pyGFC – A Python Extension to the C++ Geodesy Foundation Classes

by Binh Q. Nguyen
NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer’s or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.
pyGFC – A Python Extension to the C++ Geodesy Foundation Classes

Binh Q. Nguyen
Computational and Information Sciences Directorate, ARL
**pyGFC – A Python Extension to the C++ Geodesy Foundation Classes**

**Abstract**

This report describes the results of the development of the pyGFC module, including the multi-step procedure and the implemented computer code. The pyGFC module is a Python extension to the C++ Geodesy Foundation Class, which has been used in the range model of the Mobile Ad-hoc Network (MANET) Emulation (MANE) software system that enables the dynamic connectivity of a MANET system in the Wireless Emulation Laboratory of the U.S. Army Research Laboratory (ARL). The pyGFC module was created to support the visualization of network topologies using the ARL Topodef tool, a graphical design and animation tool for custom-designing and editing a mobility scenario to create specific network topologies.

**Subject Terms**

Python extension, GFC, geodetic distance calculation
Contents

List of Figures iv
List of Tables iv

1. Introduction 1
   1.1 Background ................................................................. 1
   1.2 Scope ........................................................................ 2
   1.3 Method ........................................................................ 2

2. The pyGFC Module Development 3
   2.1 Platform Identification and Tool Versions.................. 3
   2.2 Files and Directories .................................................... 3
   2.3 Procedure ................................................................ 4
   2.4 Result & Discussion .................................................... 6
   2.5 Other Development Tools for Creating Python Extensions to C++ 8

3. Summary 9

4. References 10

Appendix A. URL of Development Tools 11

Appendix B. Source Files 13

Disclaimer 18

Distribution List 19
List of Figures

Figure 1. Procedure for creating the pyGFC module......................................................5
Figure 2. Creating the GFC library ..................................................................................5
Figure 3. Generating wrapper files. ..................................................................................6
Figure 4. Compiling the wrapper files. ............................................................................6
Figure 5. Creating the shared library. ..............................................................................6
Figure 6. Results – an interactive session with the pyGFC module. ..............................8

List of Tables

Table B-1. The pyGFC.h file .............................................................................................13
Table B-2. The pyGFC.cpp file .......................................................................................14
Table B-3. The pyGFC.i file ............................................................................................17
1. Introduction

1.1 Background

The Wireless Emulation Laboratory (WEL) of the U.S. Army Research Laboratory (ARL) exists to support research in wireless mobile networks and mobile ad-hoc network (MANET) security (1) and to demonstrate ARL-developed solutions. A key component of the WEL is an emulation test bed, which is based on the Mobile Ad-hoc Network (MANET) Emulator (MANE) architecture that was originally developed by the U.S. Naval Research Laboratory (NRL) (2). The MANE system consists of hardware and software. The hardware system is a scalable high-performance network of physical computers. The software system enables the emulation of a dynamic MANET by managing and executing the effects of mobility on network connectivity. Effectively it creates virtual network topologies that evolve with time, a dependent variable of mobility.

Mobility is about changes of the geodetic positions of the participating mobile nodes over a fixed time interval, i.e., a mobility scenario. The geodetic positions of each node are predetermined, arranged in ascending order by time, and stored in a log file, which is basically a text file containing the mobility traces of a mobile node. The log files are usually created off-line using the ARL Topodef™ tool (3), a graphical system and method for visually creating and editing specific network topologies of an emulated MANET. A network topology is a snapshot in time showing the number of participating nodes, their relative geographical positions on the screen, and their interconnections or communication links. A custom-designed network topology consists of a predetermined number of nodes being placed at exact locations and a desirable set of communication links.

A communication link between a pair of any two nodes depends on the range of their equipped radios. A link is established whenever one of the radios can intelligibly receive the signal transmitted by the other radio and deconstructed if it cannot. When both can intelligibly receive signals from one another’s radio, a bidirectional link is established. The range is now computed using the values of their radio parameters over the horizontal distance (line-of-sight) between two nodes. The radio parameters include, but are not limited to, radio frequency, transmitting power, and receiving sensitivity. Computing the range is performed by the range model (RM) component of the MANE software system. The RM component relies on the Geodesy Foundation Classes (GFC) (4) to calculate the geodetic distances that are needed in the determination of the network connectivity between a pair of two nodes.

