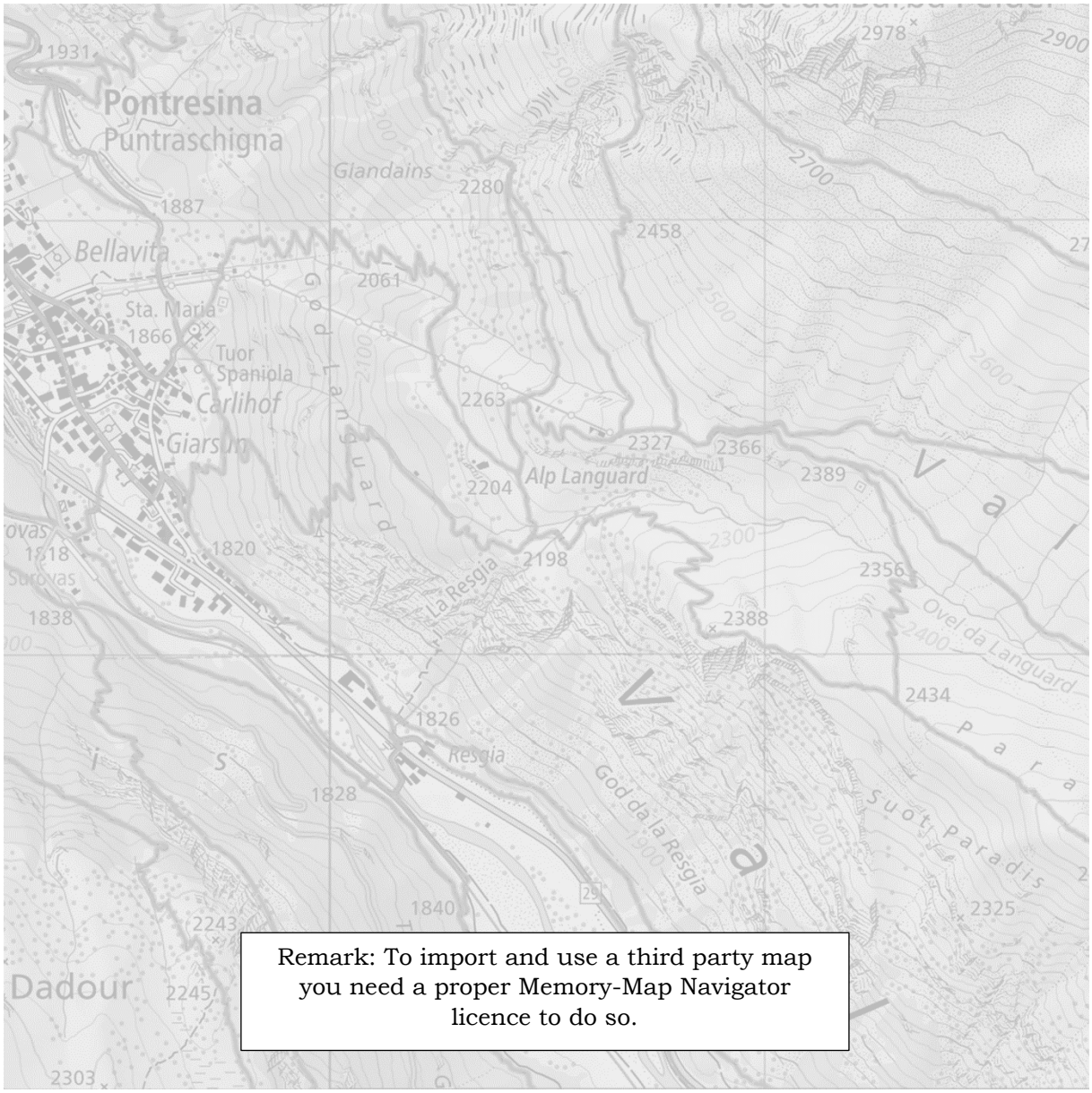
Project Maps for Memory Map (M4MM)
Technical document

By Hans Kok (info@hzns.nl)

Version: 1.0.0 (Pontresina) July 1. 2022



Remark: To import and use a third party map you need a proper Memory-Map Navigator licence to do so.

Change log

This change log covers the whole project; documentation and application.

Version	Date	Remarks
0.6.0 (Zwolle)	August 6.2021	• Full initial version of the project, including the complete documentation and the application. It replaces all earlier versions.
0.7.0 (Woombah)	November 1. 2021	Changes: • Data model change `JprData`: `JprData.sc` (scale) and `JprData.sr` (resolution) become Double precision and added `JprData.cd` (color depth) • Added a `ComboBox` to select a scale in Build forms (like `DpiBox`). • Implementation of scale (`sc`) and resolution(`sr`) as estimated values. • Moving `FileDialog` to `common` module, initiating `FileDialog` when opening a form (if needed) and code changes related to `FileDialog`. • Moving `PrintDialog` to `common` module and code changes related to `PrintDialog`. • Moving `DocumentPrint` to `StarMenu` form and changing related code to `DocumentPrint`. • Created functions to retrieve WKT and Gdal version in `common` module and changed related code in forms. • Changed handling images. Maximized to 2 G Byte of pixel information using error handling instead of file size screening. • Opening a form starts, if necessary, with a file selection dialog. Fixes (major): • Corrected some comment lines in JPR-file (added '//'). • Corrected save routine in Build Ozi Form.
0.8.0 (Blaine)	November 21. 2021	Changes: • Project ported to Visual Studio 2022 Community (Visual Basic) and .NET framework 4.8. • 64 Bit application.
0.9.0 (Whistler)	March 5. 2022	Added: • Added `ITRF92`, `ITRF2008` and `GDA94` to application. • GIMP to Cookbook Changes: • Translation of geographical coordinates to pixel coordinates.
1.0.0 (Pontresina)	July 1. 2022	• Reviewed and updated the Technical document.

Index

Introduction

Purpose of this document

The purpose of this document is to give you

- the detailed background information of the project Map for Memory-Map (M4MM)
- the detailed background information of the application M4MM

Structure of this document

This document follows the line: research, analyse, design, build, test. Although the process to create the project looks linear, in reality it is an iterative one. After an analyse phase the conclusion is often: information is missing and some research must be done. Often it is two step forward and one step back.

About Memory-Map and third party software/applications

In this document all applications will be seen as 'Black boxes', which means you put something in, you get something out. How the application the data processes is out of scoop. Only the result will be observed/used.

Research

In the research stage the first step is to get a clear picture of the way Memory-Map works. This the 'Target side'. In the second step we will try to get the same clear picture of the possible candidate calibrated maps. This is the 'Source side'. The last step is looking for candidate tools which could be used in the process of converting maps. This is the 'Tool side'. This 'backward' method is used to narrow our information.

Target side

1. To learn more about Memory-Map works in relation to calibrating and importing maps we will have a look on the help-files of memory map. You can find them with in the application and on the website Memory of map. The content of both sources are the same. Following the instructions to calibrate a map, Memory-Map will generate a so called JPR-file with all the calibration data.
2. When I bought a set of third party maps it contained PNG-image files with related JPR-files. In the documentation to this set the application Fugawi was mentioned. Both, Fugawi and Memory-Map uses more or less the same JPR-file. At the former website of Fugawi an explanation could be found (JPR file format Version 1.18, August 19, 2010).An older version (1.14) of the document above is found on the website of Path-Away (https://pathaway.com/apps5tools/MTech/English/PWMapConvert.zip). Fugawi doesn't exist anymore. De support stopped at 31. December 2019.
3. In the Memory-Map documentation is mentioned a GeoTIFF-file could be imported.
4. Because there is a glitch in the calibration function of current version of Memory-Map to two versions will be referred (current: 6.4.3 and 5.4.4 (free of the glitch)).

Source side

Strolling the internet four possible sources were identified which could be converted:

1. Maps made for OziExplorer. These maps consists an Image-file and a calibration file (MAP-file). A special case are the maps created by the Mobile Atlas Creator (MOBAC).
2. Maps based on the GeoTIFF-file format.
3. Maps based on the GeoPDF-file format.
4. Maps base on the MrSID-file format.

Maps base on other image-file formats doesn't contain relevant georeferenced data and must calibrated using Memory-Map.

Tool side

For the project four different kind of tools are needed. To qualify the tools must be “free”. This means the needed functionality must be available without a charge.

1. Tools to handle georeferenced files. The main purpose is extracting these data and to manipulate the (graphical-) content. Two relates applications were found, GDAL and QGIS.
2. Tools to handle pixel-orientated images. The main purpose is to reduce the color depth of an image and convert image format the PNG-image format. There are countless options. Most applications has their boundaries, mostly the size of the image. A maximum size of 10,000 by 10,000 pixel may look great but with a scale of 1:25,000 and a resolution of 254 DPI it covers only an area of 25 by 25 kilometer. Some applications produce a color cast during the process, something not wanted. The preferred application is Paint.NET. GIMP is a good alternative. A special case is the conversion of a PDF-File into an image file. The best option is PDF Exchange Editor.
3. Tools to edit text-files. The only purpose is to create or edit an JPR-file. User choice, but select a simple one, the content of a JPR-file is plain text. The application M4MM has a very simple editor on board.
4. A tool to build the application. Original Microsoft Visual Basic 2019/Windows Forms App (.Net framework), a part of Visual Studio Community, was used. In November 2021 the application was ported to Microsoft Visual Basic 2022. The author is familiar with development tool and so it was auteurs “natural” choice.

Analyse

Target side

Memory-Map

What does Memory-Map do

Related to the project Memory-Map provide us with the following functions:

1. Calibrating a map.
2. Adding/editing the boundary to a map.
3. Adding information to a map.
4. Merging maps.
5. Adding other position formats (geographical coordinates) to Memory-Map.
6. Importing a map.

The descriptions of the procedures below main purpose is getting a clear picture of "what is happening" In "The cookbook" you will find more handsome description "How to do".

Calibrating a Map

This procedure start with a map in a graphical format. On the website www.hzns.nl you will find some directions to make one.

The first step is to convert this map to the graphical format PNG with a color depth of 8 bit (255 colors). Memory-map does except a lower color depth but no higher. Memory-Map accepts the TIFF-format too, but the data compression is lower than PNG. Memory-Map doesn't except other graphical formats like JPG, GIF or BMP. This item must be addressed in "The cookbook".

The second step is determine which geographical coordinate system will be used. The Memory-Map manual mentions only geographical coordinates, but other grids are available. If you select another position format (*Mode* > *Position format*) you can use it to calibrate you map. A good option is *Mode* > *Position format* > *Lat, Long* > *Degrees (+/-)*, if you are going to use latitude/Longitude (you can use decimal degrees and you don't have to bother about minutes and seconds).

The third step is loading your map image into Memory-Map. Copy your PNG-file to a folder on your computer where Memory-Map can find it. By opening the *Refresh Map List* window (*Map* > *Map list* > button *Refresh Map List*) you can explore this folder(s). This item will be addressed in "The cookbook".

The fourth step is loading your PNG-file in the main window of Memory-Map. Open Memory-Map and go to the *Refresh Map List* (*Map* > *Map list* > button *Refresh Map List*) and click on button *OK*. Now Memory-Map refreshes the *Map List* and you can select your map (look for the file name of your image). Click on the button *OK* in the *Map List* window. Your map will appear in the main window. In the bottom right corner you will see the pixel coordinates of you cursor. This item will be addressed in "The cookbook".

