally this would be a separate partition on the disk so that if the recording filled up the file system then the OS partition would not be impacted.

1. Install Visual Studio 2017.
2. Download and install the fftw libraries prebuilt libraries and sources for windows from:

<http://www.fftw.org/download.html>

Move the sources to c:\src\fftw, and then unpack the zip files to the same directory

Open a Visual studio command prompt (windows key/scroll to Visual Studio 2017/Developer command prompt for VS2017) and then change directory (cd c:\src\fftw\lib) to the unpacked libraries.

Per direction in the fftw files Readme.txt and readme-windows file, run the following commands:

lib /machine: x86 /def:libfftw3f-3.def

lib /machine:x86 /def:libfftw3-3.def

lib /machine:x86 /def:libfftw3l-3.def

to create the static link libraries and the program database files. Move the binaries (.dll, .lib, .pdb, and .xxx) to c:\lib

Using Visual Studio in Windows there are options to build either 32bit or 64bit programs and with or without debugging. To build for 64 bit, use the command (/machine:x64). The program started as a 32 bit program and I have put the 32 bit libraries in c:\lib. There is no reason why it should not be compiled as 64 bit other than that the libraries originally weren’t built as 64 bit libraries. I put 64 bit libraries in c:\lib\x64.

Only the float version of the libraries (as opposed to double and long precision) are used in the RadarRTP.

Copy the include file, fftw3.h to c:\include.

1. Download portaudio from <http://www.portaudio.com/download.html>

The most recent version seems to work well. At the current time, this is pa\_stable\_v190600\_20161030.tgz

Unpack the tgz file into c:\src\portaudio.

The original install used the ASIO low-latency audio driver. I now consider this optional. SO, optionally, download the low latency ASIO software development kit from Steinberg: <https://www.steinberg.net/en/company/developers.html>

To download this you will need to return the license agreement.

Place the ASIOSDK into C:\src\portaudio\src\hostapi\asio

Change to directory build/msvc and use VS2017 to open portaudio.sln. Build the libraries using Visual Studio with the desired debug/architecture/library options. The current radar solution uses only the static release libraries. Linking with the debug version produces voluminous portaudio internals status information that I didn’t find particularly useful and quickly moved away from using the debug library version.

I built and used the Release and Debug versions for Win32, creating both static (.lib) and dynamic (.dll) libraries. I added a d to the end of the debug version filenames and moved to c\lib .

Copy the following include files to c:\include

pa\_asio.h  
pa\_win\_ds.h  
pa\_win\_wasapi.h  
pa\_win\_waveformat.h  
pa\_win\_wdmks.h  
pa\_win\_wmme.h  
portaudio.h  
If ASIO is not downloaded then there will not be a pa\_asio.h file and a compiler preprocessor directive to disable that portion of the code must be used. The directive is PA\_USE\_ASIO=1, or PA\_USE\_ASIO=0.

1. For saving raw data as audio files, download prebuilt binaries for libsndfile from:

<https://github.com/bastibe/libsndfile-binaries>. Unpack into c:\src\libsndfile-1.0.28\lib\libsndfile-binaries-master.

Open a VS2017 command prompt as before and CD to the appropriate directory. Run the librarian to build the static link library from the dynamic library. I get a few warnings when I do, but have not found a problem in using the libraries. This is a screen capture:

C:\src\libsndfile-1.0.28\lib\libsndfile-binaries-master>lib /machine:x86 /def:libsndfile32bit.dll

Microsoft (R) Library Manager Version 14.10.25019.0

Copyright (C) Microsoft Corporation. All rights reserved.

libsndfile32bit.dll : warning LNK4017: MZ statement not supported for the target platform; ignored

libsndfile32bit.dll : warning LNK4017: $ statement not supported for the target platform; ignored

Creating library libsndfile32bit.lib and object libsndfile32bit.exp

Download the source for libsndfile from: <https://github.com/erikd/libsndfile>

Copy the include files to c:\include :

sndfile.h and sndfile.hh

1. If inih is not included with the radar source then it needs to be downloaded. Download inih from <https://github.com/benhoyt/inih>. Place it in the source subdirectory. One of the projects in the Visual Studio solution is to build the inih as a static library. This would be better to accomplish as a separate Visual Studio solution rather than being embedded in the radar solution.
2. Copy the RadarRTP sources and directory structure to c:\src\RadarRTP
3. Copy the signal processing windows from Executable to the SPWindows subdirectory under c:\data
4. Copy the radar initialization file (radarconfig.ini) to c:\data (this can be put in the executable directory and edited to change all of the defaults mentioned here from c:\data to whatever you like.)
5. Octave/matlab scripts to process raw data and create the signal processing window files are provided. I keep these in c:\data . To use them either Octave or matlab needs to be installed.
6. Start the program by double-clicking on RadarRTP.exe in the Executable folder.

1. DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

   This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

   © 2022 Massachusetts Institute of Technology.

   The software/firmware is provided to you on an As-Is basis

   Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work. [↑](#footnote-ref-1)