Naval Research Laboratory

Stennis Space Center, MS 39529-5004

NRL/MR/7440--10-9306

Geotiff to CADRG Conversion Dynamic Link Library Specification Version 1.1

Michael E. Trenchard

Geospatial Sciences and Technology Branch
Marine Geosciences Division

Thomas D. Gray

QinetiQ North America
Stennis Space Center, Mississippi

November 24, 2010

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
24-11-2010

2. REPORT TYPE
Memorandum Report

3. DATES COVERED *(From - To)*

4. TITLE AND SUBTITLE

Geotiff to CADRG Conversion Dynamic Link Library Specification
Version 1.1

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Michael E. Trenchard and Thomas D. Gray*

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
74-6293-00

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7440--10-9306

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
One Liberty Center
875 N. Randolph Street, Suite 1425
Arlington, VA 22203

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*QinetiQ North America, Stennis Space Center, MS 39529

14. ABSTRACT

The Naval Research Laboratory (NRL) was tasked by the Naval Air Systems Command (NAVAIR) on behalf of the Canadian Forces (CF) to develop a dynamic link library that would convert geotiff source files containing mission planning overlays into Compressed Arc Digitized Raster Graphics (CADRG) equivalent files. This dynamic link library will be integrated as part of the Tactical Aircraft Moving Map Unique Planning Component (TUPC) onto the Joint Mission Planning System (JMPS). In 2003, NRL developed the Moving-Map Composer–Personal Computer (MMCPC) software system to support Finnish Air Force mission planning efforts. One MMCPC function, unique to this system, converted source geotiff Finnish map files into CADRG compatible files for display in TAMMAC. Software applicable to this function from MMCPC will be used to support the core functions of this new capability for the CF.

15. SUBJECT TERMS

Geotiff to CADRG Conversion Dynamic Link Library Specification Version 1.1

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	SAR	11	Michael E. Trenchard / (288) 688-4633

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Background

The Naval Research Laboratory (NRL) was tasked by the Naval Air Systems Command (NAVAIR) on behalf of the Canadian Forces (CF) to develop a dynamic link library that would convert geotiff source files containing mission planning overlays into Compressed Arc Digitized Raster Graphics (CADRG) equivalent files. This dynamic link library will be integrated as part of the Tactical Aircraft Moving Map Unique Planning Component (TUPC) onto the Joint Mission Planning System (JMPS). In 2003, NRL developed the Moving-Map Composer - Personal Computer (MMCPC) software system to support Finnish Air Force mission planning efforts. One MMCPC function, unique to this system, converted source geotiff Finnish map files into CADRG compatible files for display in TAMMAC. Software applicable to this function from MMCPC will be used to support the core functions of this new capability for the CF.

Overview

NRL will perform all software development necessary to ensure the resulting dynamic link library is fully compatible with the geotiff format produced by JMPS. Further, NRL will develop specific tools to ensure that the CADRG compatible output is fully compliant with both Raster Product Format (RPF) and National Imagery Transmission Format (NITF) Version 2.0 specifications.

The TUPC is expected to interface smoothly with the NRL-developed data processing routines. Specifically, NRL expects the TAMMAC UPC will 1) manage the paths to both the source geotiffs and the output CADRG compatible files, and 2) provide the dynamic link library with the necessary source map scale information and classification flag of JMPS-generated geotiffs. Once the source paths, source map scale and classification information are provided, the dynamic link library will 1) read the source files, 2) determine the expected number of CADRG compatible output files for a given map scale, and 3) provide that list of files to the TUPC prior to data processing. The TUPC may utilize this listing to verify with the user that the correct files are ready to be processed. Once confirmed via the TUPC, the dynamic link library will process the given set of source geotiffs into CADRG compatible output files. These files will be written to the output path provided by the TUPC.

Requirements

The following list defines the full set of requirements set forth for this software development effort by NRL in support of the CF.

- The conversion software shall exist in the form of a dynamic link library file.

Manuscript approved October 7, 2010.