The fifth and last step is the actual calibration. Open the *Map Calibration* option (version 6.4.3: *Map* > *Map Properties* > *Calibration*/version 5.4.4: *Map* > *Calibration*), You will see the cursor on the screen is changed to a crosshair and the window *Map Calibration* will be opened. To create your first reverence point, zoom in to maximum level and set your crosshair cursor over the pixel and click on the left mouse button. At the top of the Calibration window you will see tree text boxes. In the middle you will see the X-pixel coordinate and in the right the Y-pixel coordinate. In the left box you can substituted the geographical coordinates. When done click on the button *Save Point*. The reference point is saved and you can repeat this operation. After three reference points Memory-Map will make a suggestion for the geographical coordinates. You can edit them. When ready click on the button *OK*. Memory-Map will generate a JPR-file in the same folder as your PNG-file and a

QCT-file (Memory-Map own map format) which will be placed in the folder '*Folder containing converted maps*'. The location of this las folder you can find in the textbox '*Folder containing converted maps*' in the *Refresh Map List* window (*Map* > *Map List …* > *Refresh Map List button*). Unless you want to edit the boundary you an, after generating the QCT-file, remove the PNG- and JPR-file (save them elsewhere). Job done! This item must be addressed in "The cookbook".

The generated JPR-file contains the essential calibration data for Memory map. The same information is needed to import a map.

The generated JPR-file contains the following information:

```
nm=C:\Maps\NL025K-Example.qct
st=0.000000
sn=0.000000
pr=UTM
dm=WGS84
rp1=52.572430,6.493540,0,0
rp2=52.563272,7.230799,19999,0
rp3=52.024235,7.208567,19999,23999
rp4=52.033253,6.480191,0,23999
vp1=0,0
vp2=20032,0
vp3=20032,24000
vp4=0,24000
```

`Green`: Datum related information
`Blue`: Projection related information
`Red`: Reference points
`Purple`: Boundary

Remark: In version 6.4.3 of Memory-Map there is a glitch in step five. When clicking on a pixel the X/Y coordinate the registration in the *Map Calibration* will be two pixel too far to the left and two pixel too far to the top (`0,0` will be registered as `-2,-2`). Correction can be done by adding two pixels to X/Y coordinate. This issue must be addressed in "The cookbook".

A second option to calibrate a map is substitute all values in step five.

Adding/editing the boundary to a map

There are two ways to add or change a boundary: the first one is to use Memory-Map and the second by editing the JPR file.

The first case more or less strait forward. The first step is create a *Route* of the *Boundary*. If there is already a *Boundary* change it in to a route with the function *Route from Boundary* (version 6.4.3: *Map* > *Map Properties* > *Route from Boundary*/version 5.4.4: *Map* > *Route from Boundary*). You treat this *Route* as any other *Route*, adding/editing/moving/deleting Waypoints. If there is no boundary just create a *New Route* inclosing the map area. It is important the start and the end must be the same *Waypoint*. The second step is to change the *Route* in to a *Boundary* (version 6.4.3: *Map* > *Map Properties* > *Boundary from Route* /version 5.4.4: *Map* > *Boundary from Route*). If the PNG- and JPR-file are "in place", Memory-Map will pass the changes through to the JPR-file.

In the second case it is important that, after editing the JPR-file, to delete the QCT-file and to "import" the map again. Only in this way the changes will be implemented by Memory-Map.

This subject will be addressed in "The cookbook".

Adding/editing information to a map

There are two ways to add or change the information to the map: the first one is to use Memory-Map and the second by editing the JPR file.

The first option is strait forward: open the *Edit Map Property* window (version 6.4.3: *Map* > *Map Properties* > *Edit Map Properties* /version 5.4.4: *Map* > *Edit Map Properties*). You can edit all the white textboxes. When done, you can close we window by clicking on the *OK* button. More information in JPR-file paragraph.

The second option is editing the information to the JPR-file. After editing the JPR-file, delete the QCT-file and "import" the map again. Only in this way all the changes will be implemented by Memory-Map. More information in JPR-file paragraph. This option works only if you have the PNG- and JPR file.

This subject will be addressed in "The cookbook".

Merging maps.

The Memory-Map function *Merge Map* (click on your map with your right mouse button en select the option *Merge Map*) gives you the possibility to stich maps together.

This subject will be addressed in "The cookbook".

Adding other position formats (geographical coordinates or grids) to Memory-Map

Adding other position format (geographical coordinate systems) is more or less an uncharted area. In Memory-Map you will find two kinds of position formats, those who are built in and those who are added. The kind you will find in the menu item *Mode* > *Position format*. All the mentioned systems are built in. The systems mentioned in the group *Other Grids* are added. Examples are the Australian and New Zealand coordinate system. Memory map will install these systems when persuading maps with these grids.

The files with the data for these coordinate system are located in the Memory-Map program files structure (version 6.4.3: `C:\Program Files (x86)\Memory-Map\Navigator-6\Grids` and version 5.4.4: `C:\Program Files (x86)\Memory-Map\Navigator\Grids`). They are named `<coordinate system name>.dat` like `zntm.dat` for the New Zealand coordinate system.

The content of these *.dat files can be read with any text editor. Comparing the content with several sources about coordinate systems shows the content equals the proj.4 data. Knowing this, it is possible to create your own grid data file.

This subject will be addressed in "The cookbook".

Importing a map

Within the boundaries of the project Memory-Map there are two options to import a map. The first one is to import a map based on a GeoTIFF-file. The second one is to import a PNG-file with a related JPR-file.

Step one. Copy your PNG-file with the related JPR-file or the GeoTIFF-file to a folder on your computer where Memory-Map can find it. By opening the *Refresh Map List* window (*Map* > *Map list* > button *Refresh Map List*) you can explore this folder(s).

Step two: Open Memory-Map and go to the *Refresh Map List* (*Map* > *Map list* > button *Refresh Map List*) and click on button *OK*. Now Memory-Map refreshes the *Map List* and you can select your map (look for the file name of your image or the name of your map in the JPR-file (`nm=example`)). Click on the button *OK* in the *Map List* window. Your map will appear in the main window. Memory-Map will generate a QCT-file (Memory-Map own map format). Job done!

This subject will be addressed in "The cookbook".

Remarks:

- In both cases the color depth of the image must be 8 bit (255 color) or lower. This issue will be addressed in the application Memory-Map.

- There more restrictions to import a GeoTIFF-file.

JPR-file

In the following paragraph you will find a comparison between the Fugawi JPR-file description and the Memory-Map JPR-file. It seems more or less a subset. Only those 'Fugawi' variables will be addressed which has a 'counterpart' in the Memory-Map JPR file.

Variable: Name.
Name: `nm`
Description: Logical name of the map.
Used as Name of map in *Map List* window.
Used as Name of map and Long Title of map in *Edit Map Data* window.
Type: String, case sensitive.
Values: Free text.
Remarks: If omitted Memory-Map uses the name of the QCT-file with full path as `nm`.

Variable: Datum.
Name: `dm`
Description: Geodetic datum of map.
Used in *Map List* window and in *Edit Map Data* window.
Used for calibrating.
Type: String.
Values: Confirmed values `WGS84`, `NAD83`, `NAD27`, `ETRS89` and `OSGB`.
Remarks: A geodetic datum which is more or less the same as `WGS84` may be used (`ITRF92` , `ITRF2008` and `GDA94`).

Variable: Latitude datum shift.
Name: `st`
Description: Latitude datum shift compared to WGS84.
Used in *Map List* window and in *Edit Map Data* window in decimal minutes.
Used for calibrating.
Type: Numeric in decimal degrees.
Values: Mostly between `+/-0.0001` and `+/-0.000005`
Remarks: Shift of less than `+/-0.000005` is less than a meter.

Variable: Longitude datum shift.
Name: `sn`
Description: Longitude datum shift compared to WGS84.
Used in *Map List* window and in *Edit Map Data* window in decimal minutes.
Used for calibrating.
Type: Numeric in decimal degrees.
Values: Mostly between `+/-0.0001` and `+/-0.000005`
Remarks: Shift of less than `+/-0.000005` is less than a meter.

Variable: Projection.
Name: `pr`
Description: Map projection.
Used in *Map List* window and in *Edit Map Data* window.
Used for calibration.
Type: String.
Values: Confirmed values: `mercator`, `transverse mercator`; `UTM` and `lambert conformal conic`.
Remarks: None.

Variable: Zone.
Name: `zn`
Description: UTM zone.
Used for calibrating.
Type: String.
Values: `01j` to `60j` (northern hemisphere) and

`01t` to `60t` (southern hemisphere).
Remarks: Only used in combination with projection `UTM`.

Variable: Central Meridian.
Name: `pp`
Description: Central Meridian.
Used for calibrating.
Type: Numeric in decimal degrees.
Values: `-180` to `0` to `180`.
Remarks: Used in combination with `lambert conformal conic` projection and `transverse mercator` projection.

Variable: Latitude of origin.
Name: `p1`
Description: Latitude of origin.
Used for calibrating.
Type: Numeric in decimal degrees.
Values: `-180` to `0` to `180`.
Remarks: Used in combination with `lambert conformal conic` projection, `transverse mercator` projection and `mercator` projection (mid- latitude).

Variable: Scale factor.
Name: `p2`
Description: Scale factor.
Used for calibrating.
Type: Numeric.
Values: More or less `1.0`
Remarks: Used in combination with `transverse mercator` projection.

Variable: False northing.
Name: `p3`
Description: False northing.
Used for calibrating.
Type: Numeric.
Values: Depending on coordinate system.
Remarks: Used in combination with `transverse mercator` projection.

Variable: False easting.
Name: `p4`
Description: False easting.
Used for calibrating.
Type: Numeric.
Values: Depending on coordinate system
Remarks: Used in combination with `transverse mercator` projection.

Variable: Standard Parallel 1.
Name: `p5`
Description: Standard Parallel 1.
Used for calibrating.
Type:` Numeric in decimal degrees.
Values: `-90` to `0` to `90`.
Remarks: Used in combination with `lambert conformal conic` projection.

Variable: Standard Parallel 2.
Name: `p6`
Description: Standard Parallel 2.
Used for calibrating.

Type:` Numeric in decimal degrees.
Values: `-90` to `0` to `90`.
Remarks: Used in combination with `lambert conformal conic` projection.

Variable: Scale.
Name: `sc`
Description: Scale of the map.
Used in *Map List* window as sort and selection criterium and in *Edit Map Data* window.
Type: Numeric.
Value: Positive value.
Remarks: Use the number behind `1:` as value.
Related to Resolution.

Variable: Resolution.
Name: `sr`
Description: 'Scanned' resolution.
Type: Numeric.
Value: Positive value.
Remarks: Resolution in dots per inch (DPI).
Related to Scale.

Variable: Image type.
Name: `it`
Description: Graphical type of map image.
Type: String.
Value: `png` (only value).
Remarks: Only PNG-images can be imported using a JPR-file.

Variable: Contour unit.
Name: `cu`
Description: Altitude contour line unit.
Used in *Map List* window and in *Edit Map Data* window.
Type: String
Value: `meters, feet`
Remarks:

Variable: Depth contour unit
Name: `du`
Description: Depth contour line unit
Used in *Map List* window and in *Edit Map Data* window.
Type: String
Value: `meters, feet, fathoms`
Remarks:

Variable: Edition name.
Name: `ed`
Description: Name of map edition.
Used in *Map List* window and in *Edit Map Data* window.
Type: String.
Values: Free text, case sensitive.
Remarks: If the map has a collar the edition indication may printed somewhere on that collar.

Variable: Edition date.
Name: `et`
Description: Release date of map.
Used in *Map List* window and in *Edit Map Data* window.

Type: String.
Value: A date.
Remarks: Use – as day/month/year separator (dd-Memory-Mapyyyy). If the map has a collar
the edition date may printed somewhere on that collar.
(the /separator is used between Edition name and Edition date)

Variable: Revision date.
Name: `dt`
Description: Last revision date
Used in *Map List* window and in *Edit Map Data* window.
Type: String
Value: Adate
Remarks: use – as day/month/year separator (dd-Memory-Mapyyyy). If the map has a collar the last revision date may printed somewhere on that collar.