The ARL Topodef™ tool also calculates the range between a pair of every two nodes to determine a possible communication link, which is necessary for the creation of a network topology. The tool then extracts the time-dependent geodetic positions of each node from the
generated network topologies and records them in the log files. Providing the MANE software system with these log files should recreate the same network topology in the test bed as they have been designed and verified. Therefore, having consistent results of geodesic distances calculated in the tool and in the MANE software system is imperative for a successful emulation of a dynamic MANET as intended.

To achieve this objective, the same algorithm and its implementation for calculating the distance between two geodetic locations should be used in the ARL Topodef™ tool and in the MANE system. The implementation of this solution has two options: (1) selecting, implementing, and integrating an appropriate algorithm into the tool and in the MANE system, or (2) using an implementation that has already been integrated in the MANE system. Between the two options, the latter requires lesser effort by leaving the MANE software system intact and by creating a Python (5) extension to the GFC library, which consists of C++ classes.

1.2 Scope

This report describes the successful development of the pyGFC module, a Python extension that enables the integration and use of C++ classes and methods in a Python application, the ARL Topodef™ tool. The intended purposes of this report are to:

- Describe and discuss the method used in the development of the pyGFC module
- Identify other development tools and methods for creating Python extensions to C++
- Document the application programming interfaces of the pyGFC module

The rest of this report is organized in the order of its intended purposes. The next section, section 2, presents and discusses the method that was used to create the pyGFC module and briefly describes other development tools and methods for creating Python extensions to C++ libraries. Section 3 summarizes the findings and concludes the report.

1.3 Method

The development of the pyGFC module used software development tools that came with the Red Hat Enterprise operating system and on the information, tutorials, and usage instructions that were available on the Web. An objective of the development was to minimize the development effort, the modification to the host environment, and the costs of administrative overhead associated with the acquisition, installation, and configuration of additional software systems.
2. The pyGFC Module Development

- The pyGFC module was created on a computer running the Red Hat Enterprise Linux® operating system. All the tools needed for the creation of the module were already available in the host: (i) the GFC source code, (ii) the SWIG command, and (iii) the ar archive program, and (iv) the C++ compiler.

- The GFC files came with the MANE software system, and the last two were already included in the Linux OS packages. The SWIG command whose name was derived from its longer name reflecting its functionality: the Simplified Wrapper and Interface Generator (SWIG) development tool. SWIG facilitates the use of C/C++ functionality from other languages among them is the Python programming language. It is usually included in Linux and Cygwin (6) distributions and also available separately for downloading and installation in other operating environments, e.g., Microsoft Windows® operating systems and Minimalist GNU for Windows (MinGW) (7).

- The ar archive program was used to organize and combine multiple files into a single file, a library. The C++ compiler was used to translate the C++ source files into object code and to create the shared library needed by the pyGFC module.

2.1 Platform Identification and Tool Versions

The information about the platform and the version of the tool was retrieved using the following commands and options that were typed into a console window.

/bin/uname -r -o 2.6.18-8.el5 GNU/Linux
/usr/bin/c++ --version c++ (GCC) 4.1.1 20070105 (Red Hat 4.1.1-52)
/usr/bin/ar --version GNU ar 2.17.50.0.6-2.el5 20061020
/usr/bin/swig --version SWIG Version 1.3.29

The version of the GFC files was embedded in the source files, and they were the same although they had different modification time:

- CEarth.cpp Revision: 1.1.1.1 Modtime: 9/01/98 9:56p
- CEarthCoordinate.cpp Revision: 1.1.1.1 Modtime: 2/07/98 10:34a
- CPolarCoordinate.cpp Revision: 1.1.1.1 Modtime: 2/07/98 10:35a

2.2 Files and Directories

The development of the pyGFC module using the SWIG command required the creation of the following files:
• *The pyGFC.cpp file and its corresponding header file, pyGFC.h.* The files define the GFCCoord class interfacing directly with the GFC library to calculate geodetic distances and providing Python applications a way to use geodetic computing services. Although these two files were not required by SWIG, they were custom-created to support the use of the GFC library in the ARL Topodef™ tool and other Python applications that dealt with geodetic coordinates.