- The dynamic link library shall provide the capability to convert one or more geotiff file(s) into CADRG format file structure.
- The dynamic link library shall provide the capability to read a directory of geotiff files and return the minimum bounding rectangle that encompasses those files.
- The geotiff source directory shall be selectable for the conversion function.
- The geotiff source directory shall be selectable for the minimum bounding rectangle calculation function.
- The CADRG output directory shall be selectable for the conversion function.
- The output CADRG equivalent file(s) shall be compliant with NITF 2.0 and RPF specifications.
- The scale(s) of the output CADRG equivalent file(s) shall be selectable among the following scales with more than one output scale allowed:
    - 1:12,500
    - 1:50,000
    - 1:100,000
    - 1:250,000
    - 1:500,000
    - 1:1,000,000
    - 1:2,000,000
    - 1:5,000,000
- The classification that shall appear in the NITF header, RPF header and table of contents file shall be selectable among the following:
    - Unclassified
    - Confidential
    - Secret
- The conversion function shall return an error code indicating success or failure.
- The minimum bounding rectangle calculation function shall return an error code indicating success or failure.
- The CADRG color palette shall be dynamically created based on the input geotiff file(s) located in a source directory specified by the TUPC.
- Data compression procedures shall be optimized to reduce color and spatial loss from the source geotiff to the output CADRG equivalent file(s).
- The dynamic link library will provide the option to prohibit processing of partial output CADRG equivalent file(s) (i.e. ‘edge’ files containing a black border). If this option is selected, the partial file(s) will be deleted as part of the data processing procedures.
- The dynamic link library shall provide an estimate of the completion time to convert the source geotiff file(s) to CADRG equivalent format.
- The dynamic link library shall include the ability to use callback functions to provide the calling program with processing status information.

Map Data Processing Procedure

This software development effort utilizes the data processing functions embedded within the MMCPC application to provide the core functionality of the dynamic link library to support the CF. However, since this software development uses geotiff files generated within the JMPS environment, improved methods to color and spatially compress these geotiffs into CADRG equivalent files are needed. In particular, the source geotiff files contain very little redundancy in geospatial data content, so minimizing visually perceptual data loss is critical. CADRG employs a data compression technique known as vector quantization. This is the only data compression technique allowed in the production of output CADRG. Vector quantization compression for CADRG is performed in two steps: 1) color compression and 2) spatial compression. Details of how vector quantization is implemented for CADRG may be found in the CADRG product specification. For color compression, a selection of the most representative 216 colors is required to effectively color compress the data from 24-bit color to 8-bit. Within this software development effort, a *color palette* consisting of the 216 color values needed to perform color compression is automatically generated from a histogram of the unique red-green-blue color values from the source geotiff files found in a given directory. NRL is utilizing internally funded research to map and cluster colors in a more visually linear color space to limit color loss during color compression. Spatial compression also utilizes 3-D color clustering algorithms in this visual color space to again minimize spatial data loss and enhance the readability of the final output CADRG equivalent files.

Dynamic Link Library Interface Specifications

Constructor: Can_CADRG()

Description: Can_CADRG class constructor.

Input Parameters:

char * source: The source path of the geotiff files

char * destination: The destination path of the output CADRG file directory structure.

char num_ scales: The number of scales that output CADRG equivalent files will be created.

char * scales: Array containing the scales that the CADRG equivalent files will be created:

value	scale
0	1:12,500
1	1:50,000
2	1:100,000

3 1:250,000
4 1:500,000
5 1:1,000,000
6 1:2,000,000
7 1:5,000,000

enum Classification security_classification: Security classification. Valid values are unclassified, confidential, secret.

int finalize: Flag to indicate whether partial files (image edges which contain black border space) should be included in the final CADRG equivalent output.

0 = No
1 = Yes

Return:

None

Example:

```
char num_faf_scales;
char faf_scales[NUMBER_FAF_SCALES];
enum Classification security_classification;
char src_path[512];
char dest_path[512];
int finalize;

stcrpy(src_path, “c:\\src”);
strcpy(dest_path, “c:\\dest”);
num_faf_scales = 1;
finalize = 1;
faf_scales[0] = 6; /* 2M */
security_classification = Unclassified;
Can_CADRG my_Can_CADRG(src_path, dest_path, num_faf_scales, faf_scales,
security_classification, finalize);
```

Method: GetSrcExtents()

Description: Reads the spatial extents of all geotiff files in the source path and calculates a global extent for all files. The global extent is passed back to the calling function in the minimum bounding rectangle structure variable. Also, calculates the estimated time in decimal minutes to complete the conversion.