Variable: Copyright.
Name: `cr`
Description:`Copyright statement
Used in *Map List* window and in *Edit Map Data* window.
Used on top left corner of printed map
Type: String.
Value: Do not use the word 'copyright' and the character '©' in the statement.
Remarks: Be polite to the makers of maps.

Variable: Reference point
Name: `rp1 … rpN`
Description: Reference point 1 to N.
Ties a pixel coordinate to a geographical coordinate.
Type: Decimal degree, decimal degree, numeric, numeric.
Value: Latitude, longitude, x coordinate of the pixel, y coordinate of the pixel.
Remarks: Latitude and longitude in the geodetic datum (`dm=….`).
At least 3 reference points needed for calibration; the more the better.

Variable: Vertex Polygon Points.
Name: `vp1 … vpN`
Description: Boundary points 1 to N.
Type: Numeric, numeric.
Value: X coordinate of the pixel, y coordinate of the pixel'
Remarks: At least 3 points are needed to create a boundary; mostly four points.
The point must be in clockwise or anti clockwise around the map area.
If the map hasn't a collar you can use the corners of the image.
The boundary is necessary for some functionality of Memory-Map (*Map at Cursor*).

The 'data structure' above is the base for all the conversions to the target 'JPR-file'.

Source side

OziExplorer

A OziExplorer map comes always in a pair of files: the first one contain the image of the map and the second one (`example.map`) the calibration data. The format of the image file can be a standard graphical format or one of OziExplorer proprietary formats. (`example.ozf`, `example.ozf2`, `example.ozfx3` or `example.ozf4`).

Handling image files

The image files in standard graphical formats must be converted in to PNG-files with a color depth of 8 bit. This can be done with any raster based graphical application. This item will be addressed in “The cookbook”.

Handling OziExplorer proprietary formats

Searching the internet you will find several applications to convert OZF..-files to PNG-files, most of them commercial (like MAPC2MAPC and Geodata Cloud). In the non-commercial corner you will find GDAL and QGIS. More about these applications in the <….>. This item will be addressed in “The cookbook” as well in the application.

Handling OziExplorer MAP-file format

The MAP-file contains enough data to convert it into a JPR-file. This conversion is an one way, from MAP-file to JPR-file. Although the information form OziExplorer isn’t fully clear (https://www.oziexplorer4.com/eng/help/creating_maps.html and https://www.oziexplorer4.com/eng/help/map_file_format.html) It is possible to create a ’translation’. In the table below you will find for the corresponding data for nearly all necessary JPR variables an equivalent in the .MAP file content.

JPR variable	MAP variable or value	Projection			
		Mercator	Transverse Mercator	Universal Transverse Mercator	Lambert Conformal Conic
`nm=` name	Line 2 of MAP-file	X	X	X	X
`dm=` datum	Line 5, field 1	X	X	X	X
`st=` latitude datum shift	Line 5, field probably 3, value mostly 0,	X	X	X	X
`sn=` longitude datum shift	Line 5, field probably 4, value mostly 0	X	X	X	X
`pr=` projection	Line 9, field 2	X	X	X	X
`zn=` UTM zone	Line 10, field 14			X	
`pp=` Central meridian	Line 40, field 3		X		X
`p1=` Latitude of origin	Line 40, field 2	X	X		X
`p2=` Scale factor	Line 40, field 4		X		
`p3=` False northing	Line 40, field 6		X		
`p4=`	Line 40, field 5		X		

False easting					
p5= standard parallel 1	Line 40, field 7				X
p6= standard parallel 2	Line 40, field 8				X
sc= scale	To be calculated based on line MM1B of MAP-file and sr	X	X	X	X
sr= Resolution	No equivalent, must be selected	X	X	X	X
It= Image type	Line 3, file extension	X	X	X	X
rpN= Reference points	Option 1 (based on calibration data/N to max 30): Field 1: line PointNN, fields 8, 9, 10 of MAP-file; must be converted Field 2: line PointNN, fields 10, 11, 12 of MAP-file; must be converted Field 3: line PointNN, field 3 of MAP-file Field 4: line PointNN, field 4 of MAP-file Option 2 (based on contour data): Field 1: Line MMPLL, N, field 3 of MAP-file Field 2: Line MMPLL, N, field 4 of MAP-file Field 3: Line MMPXY, N, field 3 of MAP-file Field 4: Line MMPXY, N, field 4 of MAP-file	X	X	X	X
vpN= Boundary points	Field 3: Line MMPXY, N, field 3 of MAP-file Field 4: Line MMPXY, N, field 4 of MAP-file	X	X	X	X

Remarks In a MAP-file aren't resolution and scale data. Instead the variable dot size is defined. The relation between scale, resolution and dot size will be described further on in this document.

This subject will be addressed in the application.

MOBAC

The Mobile Atlas Creator (MOBAC) is an application which can create maps for many Geographical and GPS applications. One of them is OziExplorer. In the analyses above we saw OziExplorer maps can be converted to Memory-Map maps.

MOBAC generates a PNG-file with a related MAP-file of the map. This map is based on spherical Mercator projection and WGS84 as datum. The conversion rules based on those parameters are shown in the table below.

JPR variable	MAP variable or value
nm= (name)	Line 2 of MAP-file
dm= (datum)	wgs84
st= (latitude datum shift)	0
sn= (longitude datum shift)	0
pr= (projection)	Mercator
pp= (central meridian)	0
p2= (scale factor)	1
p3= (false northing)	0
p4= (False easting)	0
it= (image type)	PNG
sr= (resolution)	No equivalent, must be selected
sc= (scale)	To be calculated based on line MM1B of MAP-file and sr
RpN= (reference points)	Mask (-)###.##, (-)###.##, ###, ### Option 1 (based on calibration data/N to max 30): Field 1: line PointNN, fields 8, 9, 10 of MAP-file; must be converted Field 2: line PointNN, fields 10, 11, 12 of MAP-file; must be converted Field 3: line PointNN, field 3 of MAP-file Field 4: line PointNN, field 4 of MAP-file Option 2 (based on contour data): Field 1: Line MMPLL, N, field 3 of MAP-file Field 2: Line MMPLL, N, field 4 of MAP-file Field 3: Line MMPXY, N, field 3 of MAP-file Field 4: Line MMPXY, N, field 4 of MAP-file
vpN= (boundary points)	Mask ###, ### Field 3: Line MMPXY, N, field 3 of MAP-file Field 4: Line MMPXY, N, field 4 of MAP-file

Remark: The created PNG-file must be converted to a color depth of 8 bit.

Creating a map with MOBAC will be addressed in “The cookbook”, the conversion of the MAP-file will be addressed in the application.

GeoTIFF-files

TIFF image files can contain more than only graphical data. When a TIFF-file contains georeferenced data it is call a GeoTIFF-file. In the Memory-Map manual is written a GeoTIFF-file with a 8 bit color depth can be imported directly.

This isn't the whole trues. Based on experience there are more restrictions: Memory-Map must understand the datum (`WGS84`, `NAD83`, `NAD27`, `ETRS89` or `OSGB`) and the coordinate system must be Cartesian. This issue will be addressed in the application and mentioned in "The cookbook". If the color depth is the only 'problem' the color depth can be reduced using QGIS.

The second option to import an GeoTIFF-file to split the file in a PNG image file and a JPR calibration file. Creating the PNG-file can be done with any raster based graphical application. This item will be addressed in "The cookbook".

To extract the georeferenced data a special application is needed. In the project M4MM the Geospatial Data Abstraction Library (GDAL) is used to do so. It will 'translate' the georeferenced data in Well Known Text 2.0 (KWT 2.0). This information can be used to create a JPR-file. This conversion will be done in the application.

In the paragraph Source side/Well Known Text 2.0 you will find a detailed conversion table.

GeoPDF-files

PDF-files can contain more than text and graphical data. When a file contains georeferenced data it is call a GeoPDF-file. To import such a file it must be split in a PNG image file and a JPR calibration file.

Creating the PNG-file can be done with a PDF reader/viewer with an export function. In the project M4MM the application PDF-Xchange editor is used; the free version provide us with all the necessary functionality. This procedure will be addressed in "The cookbook".

To extract the georeferenced data a special application is needed. In the project M4MM the Geospatial Data Abstraction Library (GDAL) is used to do so. It will 'translate' the georeferenced data in Well Known Text 2.0 (KWT 2.0). This information can be used to create a JPR-file. This conversion will be done in the application.

In the paragraph Source side/Well Known Text 2.0 you will find a detailed conversion table.

MrSID-files

Multiresolution seamless image database (MrSID) is a file format which contain a n image of a map and the georeferenced data. To import a MrSID-file in must be split into a PNG image file and a JPR calibration file.

Creating the PNG-file can be done with some special programs. One of them the Geospatial Data Abstraction Library (GDAL). This conversion will be addressed in the application.

The generated file may very big and may not have a color depth of 8 bit or lower. To reduce the color depth use a raster based graphical application which can hander very big files. Paint.NET is a good option It can handle files up to 4 GByte. When the image has a transparent layer (for example 32 bit color depth), convert first to a color depth of 24 bit and then to 8 bit. This prevents a black background color of the image. This issue will be addressed in "The cookbook".

To extract the georeferenced data a special application is needed. In the project M4MM GDAL is used to do so. It will 'translate' the georeferenced data in Well Known Text 2.0 (KWT 2.0). This information can be used to create a JPR-file. This conversion will be done in the application.

In the paragraph Source side/Well Known Text 2.0 you will find a detailed conversion table.

Well Known Text 2.0

Table

Contrary to the JPR- and MAP-files documentation the line by line/field by field documentation for GeoTIFF/GeoPDF files is scattered. Most information was found on the GDAL website. Additionally it was collected by analyzing the output of a lot of different GeoTIFF-/GeoPDF-files. The result you will find in the table below. The table isn't complete, but gives, within the scope of the project M4MM, enough information to create JPR-files. The GDAL function `Gdalinfo` can generate the georeferenced content of a GeoTIFF-, GeoPDF- or MrSID-file based on Well Know Text 2.0 (WKT 2.0).