• *The pyGFC.i interface file.* The contents of the pyGFC.i file specify the required header files for the creation of the wrapper file and expose all the functions of the GFC library, the GFCCoord class, and the error functions that were available in the standard C math library, but excluded from the Python math module. The SWIG command then followed the specifications in the pyGFC.i file to create two files: a Python file and a C++ file. The C++ file wraps the GFC library, and the Python file provides transparent functionality of the GFC library in Python applications.

The three files were placed in the pygfc directory, which also had the GFC subdirectory containing the original GFC source code as shown below:

```
pygfc
  pyGFC.i
  pyGFC.h
  pyGFC.cpp
GFC
  CMath.inl
  CMath.hpp
  CEarth.hpp
  GFC.h
  CPolarCoordinate.cpp
  CEarthCoordinate.cpp
  CEarth.cpp
```

2.3 Procedure

The procedure for creating the pyGFC module consisted of five steps. Each step produced one or more files, which were then needed for subsequent steps. Figure 1 illustrates the procedure by showing connected block diagrams of processes and their required input and output files. At the end, only two files are needed by a Python application: the shared library file _pyGFC.so and the Python wrapper file pyGFC.py.
Figure 1. Procedure for creating the pyGFC module.

- Step 1: Creating the GFC library using the C++ compiler and the ar commands:

```
- cd GFC
- c++ -I -I/usr/include -c CEarth.cpp
- c++ -I -I/usr/include -c CEarthCoordinate.cpp
- c++ -I -I/usr/include -c CPolarCoordinate.cpp
- ar -rcv libgfc-linux.a CEarth.o CEarthCoordinate.o CPolarCoordinate.o
- rm -f CEarth.o CEarthCoordinate.o CPolarCoordinate.o
- cd ..
```

Figure 2. Creating the GFC library.

- Step 2: Creating the pyGFC.cpp file, its corresponding pyGFC.h file, and the pyGFC.i interface file using a text editor, e.g., vi. The pyGFC.i file contains the directives for the SWIG command: the header files that are needed by the wrapper file and the interfaces that are made available to a Python application. In this development, the interfaces are specified in the pyGFC.h, which is then included in the pyGFC.i. Appendix B lists the source code of these files.

- Step 3: Creating wrapper files using the SWIG command. SWIG took in the pyGFC.i file and produced two files: the pyGFC.py and the pyGFC_wrap.cxx files.
Figure 3. Generating wrapper files.

- Step 4: Compiling the pyGFC\_wrap.cxx and the pyGFC.cpp files using:

```
$ c++ pyGFC\_wrap.cxx pyGFC.cpp -I/usr/local/include/python2.5 -I./GFC -I. -c
```

Figure 4. Compiling the wrapper files.

- Step 5: Creating the pyGFC library using the C++ compiler:

```
$ c++ pyGFC.o pyGFC\_wrap.o -L./GFC -lgfc\_linux -L/usr/local/lib/python2.5/config -lpython2.5 -shared -o _pyGFC.so
```

Figure 5. Creating the shared library.

### 2.4 Result & Discussion

The development of the pyGFC module was straightforward and relatively smooth because the interfaces to the GFC library were simple, and throughout the process, only a minor syntactical error in the SWIG-generated pyGFC\_wrap.cxx file was encountered and easily fixed. The following lines show the errors and how they were fixed:

```
$ c++ pyGFC\_wrap.cxx pyGFC.cpp -I/usr/local/include/python2.5 -I./GFC -I. -c
pyGFC\_wrap.cxx: In function ‘int SWIG\_Python\_ConvertFunctionPtr(PyObject*, void**, swig\_type\_info*)‘:
  pyGFC\_wrap.cxx:2051: error: invalid conversion from ‘const char*’ to ‘char*’
pyGFC\_wrap.cxx: In function ‘void SWIG\_Python\_FixMethods(PyMethodDef*, swig\_const\_info*, swig\_type\_info**, swig\_type\_info**)‘:
  pyGFC\_wrap.cxx:7961: error: invalid conversion from ‘const char*’ to ‘char*’
```