Input Parameters:

struct MBR *mbr: The minimum bounding rectangle (ie. extents) of all geotiff files in the source directory. The memory for this variable needs to be allocated prior to passing in to this method.

float *time estimate: The estimated completion time in decimal minutes. This value will be returned to the calling function.

The minimum bounding rectangle (MBR) structure is defined as:

```
struct MBR
{
    double northLat;
    double southLat;
    double westLon;
    double eastLon;
};
```

Return:

```
int:     0 = Success
        -1 = GENERAL_ERR
        -2 = PFG_LOAD_GTIF_ERR
        -8 = PFG_GET_INFO_ERR
```

Example:

```
…
MBR mbr;
int rc;
float completion_time;
…
…
/* Make sure constructor has been called prior to calling this method. */

rc = my_Can_CADRG. GetSrcExtents(&mbr, &completion_time);
```

.

Method: ConvertToCADRG()

Description: Main function to convert geotiff files to CADRG equivalent format.

Input Parameters:

void(*funct_ptr)(char *): A pointer to a callback function which is used to provide processing status information. The callback function will need to be implemented by the client software that calls the dll.

Return:

```
int:     0 = SUCCESS
        -1 = GENERAL_ERR
        -2 = PFG_LOAD_GTIF_ERR
        -3 = GET_RBW_COMPRESSED_ERR
        -4 = RPF_INFO_FILE_ERR
        -5 = CADRG_PROCESS_GTIF_ERR
        -6 = COMPRESS_ALL_PARTIALS_ERR
        -7 = FINALIZE_PROCESSING_ERR
```

Example:

```
void status_callback(char * status);    /* prototype */
…
…
void (*funct_ptr)(char*);
int rc;
…
…
funct_ptr = &status_callback;

/* Make sure constructor has been called prior to calling this method. */
rc = my_Can_CADRG.ConvertToCADRG(funct_ptr);
```

Calling Function Example

The following demonstrates the general flow of creating the conversion object, getting the extents of the source geotiff file(s) and estimated completion time, and calling the conversion method:

/* prototypes */

```
void status_callback(char * status);
```

/* variable declarations */

```
char num_faf_scales;
char faf_scales[NUMBER_FAF_SCALES];
```

```
enum Classification security_classification;
char src_path[512];
char dest_path[512];
int rc;
MBR mbr;
int finalize;
float                                          completion_time;
void (*funct_ptr)(char*);
```

/* populate the variables that will be used to call the class constructor */

```
stcrpy(src_path, “c:\\src”);
strcpy(dest_path, “c:\\dest”);
num_faf_scales = 1;
faf_scales[0] = 6; /* 2M */
security_classification = Unclassified;
finalize = 1;
```

/* create an instance of the class by calling the constructor */

```
Can_CADRG my_Can_CADRG(src_path, dest_path, num_faf_scales, faf_scales,
security_classification, finalize);
```

/* optionally, get the extents of the source image(s) and estimated completion time. This step is not necessary to perform the actual conversion. */

```
rc = my_Can_CADRG. GetSrcExtents(&mbr, &completion_time);
```