Variable	Description	Remarks/Relation with Gdalinfo
`sc=`	Scale	Must be calculated based on lines: `Pixel Size`, `TIFFTAG_XRESOLUTION`, `TIFFTAG_YRESOLUTION` and `TIFFTAG_RESOLUTIONUNIT`
`pr=`	Projection	Line: `PROJECTION`
`pp=`	Central Meridian if Lambert Conic Conformal, and Transverse Mercator.	Line: `PARAMETER "central meridian"` Gdalinfo generated this value for all Mercator projections.
`p1=`	Latitude of origin if Lambert Conic Conformal, Transverse Mercator Mid-latitude if Mercator	Line: `PARAMETER "latitude_of_origin"` Gdalinfo generated this value for all Mercator projections.
`p2=`	Scale factor if Transverse Mercator	Line: `PARAMETER "scale factor"` Gdalinfo generated this value for all Mercator projections.
`p3=`	False northing if Transverse Mercator	Line: `PARAMETER "false northing"` Gdalinfo generated this value for all Mercator projections.
`p4=`	False easting if Transverse Mercator	Line: `PARAMETER "false easting"` Gdalinfo generated this value for all Mercator projections.
`p5=`	Standard Parallel 1 if Lambert Conic Conformal	Line: `PARAMETER "Latitude of 1st standard parallel"`
`p6=`	Standard Parallel 2 if Lambert Conic Conformal	Line: `PARAMETER "Latitude of 2sd standard parallel"`
`zn=`	Zone UTM number	Line: `PROJCS`
`dm=`	Datum	Line: `DATUM`
`st=`	Latitude datum shift in degrees	Value = 0 (no correlation found)
`sn=`	Longitude datum shift in degrees	Value = 0 (no correlation found)
`nm=`	Name of map	Line: `Files`: use filename without path and without extension
`rp1 . . . . rp5=`	Reference point (latitude, longitude, x, y)	Lines: `Upper Left, Lower Left, Upper right, Lower Right` and `Center` for latitude and longitude Line: `Size is` for pixel coordinates
`vp1 . . . . vp4=`	Boundary polygon in pixels	If the map hasn't a collar: line: `Size is` for pixel coordinates, if it has a `NEATLINE` the boundary polygon can be calculated (not in all cases).
`sr=`	scan resolution	Lines: `TIFFTAG_XRESOLUTION`, `TIFFTAG_YRESOLUTION` and `TIFFTAG_RESOLUTIONUNIT`
`it=`	image type	Line: `Files`: use filename extension

	Type of coordinate system and length units	Line CS[for coordinate system Lines LENGTHUNIT for length unit Used to handle scale, resolution and boundary polygon.
	Amount of coordinate systems	Line BASEGEOGCRS means the georeferenced data contains only one coordinate system.
		Line BOUNDCRS means the georeferenced data contains two coordinate systems, a SOURCECRS and a TARGETCRS. The data of the SOURCECRS must be used.
	Pixelsize/dotsize	Line Pixel Size and lines LENGTHUNIT can be used to establish/calculate resolution and scale.

For the GeoTIFF-file, the GeoPDF-file and MrSID-file the creation of a JPR-file will be addressed separately. Necessary details will be addressed in "The cookbook".

Correlations coordinate system and other variables in WKT 2.0

Between many variables in the WKT 2.0 text exists correlations. Some of them are essential to convert the WKT 2.0 text into a JPR-file. The information below is based on experience. The correlations will be illustrated with real examples.

Correlation between the (BASE)GEOGCRS/SOURCECRS, the Corner Coordinates and NEATLINE from the Metadata. All important elements are accented yellow. In Example 1 you will see the coordinate system is ellipsoidal (CS[ellipsoidal,2]) and the first sets corner coordinates are in decimal degrees. In example 2 you will see the coordinate system is Cartesian (CS[Cartesian,2]) and the first sets corner coordinates are UTM coordinates zone 11N (UTM zone 11N).
In example 3 you see a BOUNDCRS. In this case the correlation between coordinate system and the corner coordinates is between the SOURCECRS and the corner coordinates. I the same example you will see the same correlation between the SOURCECRS and the NEATLINE.

Correlation between Pixel Size and ANGLEUNIT/LENGTHUNIT. All important elements are accented green. In Example 1 you will see the that the unit for the AXIS is defined as ANGLEUNIT["degree" and Pixel Size is also measured (decimal) degrees.
In Example 2 you will see the that the unit for the AXIS is defined as LENGTHUNIT["metre" and Pixel Size is also measured metre.

Both correlations will be used to handle resolution, scale and boundary (NEATLINE) in the application.

Example 1:

```
Driver: GTiff/GeoTIFF
Files: C:\Data\Projects\M4MM\MEMORY-MSPtest\toporama_082o04_9_0_geo.tif
Size is 4500, 3600
Coordinate System is:
GEOGCRS["NAD83",
    DATUM["North American Datum 1983",
        ELLIPSOID["GRS 1980",6378137,298.257222101004,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["geodetic latitude (Lat)",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
```

```
        AXIS["geodetic longitude (Lon)",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4269]]
Data axis to CRS axis mapping: 2,1
Origin = (-116.000000000000000,51.250000000000000)
Pixel Size = (0.000111111111111,-0.000069444444444)
Metadata:
  AREA_OR_POINT=Area
Image Structure Metadata:
  INTERLEAVE=PIXEL
Corner Coordinates:
Upper Left  (-116.0000000,  51.2500000) (116d 0' 0.00"W, 51d15' 0.00"N)
Lower Left  (-116.0000000,  51.0000000) (116d 0' 0.00"W, 51d 0' 0.00"N)
Upper Right (-115.5000000,  51.2500000) (115d30' 0.00"W, 51d15' 0.00"N)
Lower Right (-115.5000000,  51.0000000) (115d30' 0.00"W, 51d 0' 0.00"N)
Center      (-115.7500000,  51.1250000) (115d45' 0.00"W, 51d 7'30.00"N)
Band 1 Block=4500x1 Type=Byte, ColorInterp=Red
Band 2 Block=4500x1 Type=Byte, ColorInterp=Green
Band 3 Block=4500x1 Type=Byte, ColorInterp=Blue
```

Example 2

```
Driver: GTiff/GeoTIFF
Files: C:\Data\Projects\M4MM\MEMORY-MSPtest\toporama_082o04_9_0_utm.tif
Size is 4728, 3786
Coordinate System is:
PROJCRS["NAD83 / UTM zone 11N",
    BASEGEOGCRS["NAD83",
        DATUM["North American Datum 1983",
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4269]],
    CONVERSION["UTM zone 11N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-117,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
```

```
        SCOPE["unknown"],
        AREA["North America - 120°W to 114°W and NAD83 by country"],
        BBOX[30.88,-120,83.5,-114]],
    ID["EPSG",26911]]
Data axis to CRS axis mapping: 1,2
Origin = (569791.076844661962241,5678695.235510840080678)
Pixel Size = (7.500176110693534,-7.499854461001148)
Metadata:
  AREA_OR_POINT=Area
Image Structure Metadata:
  INTERLEAVE=PIXEL
Corner Coordinates:
Upper Left  (  569791.077, 5678695.236) (115d59'59.58"W, 51d15'19.22"N)
Lower Left  (  569791.077, 5650300.787) (116d 0'19.38"W, 51d 0' 0.17"N)
Upper Right (  605251.909, 5678695.236) (115d29'30.78"W, 51d14'59.62"N)
Lower Right (  605251.909, 5650300.787) (115d30' 0.62"W, 50d59'40.75"N)
Center      (  587521.493, 5664498.011) (115d44'57.56"W, 51d 7'30.93"N)
Band 1 Block=4728x1 Type=Byte, ColorInterp=Red
Band 2 Block=4728x1 Type=Byte, ColorInterp=Green
Band 3 Block=4728x1 Type=Byte, ColorInterp=Blue
```

Example 3

```
Driver: PDF/Geospatial PDF
Files: C:\Data\Projects\M4MM\MEMORY-MSPtest\WA_Blaine_20170308_TM_geo.pdf
Size is 3412, 4350
Coordinate System is:
BOUNDCRS[
    SOURCECRS[
        PROJCRS["unnamed",
            BASEGEOGCRS["unknown",
                DATUM["North American Datum 1983",
                    ELLIPSOID["GRS 1980",6378137,298.257222101,
                        LENGTHUNIT["metre",1]],
                    ID["EPSG",6269]],
                PRIMEM["Greenwich",0,
                    ANGLEUNIT["degree",0.0174532925199433,
                        ID["EPSG",9122]]]],
            CONVERSION["UTM zone 10N",
                METHOD["Transverse Mercator",
                    ID["EPSG",9807]],
                PARAMETER["Latitude of natural origin",0,
                    ANGLEUNIT["degree",0.0174532925199433],
                    ID["EPSG",8801]],
                PARAMETER["Longitude of natural origin",-123,
                    ANGLEUNIT["degree",0.0174532925199433],
                    ID["EPSG",8802]],
                PARAMETER["Scale factor at natural origin",0.9996,
                    SCALEUNIT["unity",1],
                    ID["EPSG",8805]],
                PARAMETER["False easting",500000,
                    LENGTHUNIT["Meter",1],
                    ID["EPSG",8806]],
                PARAMETER["False northing",0,
                    LENGTHUNIT["Meter",1],
                    ID["EPSG",8807]],
                ID["EPSG",16010]],
            CS[Cartesian,2],
                AXIS["easting",east,
                    ORDER[1],
                    LENGTHUNIT["Meter",1]],
```

```
                AXIS["northing",north,
                    ORDER[2],
                    LENGTHUNIT["Meter",1]]]],
    TARGETCRS[
        GEOGCRS["WGS 84",
            DATUM["World Geodetic System 1984",
                ELLIPSOID["WGS 84",6378137,298.257223563,
                    LENGTHUNIT["metre",1]]],
            PRIMEM["Greenwich",0,
                ANGLEUNIT["degree",0.0174532925199433]],
            CS[ellipsoidal,2],
                AXIS["latitude",north,
                    ORDER[1],
                    ANGLEUNIT["degree",0.0174532925199433]],
                AXIS["longitude",east,
                    ORDER[2],
                    ANGLEUNIT["degree",0.0174532925199433]],
            ID["EPSG",4326]]],
    ABRIDGEDTRANSFORMATION["Transformation from unknown to WGS84",
        METHOD["Position Vector transformation (geog2D domain)",
            ID["EPSG",9606]],
        PARAMETER["X-axis translation",0.991,
            ID["EPSG",8605]],
        PARAMETER["Y-axis translation",-1.9072,
            ID["EPSG",8606]],
        PARAMETER["Z-axis translation",-0.5129,
            ID["EPSG",8607]],
        PARAMETER["X-axis rotation",0,
            ID["EPSG",8608]],
        PARAMETER["Y-axis rotation",0,
            ID["EPSG",8609]],
        PARAMETER["Z-axis rotation",0,
            ID["EPSG",8610]],
        PARAMETER["Scale difference",1,
            ID["EPSG",8611]]]]
Data axis to CRS axis mapping: 1,2
GeoTransform =
  515922.4086164539, 4.063965482356021, 0.01669691682317546
  5429000.127167323, 0.01669691682317546, -4.063965482356021
Metadata:
  AUTHOR=USGS National Geospatial Technical Operations Center
  CREATION_DATE=D:20170308154420Z
  CREATOR=ESRI ArcSOC 10.0.2.3200
  KEYWORDS=Topographic, Transportation, Hydrography, Orthoimage, U.S.
National Grid, imageryBaseMapsEarthCover, Imagery and Base Maps, Geographic
Names Information System
  NEATLINE=POLYGON ((527558.215332893 5413567.30940932,518281.991871304
5413529.19778356,518224.400914495 5427546.6156721,527500.624376085
5427584.72729786,527558.215332893 5413567.30940932))
  SUBJECT=This image map depicts geographic features on the surface of the
earth.  It was created to provide a representation of accessible geospatial
data which is readily available to enhance the capability of Federal,
State, and local emergency responders for homeland security efforts.  This
image map is generated from selected National Map data holdings and other
cartographic data.
  TITLE=USGS 7.5-minute image map for Blaine, Washington
Corner Coordinates:
Upper Left  (  515922.409, 5429000.127) (122d46'56.10"W, 49d 0'49.27"N)
Lower Left  (  515995.040, 5411321.877) (122d46'55.02"W, 48d51'16.77"N)
Upper Right (  529788.659, 5429057.097) (122d35'33.43"W, 49d 0'49.27"N)
Lower Right (  529861.290, 5411378.847) (122d35'34.52"W, 48d51'16.77"N)
```

```
Center      (  522891.850, 5420189.487) (122d41'14.77"W, 48d56' 3.16"N)
Band 1 Block=3412x1 Type=Byte, ColorInterp=Red
Band 2 Block=3412x1 Type=Byte, ColorInterp=Green
Band 3 Block=3412x1 Type=Byte, ColorInterp=Blue
```

Calculating dot size/grid rotation/coordinates

Based in the assumption of a Cartesian coordinate system en known unit of measurement the dos size (one dot is X metre in reality) can be calculated based on the distance between corner coordinates in pixel and the same distance between the corners in grid coordinates.