These two errors that occurred in line numbers 2051 and 7961 were of the same category. Fixing them required the use of a text editor to insert the expression `(char *)` to the lines that caused the error as shown in the following two lines:

```c
2051  char *doc = (char *)(((PyCFunctionObject *)obj) -> m_ml -> ml_doc);
7961  char *c = (char *) methods[i].ml_doc;
```
Once the problem had been fixed, the process continued without a hitch. The final phase was to test the newly created pyGFC module in a Python development environment, the Python’s Integrated Development Environment (IDLE). Figure 6 is the screen dump of a Python interactive session that imported the pyGFC module, showing four different actions performed to ensure the operability of the module:

- Displaying the contents of the pyGFC module. All the C++ classes defining the GFC library are available as they are listed on the screen, e.g., CEarth, CEarthCoordinate, CPolarCoordinate.

- Ensuring that the erf and the erfc functions would work correctly. The sum of erf(x) and erfc(x) equals 1.0 because erfc(x) = 1 – erf(x) by definition.

- Displaying the attributes and methods of the GFCCoord class; e.g., alt, lat, lon, get_distance.

- Setting two geodetic locations and calculating the distance between them.
2.5 Other Development Tools for Creating Python Extensions to C++

In addition to the SWIG development tool, several other open-source tools for creating Python extensions are available on the Web. They include but are not limited to Boost.Python, PyRex, PyCXX, and Weave. A brief description of what they are is included in this section. More detailed description and usage instructions can be obtained from their respective Web sites (appendix A).
• *Boost.Python* emerges as a competent development tool for the development of Python extensions to C/C++. Using *Boost.Python* requires extensive knowledge of the C++ programming language, especially the C++ templates.

• *PyCXX* appears to be a capable development tool whose principal goal is to ease the writing of Python extensions.

• *PyRex* enables the mixing of Python and C/C++ in a Python program using its own language which is very similar to Python and C; therefore, *PyRex* itself is a computer language.

• *Weave* also enables the mixing C/C++ code in a Python program to improve its performance and to generate Python extensions.

3. **Summary**

The creation of the pyGFC module using SWIG was relative easy on a Linux® environment because the interfaces to the GFC library were simple. The described multi-step procedure is expected to reproduce the same results in a Linux-like environment; for example, Cygwin and MSYS/MinGW environments.
4. References


## Appendix A. URL of Development Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWIG</td>
<td><a href="http://www.swig.org">http://www.swig.org</a></td>
</tr>
<tr>
<td>PyRex</td>
<td><a href="http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex">http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex</a></td>
</tr>
<tr>
<td>Boost.Python</td>
<td><a href="http://www.boost.org">http://www.boost.org</a></td>
</tr>
<tr>
<td>PyCXX</td>
<td><a href="http://sourceforge.net/projects/cxx/">http://sourceforge.net/projects/cxx/</a></td>
</tr>
<tr>
<td>Weave</td>
<td><a href="http://www.scipy.org/Weave">http://www.scipy.org/Weave</a></td>
</tr>
</tbody>
</table>
INTENTIONALLY LEFT BLANK.
Appendix B. Source Files