/* call the method to perform the conversion */

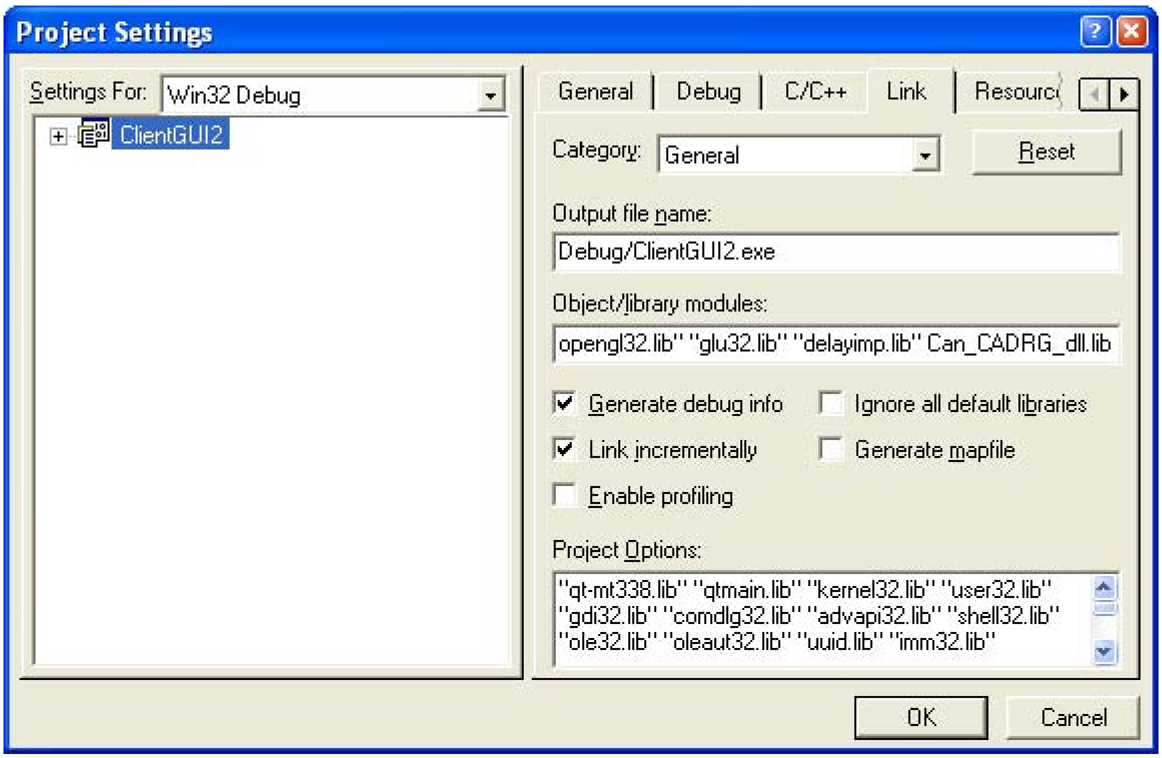
```
funct_ptr = &status_callback;
rc = my_Can_CADRG.ConvertToCADRG(funct_ptr);
```

/* ***example definition of the callback function (to be implemented by calling client program) */***

```
void status_callback(char * status)
{
   printf("%s", status);
   return;
}
```

Compile Instructions

The Can_CADRG.h header file must be included in the source code directory or in an include directory that exists for the project. For Microsoft Visual Studio projects, the Can_CADRG_dll.lib must reside in the project directory of the program that will be utilizing it, and must be added to the project settings. For Visual Studio 6, the project settings can be opened by clicking Alt+F7. Select the Link tab and add Can_CADRG_DLL.lib to the Object/library modules text box and then compile normally.



Acknowledgements

The map data processing research work was supported by Program Element 602435N as part of the NRL 6.2 Base Program. The authors wish to thank Ms. Maura Lohrenz for her support under the NRL 6.2 Base Program. Further, the authors wish to thank Mr. Alfredo Fajardo and Ms. Maivan Ly who support the TAMMAC Unique Planning Component software development for their support and collaboration in this joint effort.

References

National Geospatial-Intelligence Agency. Military Standard – Compressed ARC Digitized Raster Graphics, MIL-C-89038. 6 October 1994.

Y. Linde, A. Buzo, and R. M. Gray, ``An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, pp. 702--710, January 1980.

R. M. Gray, ``Vector Quantization," *IEEE ASSP Magazine*, pp. 4--29, April 1984.

S. Myrick, M. Trenchard, et. al., "Moving Map Composer – Personal Computer (MMCPC) Version 1.0 for the Finland Air Force: Software User's Manual", *NRL/FR/7440—08-10,131*, 21 October 2008.

M. Trenchard, M. Lohrenz, et. al., "Color Palette Considerations for Digital Map Displays", *NRL/FR/7441—92-9415*, 3 August 1993.