Based on the same assumption the rotation angle between the pixel grid and the geographical grid can be establish.

Knowing both these values geographical coordinates can be converted into pixel coordinates and vice versa. This means the NEATLINE can be converted into a boundary in pixels.

The information above will be used in the application.

Proj.4 coordinate system data

An undescribed feature of Memory-Msp is adding 'other grids' which can be used to establish a location or to calibrate maps. This information stored in a grid file (DAT-file) equals the content of a Proj.4 string. The best place to find these string is the webside www.epsg.io from MapTiler Team. The same information can be found with in QGIS. This subject will be addressed in "The cookbook".

Image size/color depth

The size and the color depth of an image can be establish by looking on the detailed properties of the image file. Visual Basic has similar options, be these are restricted to a maximum image size of 385938432 byte (= 368 Mbyte). (this value was empirical establish).

This issue will be addressed in "The cookbook" as well as in the application.

Toolside

GDAL

Visual studio does not support handling georeferenced images. Many raster orientated graphical editors destroy the georeferenced data when manipulating images. This means 'external' support is needed. This support is found in the Geospatial Data Abstraction Library (GDAL). This is a translator library for raster and vector geospatial data formats that is released by the Open Source Geospatial Foundation. It supports calling applications by providing DLL-files, Command line executables (EXE-files) and Python scripts. GDAL comes with a command line environment OSGeoW. This environment can be 'activated' using a `shell` command from within a Visual Basic Application. If necessary the result will be written to a temporary file, which will be processed by M4MM.

Functions

M4MM uses the GDAL functions `GDALINFO` to extract georeferenced data form GeoTIFF-, GeoPDF- and MrSID-files and `GDAL_TRANSLATE` to convert OziExplorers OZF2- and OZFX3-files as well as MrSID-files to PNG-images. Both functions are executables.

The function `GDALINFO` must provide the georeferenced data in the Well Known Text 2.0 (WKT 2.0) format to the application M4MM. To do so GDAL version 3.0.4 (or higher) is needed. The information above will be addressed in the application.

Installing GDAL

GDAL can be installed in two ways. The first one is a standalone installation and the second is an installation as a part of QGIS (a geographical information system). For both options you need to download the installation files from the website of QGIS (Download QGIS) and follow the instructions. The installation will be addressed in "The cookbook".

Python scripts

The reliable use of the GDAL Python scripts in a SHELL command can be disrupted by other Python bases applications on users computer (experience). The application does not use them therefor. In those cases QGIS may proposed. In “The cookbook” the conversion from 24 bits GeoTIFF-files to 8 bit files

QGIS

QGIS is a powerful open source geographic information system. It is party based on GDAL. This is the reason you can install QGIS is stead of GDAL. QGIS isn’t only a replacement for GDAL but it comes with a lot of functionally, for example reducing the color depth of a GeoTIFF file without destroying the georeferenced data. Another option is to extract the Proj.4 coordinate system string. Both functions will be described in “The cookbook”.

Installing QGIS

There are two important versions of QGIS. The first one is the long term release repository (LTR) version. This is the best option if like to use QGIS only as an “extension” of M4MM. If you are more adventuristic and like to explore all the possibilities of QGIS try the latest version. The functionality of M4MM is based on the latest LTR version.

To install QGIS download the installation file from the website of QGIS (Download QGIS) and follow the instructions. If you only use GDAL don’t install the data sets. The installation will be addressed in “The cookbook”

PDF reader/viewer/editor

To extract the image of a GeoPDF-file a PDF reader/viewer/editor is needed. The application must have the following functions:

- An export function to a graphics file format, preferably PNG and 8 bit color depth
- Selecting layers to be exported.
- Changing the export resolution.

A good choice is PDF Exchange Editor. In it free version it provides all the necessary functionality.

Raster graphics application

To reduce the color depth to 8 bit a raster graphics application is needed. Although most raster graphics applications are able to do so, there are some considerations to select one.

- Colorcast; some applications generates a colorcast something not wanted.
- Size of the image; all applications have their limits, the maximum size of an image is mostly one of them. 10,000 by 10,000 pixel may looks great but isn’t. As an example: 10,000 by 10,000 with a scale of 1:25.000 and a resolution of 254 DPI means a covered area is 25 by 25 km.

A good choice is Paint.Net. I most cases it doesn’t has a colorcast and in can handle files with a size up to 4Gbyte. A more complicated alternative is GIMP, which has an option to save the GeoTIFF-data in an image.

Text editor

It may be necessary to edit JPR-files. This will be addressed in a rudimental way in the application M4MM. If you want you can use an external editor. The most simple one like Windows Notepad, will do.

Application development tool

To develop an application a development tool is needed. Originally the community version of Visual Basic 2019/Windows Forms App (.Net framework) as a part of Visual Studio was selected. During the development of the application the project was ported to version Visual Basic 2022.

The used version has its limitations in handling images. A maximum of 2 Gbyte of pixel information (uncompressed) can be handled, which means images about:

- 47,000 by 47000 pixel (8 bit color depth),
- 25,000 by 25000 pixel (24 bit color depth) or
- 22,000 by 22,000 pixel (32 bit color depth).

Word processor

The documentation will be written in Microsoft Word 2013. The auteur is licensed to this version. The finish documents will be exported to PDF with labels for de document structure.

Other aspects

Relation between scale, resolution and pixel size.

In Memory-Maps the scale is one of the variables which can be used to order maps. There for it is useful to include the scale, if known, in to a JPR-file. The resolution is also a variable in the JPR-file. The third useful one, which is not a variable in the JPR-file, is the pixel size (or dot size). If two of these variables are known, the third can be calculated.

Scale

The scale of a paper map is a relation between distances between two point on the map and the same distance in reality. The smaller the scale, the more details you will see on the map.

```
Scale = Length on the map / Real length
```

(same unit of measurement must be used)

A scale of 1:25,000 means 1 centimeter on the map represents 25,000 centimeter (= 250 meter) in reality.

Resolution

The accuracy relation between printed images and digital raster images is called resolution; the amount of pixels per measurement unit. The most common unit of resolution is Dot per Inch (DPI).

```
Resolution (in DPI) = length in pixels / length in inches
```

When scanning or creating a map there is always a trade-off between accuracy and the size of a map. A good guideline is a resolution of 254 DPI or 100 dot per centimeter.

Be aware: Modern scanned maps does have the same resolution for height and width. Older scanned maps may have a different resolution.

Pixel size

In digital raster maps the image is created with pixels. Each pixel represents an area in reality. The smaller this area, the more details you can see on the map.

```
Pixel size = Length in dot / real length (mostly in meter per dot)
```

Mutual relation

Based on the formulas above the mutual relation between the three variable be described in the formula below:

$$\text{Scale (1:x)} = \frac{\text{number of pixel} * \text{resolution [in DPI]} * 2.54}{\text{number of pixel} * \text{dot size [in meter]} * 100}$$

The simplified formula:

$$\text{Scale [1:x]} = \frac{\text{resolution [in DPI]} * 2.54}{\text{dot size [in meter]} * 100}$$

From grid coordinates to pixel coordinates

The presumptions for the math below are

- The geographical coordinate system is orthogonal (Cartesian).
- The pixel coordinate system is orthogonal (Cartesian).

The coordinate conversion is used to convert a NEATLINE to Memory-Map boundary.

Distance between two points

The distance between two points can be calculated using Pythagoras formula. The real distance between a point Q (`Easting, Northing`) and point P (`Easting, Northing`) can be established in the measurement unit of the grid.

The same can be done for pixel coordinates (Q(`x, y`) and P(`x, y`)).

Both distances can be used to calculate the pixel size in in the unit of measurement of the geographical coordinates.

Angle between two points

The angle between two points can be calculated using the distance those points and the signed difference between the Northing of those Points (`Q (Northing) - P (Northing)` (geometrical Sine function).

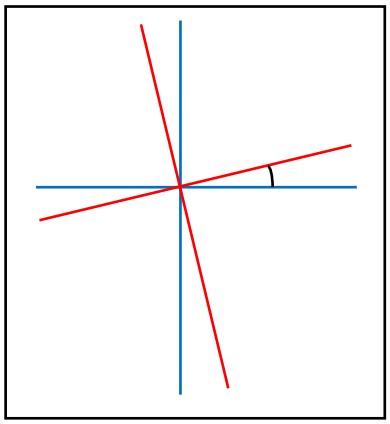
The same can be done for pixel coordinates (Q(`x, y`) and P(`x, y`)).

Be aware the sign of the angle for pixel coordinates is opposite to the geographical ones.

Angle between geographical grid and pixel grid

To convert a geographical coordinate to a pixel coordinate it is necessary to establish the angel between the axis's of both coordinate systems. The most simple way to do so is taking the top left corner as point P and the top right corner as Q.

The angle for the pixel system will be in this case zero and the angle for the geographical system is the angle between the systems.



Convert to Pixel coordinate

The first step is to convert the geographical coordinate to a polar coordinate for point Q where Point P is the origin. Step two is converting this geographical polar coordinate to pixel polar coordinate. The last step converting this pixel polar coordinate to a orthogonal pixel (using Sine an Cosine geometrical functions).

Pixel: Top left coordinate or center coordinate

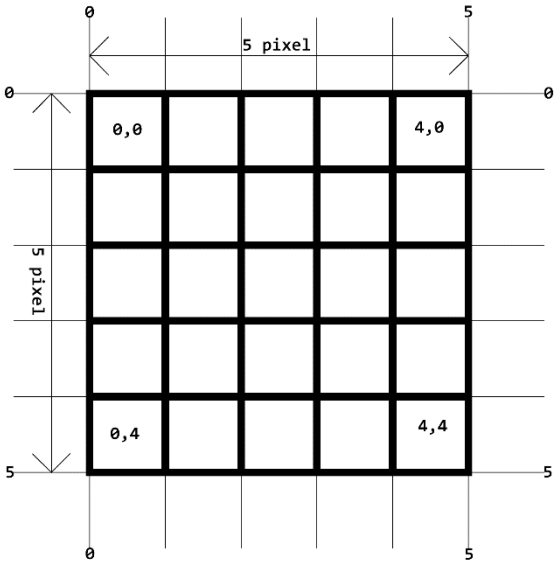
Examining the meta data in a GeoTIFF-file, I found a variable `AREA_OR_POINT`. This variable triggered me to examine the nature of a pixel coordinate. Is it a (Cartesian) coordinate or is it just a location in a two dimensional array (raster).

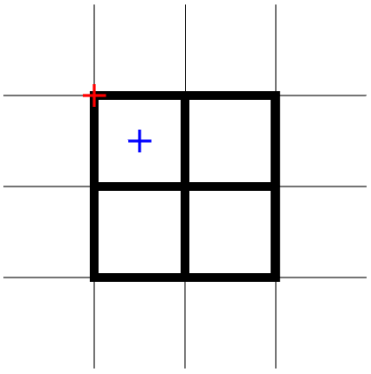
The nature of a (Cartesian) coordinate

A Cartesian coordinate is a mathematical intersection between two (perpendicular) lines, like geographical coordinates (it doesn't matter it is an Cartesian or latitude/longitude). Such a point doesn't has any dimensions.

The nature of a pixel

A pixel is a little spot with a length and a width. Raster images are based on these pixels. These pixels are arranged a in a two dimensional array where the top left pixel gets the indication (0,0). In normal circumstances the pixels are square and have all the same size (like the example to the right).