Table B-1. The pyGFC.h file.

```c
#ifndef __PYGFC__
#define __PYGFC__

#include <iostream>
#include "GFC.h"

using namespace std;

#define ONE_MINUTE METERS 1852.0
#define ONE DEGREE METERS (60.0 * ONE_MINUTE METERS)
#define COORD STRLEN 64

class GFCCoord {
public:
    GFCCoord( const GFCCoord * );
    GFCCoord(double lat=0.0, double lon=0.0, double alt=0.0);
    virtual ~GFCCoord();
    void set(const GFCCoord *);
    void set(double , double , double);
    void print_coords();
    char * get_coords_str();
    double get_distance(const GFCCoord *);
    double get_distance(const GFCCoord *, const GFCCoord *);
    double estimate_latitude_coord( GFCCoord *, double);
    double estimate_longitude_coord( GFCCoord *, double);
    GFCCoord * estimate_southeast_coords(double, double);
    double lat;
    double lon;
    double alt;

private:
    CEarth * earth;
    char * cstr;
};

#endif
```
Table B-2. The pyGFC.cpp file.

```cpp
#include "pyGFC.h"

GFCCoord::GFCCoord(double lat, double lon, double alt)
{
  this->lat = lat;
  this->lon = lon;
  this->alt = alt;
  this->earth = new CEarth();
  this->cstr = (char *)malloc(COORD_STRLEN);
}

GFCCoord::GFCCoord( const GFCCoord * p )
{
  this->lat = p->lat;
  this->lon = p->lon;
  this->alt = p->alt;
  this->earth = new CEarth();
  this->cstr = (char *)malloc(COORD_STRLEN);
}

GFCCoord::~GFCCoord()
{
  delete this->earth;
  delete this->cstr;
}

void GFCCoord::set(const GFCCoord * p)
{
  this->lat = p->lat;
  this->lon = p->lon;
  this->alt = p->alt;
}

void GFCCoord::set(double lat, double lon, double alt)
{
  this->lat = lat;
  this->lon = lon;
  this->alt = alt;
}

void GFCCoord::print_coords()
{
  cout << "(";
    << this->lat << ",";
    << this->lon << ",";
    << this->alt
    << ")";
  << endl;
}

double GFCCoord::get_distance(const GFCCoord *a, const GFCCoord *b)
{
  /****************************************************************
   * This code is taken from the NRL-designed RangeDrop::getDistance()
   */
  CPolarCoordinate here, there;
  here.SetUpDownAngleInDegrees(a->lat);
  here.SetLeftRightAngleInDegrees(a->lon);
  here.SetDistanceFromSurfaceInMeters(a->alt);
  there.SetUpDownAngleInDegrees(b->lat);
  there.SetLeftRightAngleInDegrees(b->lon);
  there.SetDistanceFromSurfaceInMeters(b->alt);
  return this->earth->GetLineOfSightDistance(here,there);
}

char *GFCCoord::get_coords_str()
{
  sprintf(this->cstr, "%.8f, %.8f, %8f", this->lat, this->lon, this->alt);
  return this->cstr;
}

double GFCCoord::get_distance(const GFCCoord * p)
{
  return this->get_distance(this, p);
}

double GFCCoord::estimate_latitude_coord( GFCCoord *p, double target_distance 

{
 /*** estimate the latitude that is SOUTH of where p is */
 CPolarCoordinate here;
 here.SetUpDownAngleInDegrees(p->lat);
 here.SetLeftRightAngleInDegrees(p->lon);
 here.SetDistanceFromSurfaceInMeters(p->alt);
 GFCCoord *e = new GFCCoord( p );

 /* estimate the latitude then adjust it later. */
 e->lat = p->lat - target_distance/(double)ONE_DEGREE_METERS;
 CPolarCoordinate there;
 there.SetUpDownAngleInDegrees( e->lat );