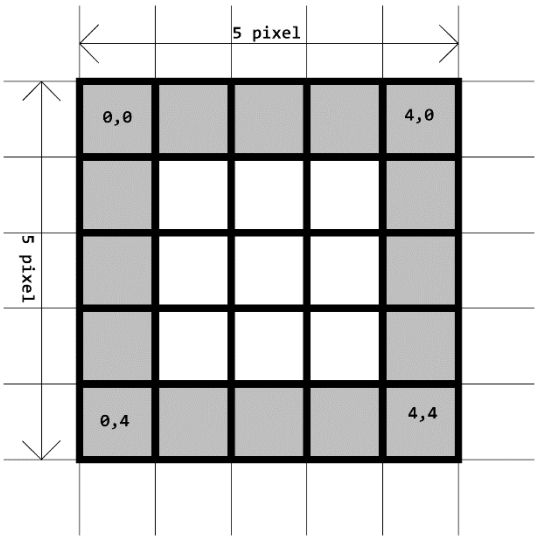
From raster to Cartesian coordinate

If the pixel are square and all the pixel do have the same size it is possible to make a conversion. There are more or less two option. The first to presume the top left corner of pixel represents the pixel (red crosshair in the example to the left) or second the center represents the pixel (blue crosshair). The difference between both coordinates is half a pixel to the right and half a pixel to the bottom.

Memory-Map

In the documentation of Memory-Map there is no clue which tells us how the application handles the coordinates. By observing the results of the calibration of a map and creation boundary of a map may give us a clue.

To do so I took a simple square map without a collar. The first step was the calibration of the map using the substitute method (Calibrating a map/second option) with the pixel coordinates (like the example to the right (0,0) (4,0) (4,4) (0,4) and observe where the black crosshair "landed" on the map (zoomed in to the maximum). Every time it landed on the top left corner of the pixel.



The second step was to repeat the procedure above with Cartesian coordinates of the map (0,0) (5,0) (5,5) (0,5). The crosshairs "landed" exactly on the corners of the map.

From this little experiment we can learn that within Memory-Map top left coordinates must be used to calibrate a map (based on the outline of that map).

When repeating the experiment with the boundary points it had the same result. Using the pixel coordinates (0,0 4,0 4,4 0,4) the most right column and lowest row of pixels were excluded. Using the Cartesian coordinates (0,0 5,0 5,5 0,5) these row and column were included.

This means that top left based coordinates must be used to describe the boundary.

GeoPDF, GeoTIFF and MrSID maps

The the geographical information in these files is mostly based on top left coordinates and the application M4MM will treat them as such.

OziExplorer and MOBAC maps

The documentation of OziExplorer does not give any information about the location of pixel to coordinate. Studying the MAP-files you may draw the conclusion the coordinates OziExplorer uses are based on the center of the pixel. The application M4MM will not convert the information.

Coordinates in OziExplorer maps (PNG- and MAP-file) created by MOBAC will be converted to top left coordinates (calibration and boundary). The four coordinates represents always the corners of the image.

Related to the project M4MM

```
MOBAC > Memory-Map (calibration/boundary)
(0 , 0) > (0 , 0)
(MOBAC width , 0) > (MOBAC width + 1 , 0)
(MOBAC width , MOBAC height) > (MOBAC width + 1 , MOBAC height +1)
(0 , MOBAC height) > (0 , MOBAC height +1)
```

Remarks:

1. In the worst case the calibration can be one pixel "off grid" (2.5 meter on a 1:25,000 scale map with 254 DPI). If this the only source of inaccuracy, it is no problem. But if there are more sources of inaccuracy they may add up to a visible abnormality.
2. If you are merging touching maps (no overlap) one pixel "off grid" if may results in a white line between the two maps or losing a row or column of pixels. Exact calibration and accurate setting of the boundary is important.

Addressing

The issue above must be addressed in the JPR-file build forms and in "The cookbook".

Design

The project M4MM provides the user with procedures to calibrates and/or to coverts maps to be used with the application Memory-Map. These procedures are partly supported with the application M4MM, which automates, if possible, labour intensive and error sensitive steps in these procedures.

To make these procedures available to the user they will be described in a handsome way in a document called “The cookbook”. This document will be supported by the application M4MM. “The cookbook” and the application are the main products of the project.

In the analyse is already made a choice where subjects, items and issues will land, in “The cookbook”, in the application M4MM or in both.

“The cookbook”

To lead a user in a comfortable way through the procedures workflows will be used. These workflows aren’t drawing with fancy symbols, but more or less ‘stories how to do, from begin to end’. Some steps in different procedures are the same. It isn’t efficient to describe them more than once. Many of these repeating steps relate to handling applications (Memory-Map, M4MM, GDAL, etcetera). For this reason the handling of applications is abstracted from the workflows and got their own part in “The cookbook”.

Content of “The cookbook”

“The cookbook” will contains the following parts:

1. An introduction.
2. The workflows: the descriptions of the procedures.
3. The application M4MM explained: the M4MM manual.
4. Handling calibration in Memory-Map explained: alternative manual for Memory-Map functions related to Calibration.
5. Other applications and sources of information explained: alternative manuals and directions for subjects mentioned in the workflows related to the project M4MM.”
6. Background information.

The application M4MM

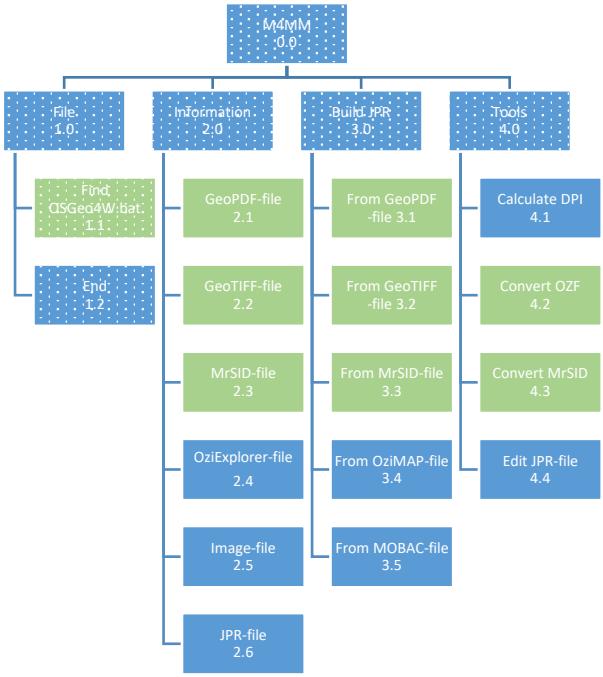
Application structure

The application is in contrast to "The cookbook" function bases designed. The application will open with a form `StartMenu`. From this form you can navigate to all the functionality. These functionalities are grouped around the themes 'Getting information', 'Building a JPR-file' and 'Tools'. Below a diagram of the menu structure.

Forms

The form `StartMenu` (0.0) contains following functions:

- Selecting a function (forms).
- Finding the location of the file `OSGeo4W.bat` (1.1).
- Ending the application (1.2).



Green forms/functions *needs GDAL to work*

Information group

- The form `PdfInfo` (2.1) allows to view the WKT 2.0 content of a GeoPDF-file.
- The form `TiffInfo` (2.2) allows to view the WKT 2.0 content of a GeoPDF-file.
- The form `SidInfo` (2.3) allows to view the WKT 2.0 content of a MrSID-file.
- The form `OziInfo` (2.4) allows to view the content of a OziExplorer MAP-file.
- The form `ImageInfo` (2.5) allows to view the essential meta data of a TIFF-/PNG-file. The form can open the form `ImageView` which allow to view the image of the TIFF-/PNG-file.
- The form `JprInfo` (2.6) allows to view the content of a OziExplorer MAP-file.

Build JPR group

- The form `PdfBuild` (3.1) allows to build, based on the WKT 2.0 content of a GeoPDF-file, a JPR-file.
- The form `TiffBuild` (3.2) allows to build, based on the WKT 2.0 content of a GeoTIFF-file, a JPR-file.
- The form `SidBuild` (3.3) allows to build, based on the WKT 2.0 content of a MrSID-file, a JPR-file.
- The form `OziBuild` (3.4) allows to build, based on the content of a OziExplorer MAP-file, a JPR-file.
- The form `SidBuild` (3.5) allows to build, based on the content of a MAP-file created by MOBAC, a JPR-file.

Tool group

- The form `CalcDpi` (4.1) allows to calculate the resolution based on physical and pixel size of an image.
- The form `OziConv` (4.2) allows to convert an OziExplorer OZF2-/OZFX3-file to a PNG-file.
- The form `SidConv` (4.3) allows to convert an MrSID-file to a PNG-file.
- The form `JprEdit` (4.4) allows to edit (and save) a JPR-File.

The application will have a module common in which all common variables data structure declarations and functions are gathered.

Other resources

Besides forms the following files are added:

- A module `common` , which contains the variables, variable structures, functions and subroutines which are common throughout the whole application M4MM.
- A background image file for all forms: `M4MM.png`.
- An icon image file for all forms: `m4mm.ico`.
- An image `nofile.png` for `ImageInfo` form.
- An image `largefile.png` for `ImageInfo` form.
- A text file containing information about JPR-data used in form `JprEdit`: `jprinfo.txt`.

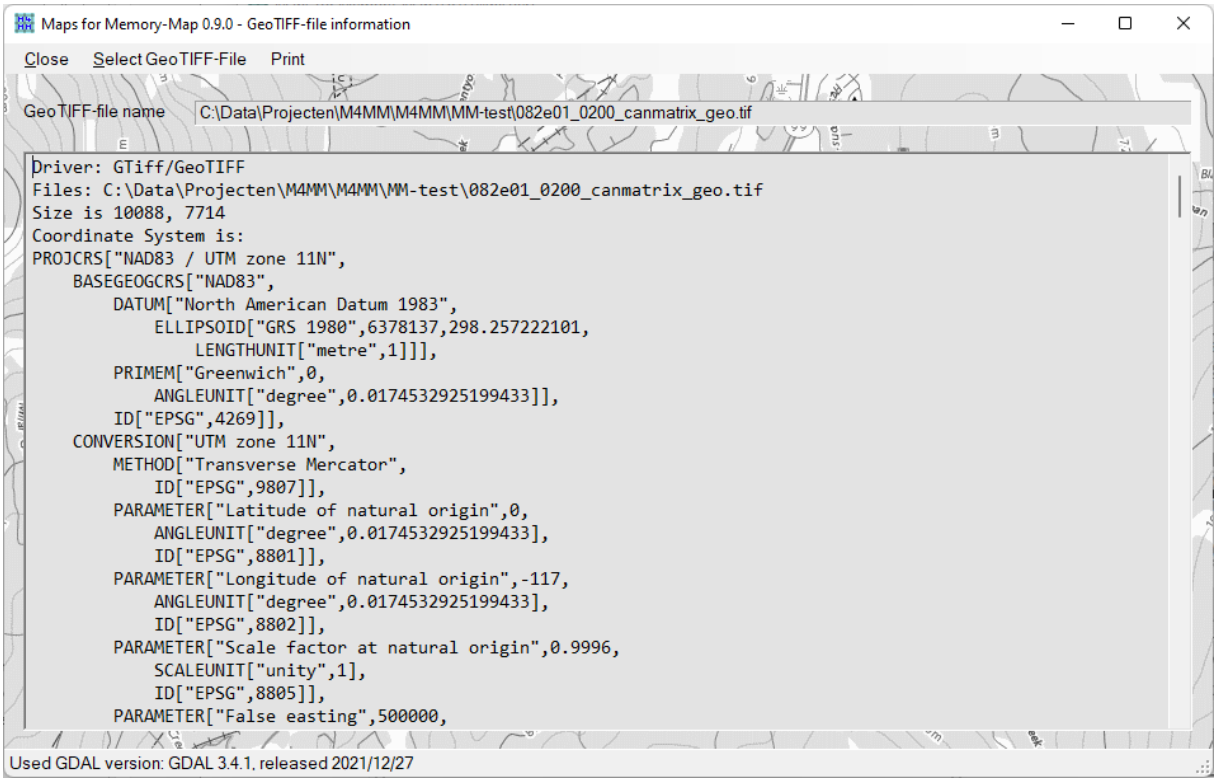
Form structure

Every form has more or less the same design. At the top you will find a `MenuStrip`. The `ToolStripMenuItems` follow the workflow from left to right and from top to bottom. The items are activated in case they can be used.