 there.SetLeftRightAngleInDegrees( e->lon );
 there.SetDistanceFromSurfaceInMeters( e->alt );

 /* dlat = value used for incrementally searching for the closest value */
 double dlat = -0.010/ONE_DEGREE_METERS;
 double est_dist = this->earth->GetLineOfSightDistance(here,there);
 if ( est_dist > target_distance ) dlat = -1.0 * dlat;

 while ( est_dist < target_distance )
 {
 e->lat += dlat;
 there.SetUpDownAngleInDegrees(e->lat );
 est_dist = earth->GetLineOfSightDistance(here,there);
 }
 return e->lat;
 }

double GFCCoord::estimate_longitude_coord( GFCCoord *p, double target_distance)
{
 /*** estimate the longitude that is EAST of where p is */

 CPolarCoordinate here;
 here.SetUpDownAngleInDegrees(p->lat);
 here.SetLeftRightAngleInDegrees(p->lon);
 here.SetDistanceFromSurfaceInMeters(p->alt);
 GFCCoord *e = new GFCCoord( p );

 /* estimate the longitude then adjust it later. */
 e->lon = p->lon + target_distance/(double)ONE_DEGREE_METERS;

 CPolarCoordinate there;
 there.SetUpDownAngleInDegrees(e->lat );
 there.SetLeftRightAngleInDegrees(e->lon );
 there.SetDistanceFromSurfaceInMeters(e->alt );

double dlon = 0.010/ONE_DEGREE_METERS;
 double est_dist = this->earth->GetLineOfSightDistance(here,there);
 if ( est_dist > target_distance ) dlon = -1.0 * dlon;

 while ( est_dist < target_distance )
 {
 e->lon += dlon;
 there.SetLeftRightAngleInDegrees(e->lon );
 est_dist = earth->GetLineOfSightDistance(here,there);
 }
 return e->lon;
 }

GFCCoord * GFCCoord::estimate_southeast_coords(double width, double height)
{
 /* estimate and return the SE corner given w, h in meters
 width & height should be less than 20 km.
 caveats: this implementation does not handle any location on earth. */

 CPolarCoordinate here;
 here.SetUpDownAngleInDegrees(this->lat);
 here.SetLeftRightAngleInDegrees(this->lon);
 here.SetDistanceFromSurfaceInMeters(this->alt);
 GFCCoord *e = new GFCCoord( this );

 /* estimate the latitude then adjust it later. */
 e->lat = this->lat - height/ONE_DEGREE_METERS;
 CPolarCoordinate there;
 there.SetUpDownAngleInDegrees( e->lat );
 there.SetLeftRightAngleInDegrees( e->lon );
 there.SetDistanceFromSurfaceInMeters( e->alt );
}
/* dlat = value used for incrementally searching for the closest value */
double dlat = -0.010/ONE_DEGREE_METERS;
double est_dist = this->earth->GetLineOfSightDistance(here,there);
if ( est_dist > height ) dlat = -1.0 * dlat;

while ( est_dist < height )
{
    e->lat += dlat;
    there.SetUpDownAngleInDegrees(e->lat);
    est_dist = earth->GetLineOfSightDistance(here,there);
}

double saved_lat = e->lat;
e->lat = this->lat;
there.SetUpDownAngleInDegrees(e->lat);
e->lon = this->lon + width/ONE_DEGREE_METERS;
double dlon = 0.010/ONE_DEGREE_METERS;
est_dist = this->earth->GetLineOfSightDistance(here,there);
if ( est_dist > width ) dlon = -1.0 * dlon;

while ( est_dist < width )
{
    e->lon += dlon;
    there.SetLeftRightAngleInDegrees(e->lon);
    est_dist = earth->GetLineOfSightDistance(here,there);
}
e->lat = saved_lat;
return e;
Table B-3. The pyGFC.i file.

```c
#define pyGFC
{%
  #include "pyGFC.h"
%
#include "GFC.h"
#include "pyGFC.h"

// error functions from the math lib.
extern double erf(double x);
extern double erfc(double x);
extern float erff(float x);
extern float erfcf(float x);
```
Disclaimer

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents. Citation of manufacturer’s trade names does not constitute an official endorsement or approval of the use thereof.
<table>
<thead>
<tr>
<th>NO OF COPIES</th>
<th>ORGANIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADMNSTR ELEC DEFNS TECHL INFO CTR</td>
</tr>
<tr>
<td></td>
<td>ATTN DTIC OCP</td>
</tr>
<tr>
<td></td>
<td>8725 JOHN J KINGMAN RD STE 0944</td>
</tr>
<tr>
<td></td>
<td>FT BELVOIR VA 22060-6218</td>
</tr>
<tr>
<td>1</td>
<td>DARPA</td>
</tr>
<tr>
<td></td>
<td>ATTN IXO S WELBY</td>
</tr>
<tr>
<td></td>
<td>3701 N FAIRFAX DR</td>
</tr>
<tr>
<td></td>
<td>ARLINGTON VA 2203-1714</td>
</tr>
<tr>
<td>1</td>
<td>CD</td>
</tr>
<tr>
<td></td>
<td>OFC OF THE SECY OF DEFNS</td>
</tr>
<tr>
<td></td>
<td>ATTN ODDRE (R&amp;AT)</td>
</tr>
<tr>
<td></td>
<td>THE PENTAGON</td>
</tr>
<tr>
<td></td>
<td>WASHINGTON DC 20301-3080</td>
</tr>
<tr>
<td>1</td>
<td>US ARMY RSRCH DEV AND ENGRG CMND</td>
</tr>
<tr>
<td></td>
<td>ARMAMENT RSRCH DEV AND ENGRG CTR</td>
</tr>
<tr>
<td></td>
<td>ARMAMENT ENGRG AND TECHNLGY CTR</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD AAR AEF T J MATTS</td>
</tr>
<tr>
<td></td>
<td>BLDG 305</td>
</tr>
<tr>
<td></td>
<td>ABERDEEN PROVING GROUND MD 21005-5001</td>
</tr>
<tr>
<td>1</td>
<td>US ARMY TRADOC</td>
</tr>
<tr>
<td></td>
<td>BATTLE LAB INTEGRATION &amp; TECHL DIRCRT</td>
</tr>
<tr>
<td></td>
<td>ATTN ATCD B</td>
</tr>
<tr>
<td></td>
<td>10 WHISTLER LANE</td>
</tr>
<tr>
<td></td>
<td>FT MONROE VA 23651-5850</td>
</tr>
<tr>
<td>1</td>
<td>PM TIMS, PROFILER (MMS-P)</td>
</tr>
<tr>
<td></td>
<td>AN/TMQ-52</td>
</tr>
<tr>
<td></td>
<td>ATTN B GRIFFIES</td>
</tr>
<tr>
<td></td>
<td>BUILDING 563</td>
</tr>
<tr>
<td></td>
<td>FT MONMOUTH NJ 07703</td>
</tr>
<tr>
<td>1</td>
<td>US ARMY INFO SYS ENGRG CMND</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSEL IE TD F JENIA</td>
</tr>
<tr>
<td></td>
<td>FT HUACHUCA AZ 85613-5300</td>
</tr>
<tr>
<td>1</td>
<td>COMMANDER US ARMY RDECOM</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD AMR</td>
</tr>
<tr>
<td></td>
<td>W C MCCORKLE</td>
</tr>
<tr>
<td></td>
<td>5400 FOWLER RD</td>
</tr>
<tr>
<td></td>
<td>REDSTONE ARSENAL AL 35898-5000</td>
</tr>
<tr>
<td>1</td>
<td>US GOVERNMENT PRINT OFF DEPOSITORY RECEIVING SECTION</td>
</tr>
<tr>
<td></td>
<td>ATTN MAIL STOP IDAD J TATE</td>
</tr>
<tr>
<td></td>
<td>732 NORTH CAPITOL ST NW</td>
</tr>
<tr>
<td></td>
<td>WASHINGTON DC 20402</td>
</tr>
<tr>
<td>1</td>
<td>US ARMY RSRCH LAB</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL CI OK TP</td>
</tr>
<tr>
<td></td>
<td>TECHL LIB T LANDFRIED</td>
</tr>
<tr>
<td></td>
<td>BLDG 4600</td>
</tr>
<tr>
<td></td>
<td>ABERDEEN PROVING GROUND MD 21005-5006</td>
</tr>
<tr>
<td>1</td>
<td>DIRECTOR US ARMY RSRCH LAB</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL RO EV</td>
</tr>
<tr>
<td></td>
<td>W D BACH</td>
</tr>
<tr>
<td></td>
<td>PO BOX 12211</td>
</tr>
<tr>
<td></td>
<td>RESEARCH TRIANGLE PARK NC 27709</td>
</tr>
<tr>
<td>1</td>
<td><a href="mailto:SAM_BLACKBURN@POBOX.COM">SAM_BLACKBURN@POBOX.COM</a> ELEC</td>
</tr>
<tr>
<td>8</td>
<td>US ARMY RSRCH LAB</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL CN G RACINE</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL CN NT</td>
</tr>
<tr>
<td></td>
<td>B NGUYEN</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL CN NT B RIVERA</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL CN NT N IVANIC</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL CN NT R HARDY</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL CN OK PE</td>
</tr>
<tr>
<td></td>
<td>TECHL PUB</td>
</tr>
<tr>
<td></td>
<td>ATTN AMSRD ARL CN OK TL</td>
</tr>
<tr>
<td></td>
<td>TECHL LIB</td>
</tr>
<tr>
<td></td>
<td>ATTN IMNE ALC IMS</td>
</tr>
<tr>
<td></td>
<td>MAIL &amp; RECORDS MGMT</td>
</tr>
<tr>
<td></td>
<td>ADELPHI MD 20783-1197</td>
</tr>
</tbody>
</table>

Total: 20 (2 ELEC, 1 CD, 17 HC)