If necessary a `StatusStrip` is found at the bottom of the form with additional (general) information.

In the middle of the form the retrieved/generated information will be presented. Sometimes an additional input options.

Warnings and questions will be presented using `MsgBoxes`.



Example of a form

If meaningful a print function is added to a form. In the menu options *Print* or *Print data* there will be an option to (de)select landscape printing using the `CheckOnClick` property of `landscapeItem`.

Screens and data/variable structure

Most variables will be declared on form, function and/or subroutine level. Some variables will be declared on application level in the module common, because they are used throughout the whole application and value is more or less constant. The variables declared on form level supports the interaction (data transfer) between modules/forms and/or between functions and subroutines

To keep the Visual Basic code readable/understandable the names for variables with the same purpose are the same name, on form level and on function/subroutine level. In this document only the form level variables will be explained, the function/subroutine level variables will be explained within de Visual Basic code.

Module `Common`

The module `common` contains the variables, variable structures, functions and subroutines which are common throughout the whole application M4MM. All objects in this module will be declared `public`.

Functions

`Function SplitFileName`

Function splits a string representing a filename with a full path into its components.
Output: a variable with the `FileNameStructure` structure.

`Function TranslateDatum`

Function translates into a standard (geographical) datum (NAD27, NAD83, WGS84, GDA94, ITRF92, ITRF2008, OSBG). If the datum isn't recognized it returns its input value.
Output: a string value

`Function TranslateProjection`

Function translates into a standard projection (mercator, transverse mercator, utm, lambert conformal conic). If the projection isn't recognized it returns its input value.
Output: a string value

`Function ClearFileName`

Function resets a variable with the `FileNameStructure` structure. All instances will become a string with on length.
Output: a variable with the `FileNameStructure` structure.

`Function ClearResolution`

Function resets a variable with the `ResStructure` structure. All instances will get value 0.
Output: a variable with the `ResStructure` structure.

`Function ClearCoordinate`

Function resets a variable with the `Coordinate` structure. All instances will become a string with on length except instance `.valid`. It will get value 0.
Output: a variable with the `Coordinate` structure.

`Function ClearJprData`

Function resets a variable with the `JprStructure` structure. All instances will become a string with on length except instances `.ds` and `.ps`. They will get value 0.
Output: a variable with the `JprStructure` structure.

`Function CommaToDot`

Function replaces in a string representing as number, the comma (,) used as decimal separator in to a dot (.).
Output: a string value.

Function CvtLatLonStr

Function converts a string expression, representing a Latitude or longitude, (in degree, minutes, seconds and sign) in a string expression, representing a Latitude or longitude in decimal degrees.
Output: a string value.

Function PixelSize

The Function calculates the pixelsize in units of the grid based on the pixel and grid coordinates of the images corners with the presumption grid is Cartesian. This function is also used to construct a boundary based on the NEATLINE.
Output: a double precision value.

Function ConvertToMaxLineSize

The function checks, within a string with multiple lines, the length of each line and split it, if needed in smaller lines.
Input: a string with multiple lines and a integer representing the maximum line size.
Output: a string with multiple lines with a specific maximum length.

Function CreateImageData

The function returns height, width, horizontal and vertical resolution, colordepth (as string) and indications if the file could be found, the data could extracted and the color depth is usable by Memory-Map
Input: a string with the name of the file.
Output: a variable with the `ImageDataStructure`.

Function UsedGdal

The function returns the used version of GDAL
Input: None.
Output: a string with GDAL version with the release date.

Function WktText

The function returns the WKT 2.0 text included GeoTIFF, GeoPDF and MrSID image files.
Input: Name of the image file.
Output: a string with multiple lines of text.

Subroutines

Subroutine BuildPages

Executed:
By forms when executing their print functions.
Actions:
Builds the array `PageToPrint` with `MaxCharsPerLine` and `MaxLinesPerPage`

Variable structures

Data related to the content of JPR-file

The data used in a JPR-file will be concentrated in a variable called `JprData` (declared on form level) with a structure `JprStructure` (declared on application level). In this structure you will find all entities except the arrays with reference and boundary points. The structure of these points will be declared in `Coordinate` (declared as application level) the arrays `ReferencePoint` and `VertexPoint` (declared on form level).

JprStructure

All data which can be transferred to or used by Memory-Map. Some of these data aren't used in the current version of the application M4MM but mend for future use.

Name of entity	Type of entity	Description/Remarks
nm	String	Name of the map
sc	Double	Estimate scale of the map, to be rounded and converted to string when written to textboxes
sr	Double	Estimate resolution of the map, to be rounded and converted to string when written to textboxes
it	String	Type of image format of the map
dm	String	Geographic datum of the map
st	String	Datum shift (latitude)
sn	String	Datum shift (longitude)
pr	String	Projection of the map
zn	String	UTM zone of the map
pp	String	Central meridian of the map
p1	String	Latitude of the origin of the map
p2	String	Scalefactor of the map
p3	String	False Easting of the map
p4	String	False Northing of the map
p5	String	First standard parallel of the map
p6	String	Second standard parallel of the map
cu	String	Unit of contour lines (height) values: `meters`, `feet`
du	String	Unit of depth values: `meters`, `feet`, `fathoms`
cr	String	Copyright statement
ed	String	Edtion name
et	String	Edition date Mask: `dd-mm-yyyy`
dt	String	Last revision date Mask: `dd-mm-yyyy`
cs	String	Type of coordinate system (not a formal JPR variable)
gu	String	Unit of coordinate system (grid unit) (not a formal JPR variable/only meter)
ds	Double	Dot size in meter (not a formal JPR variable/only meter)
ps	Double	Dot size in units of coordinate system (not a formal JPR variable/only meter)
dc	String	Description of color depth if image (not a formal JPR variable)

Coordinate

All data are related to the same pixel of the image. The entity `valid` is used to validate pixel to be used for a dedicated purpose.

Name of entity	Type of entity	Description

X	String	Pixel X coordinate
Y	String	Pixel Y coordinate
Lat	String	Latitude
Long	String	Longitude
East	String	Easting
North	String	Northing
Valid	Integer	Validation of coordinate

ResStructure

Variables with the resolution structure (ResStructure) are used to establish the resolution and dot size of an image.

Name of entity	Type of entity	Description
ResolutionX	Single	Horizontal resolution of an image
ResolutionY	Single	Vertical resolution of an image
ResolutionUnit	Integer	Unit used for resolution (2=DPI/3=Dot per cm)
PixelSizeX	Double	Horizontal size of a pixel in meter
PixelSizeY	Double	Vertical size of a pixel in meter

ImageDataStructure

Variables with the Image data structure (ImageDatasSructure) are used to extract data form an image file.

Name of entity	Type of entity	Description
Exist	Boolean	Indication file exits
Valid	Boolean	Indication could extracted
Usable	Boolean	Indication color depth ca be used by Memory-Map
PixelSizeX	Double	Horizontal size of a pixel in meter
PixelSizeY	Double	Vertical size of a pixel in meter
ResolutionX	Single	Horizontal resolution of an image
ResolutionY	Single	Vertical resolution of an image
ColorDepth	String	Text description of the color depth

FileNameStructure

Variables with a file name structure (FileNameStructure) are used as an identifier of an specific file(as a whole or as components of that file name).

Name of entity	Type of entity	Description
LongName	String	Name of file with path
Name	String	Name of file without path
ShortName	String	Name of file without path and extension
Path	String	Path to file
Ext	String	Extension of file

Variable declarations on application level (in module common)

Name of variable	Type of variable	Value of variable	Description/Remarks
C34	String	Chr(34)	Represents the character 34 ("), used building command line strings
EndLine	String	Chr(13) + Chr(10)	Represents the end a text line
AppName	String	"Maps for Memory-Map "	Represents the name of the application
Version	String	"#.#.#"	Represents the version of the application
VersionName	String		Represents the version name of the application

BatFile	FileNameStructure	Path\filename	Represents the batch file to activate the OSgeo4W environment
ImageHeight	Long (integer)		Represents the Height in pixel of an image
ImageWidth	Long (integer)		Represents the width in pixel of an image
ImageDiagonal	Double		Represents the length in pixel of diagonal of an image
StringToPrint	String		Represents text to be printed (by DocumentToPrint)*
PageToPrint(#)	String		Array represents pages to print (will be redimensioned)
NumberOfPages	Integer		Represents the number of pages in StringToPrint array (0 is the first)
PagePointer	Integer		Represents the pointer used with array PageToPrint
LineInString(#)	String		Array represents lines in a string (will be redimensioned)
NumberOfLines	Integer		Represents the number of lines in LineInString array (0 is the first)
LinePointer	Integer		Represents the pointer used with LineInString array

* In each form with a print function DocumentToPrint will be declared.

Controls

Two Controls were added to the module common.

Control SelectFile

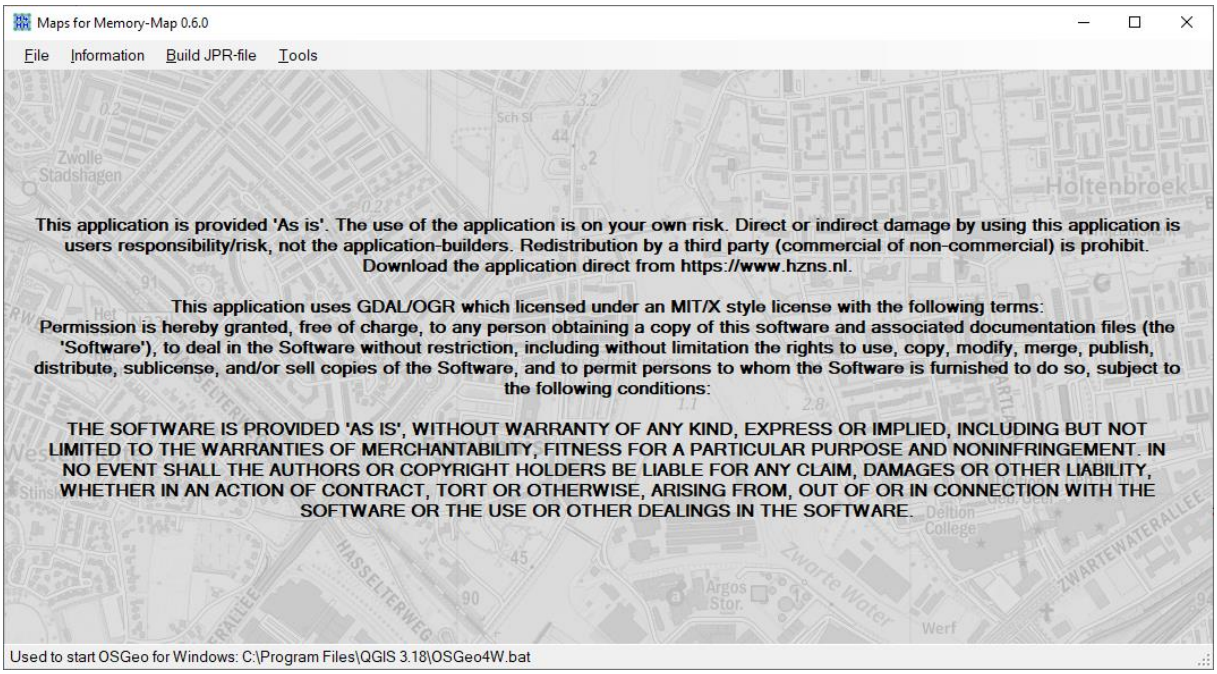
This a FileDialog which is used to select a file.

Control SelectPrinter

This a PrintDialog which is used to select a printer.

Form StartMenu

Screen



Added Controls to the form:
A MenuStrip: `TopStrip`.
A StatusStrip: `BottomStrip`.
A PrintDocument: `PrintSring`.

`ToolsStrip`:
File contains: `IniItem_Click, EndItem_Click`
Information contains: `InfoPdfItem_click, InfoTiffItem_click, InfoMrSidItem_click, InfoOziItem_click, InfoImageItem_click, InfoJprItem_click`
Build JPR-file contains: `Pdf2JprItem_Click, Tiff2JprItem_Click, MrSid2JprItem_Click, Map2JprItem_Click, Mobac2JprItem_Click`
Tools contains: `CalcDpiItem_Click, ConvOziItem_Click, ConvMrSidItem_Click, EditJprItem_Click`

Functions

Form StartMenu contains no functions.

Subroutines

`Subroutine M4MM_Load`
Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Shows the disclaimer
Enable/disable menu items
Calls subroutine `CheckGdalPath` (to load 'BatFilePath').

`Subroutine CheckGdalPath`
Executed:
On demand (by `M4MM_Load`)
Actions:
Finding the file `OSGeo4W.bat`.
If found: Show information and activate items which depend on `OSGeo4W`.

If not found: warning.

Subroutine FindChangeOS
Executed:
On demand (by IniItem)
Actions:
Selecting the file OSGeo4W.bat.
If found: Show information and activate items which depend on OSGeo4W.
If not found: warning.

Subroutine IniItem_Click
Executed:
On demand (by user)
Actions:
Executes subroutine FindChangeOS.

Subroutine EndItem_Click
Executed:
On demand (by user)
Actions:
Ends application M4MM

Subroutine InfoPdfItem_Click
Executed:
On demand (by user)
Actions:
Initiates form InfoPdf.

Subroutine InfoTiffItem_Click
Executed:
On demand (by user)
Actions:
Initiates form InfoTiff.

Subroutine InfoMrSidItem_Click
Executed:
On demand (by user)
Actions:
Initiates form InfoSid.

Subroutine InfoOziItem_Click
Executed:
On demand (by user)
Actions:
Initiates form InfoOzi.

Subroutine InfoMobacItem_Click
Executed:
On demand (by user)
Actions:
Initiates form InfoMobac.

Subroutine InfoImageItem_Click
Executed:
On demand (by user)
Actions:

Initiates form `InfoImage`.

`Subroutine InfoIprItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `InfoJpr`.

`Subroutine Pdf2JprItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `PdfBuild`.

`Subroutine Tiff2JprItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `TiffBuild`.

`Subroutine MrSid2JprItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `SidBuild`.

`Subroutine Map2JprItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `OziBuild`.

`Subroutine Mobac2JprItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `MobacBuild`.

`Subroutine CalcDpiItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `CalcDpi`.

`Subroutine ConvOziItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `OziConv`.

`Subroutine ConvMrSidItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `SidConv`.

`Subroutine EditJprItem_Click`
Executed:
On demand (by `user`)
Actions:
Initiates form `EditJpr`.

`Subroutine PrintContent`
Executed:
By other forms with a print function
Actions:
Selecting a printer, Formatting the text string (with a specific max line length and page length) and initiating printing.

`Subroutine PrintString_PrintPage`
Executed:
By `PrintContent`.
Actions:
Parsing text and parameters to `PrintString` control to print a page.

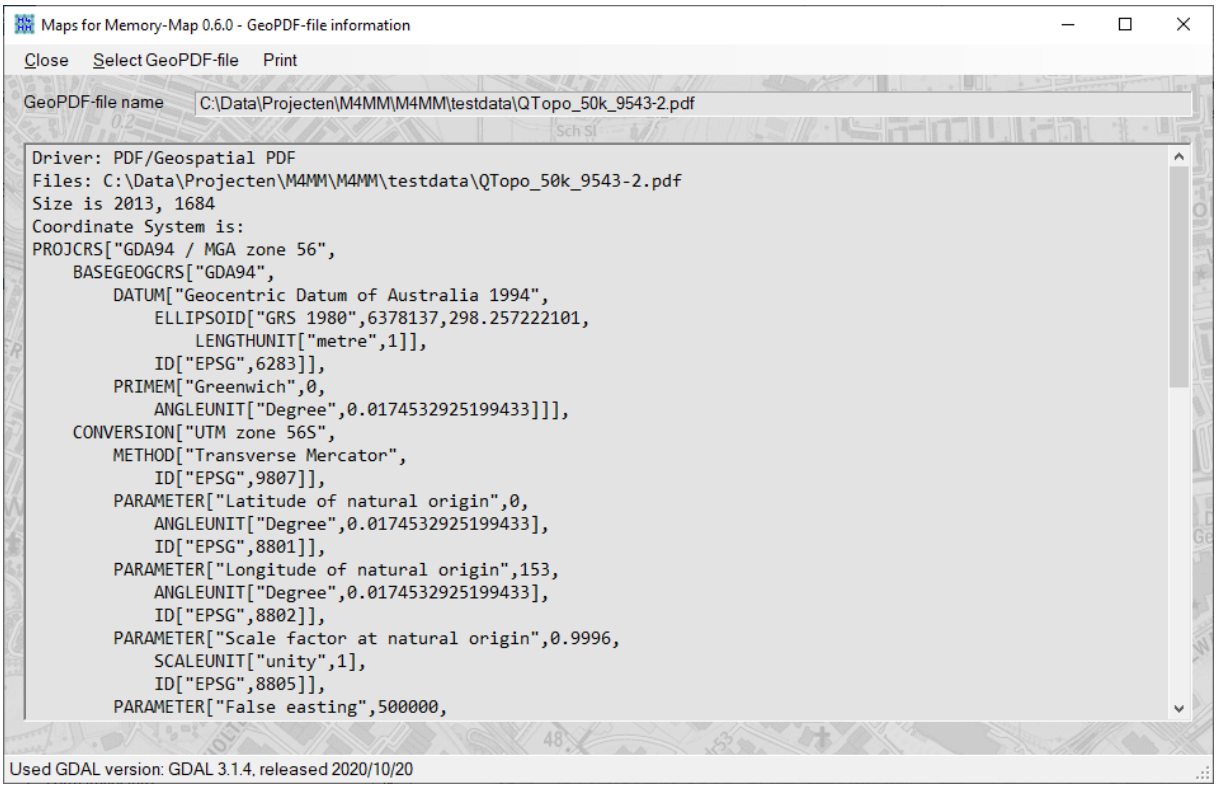
Variable structures
Form `StartMenu` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`IniFile`	`FileNameStructure`		Represents the file name of the `m4mm.ini` which is located in users MyDocuments directory.
`IniContent`	String		Represents the content of `m4mm.ini`.
`PrintFont`	Font		Represents the font to be used with printing
`ContentString`	String		Represents a text string to be printed

Form PdfInfo

Screen



Added Controls to the form: a MenuStrip: `TopStrip` and a StatusStrip: `BottomStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

Form `PdfInfo` contains no functions.

Subroutines

`Subroutine PdfInfo_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine PdfInfo_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`

Executed:
On demand (by user)
Actions:

Selecting a PDF-file.
Presenting the name of selected PDF-file.
Presenting the WKT 2.0 of selected PDF-file.
Presenting the GDAL version

`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from PDF-file using `PrintContent` function (StartMenu).

`Subroutine PrintDataNoClrItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from PDF-file without color data using `PrintContent` function (StartMenu).

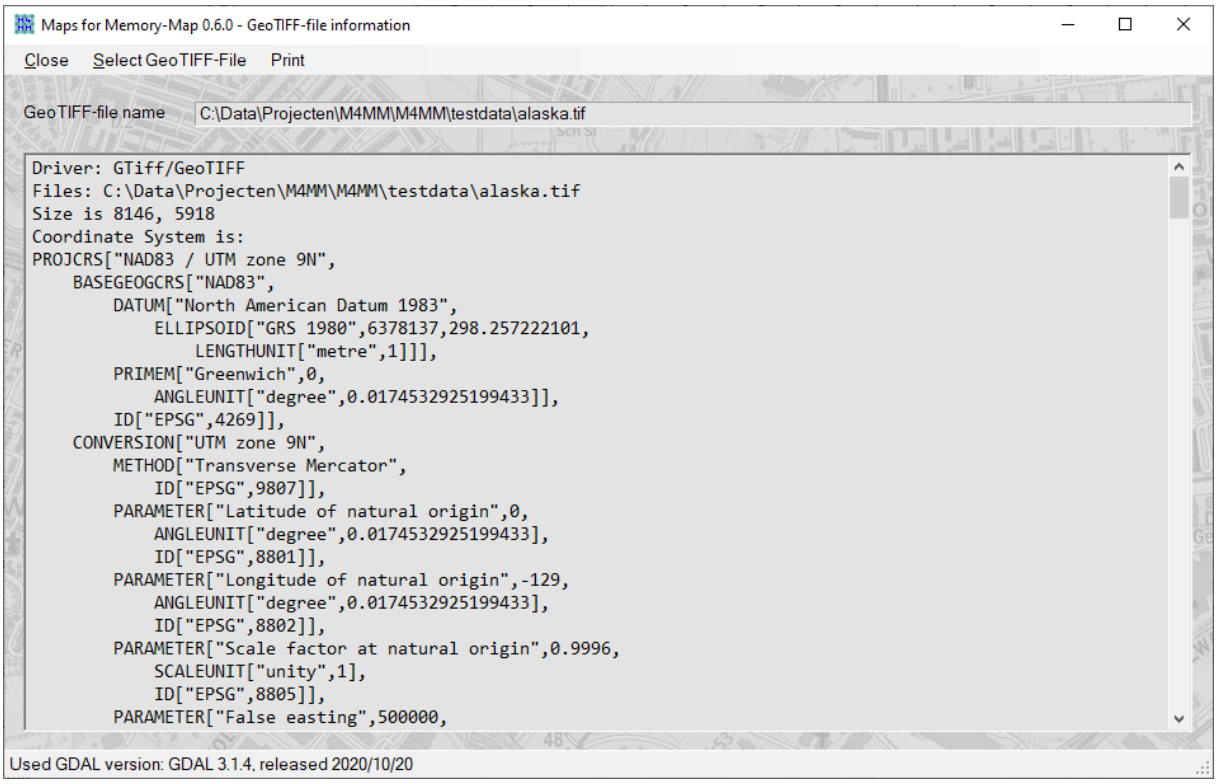
Variable structures
Form `PdfInfo` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`SourceFile`	`FileNameStructure`		Represents the file name of the GeoPDF-file.

Form TifInfo

Screen



Added Controls to the form: a MenuStrip: `TopStrip` and a StatusStrip: `BottomStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

Form `TifInfo` contains no functions.

Subroutines

`Subroutine TifInfo_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine TifInfo_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`

Executed:
On demand (by user)
Actions:

Selecting a GeoTIFF-file.
Presenting the name of selected GeoTIFF-file.
Presenting the WKT 2.0 of selected GeoTIFF-file.
Presenting the GDAL version

`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from TIFF-file using `PrintContent` function (StartMenu).

`Subroutine PrintDataNoClrItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from TIFF-file without color data using `PrintContent` function (StartMenu).

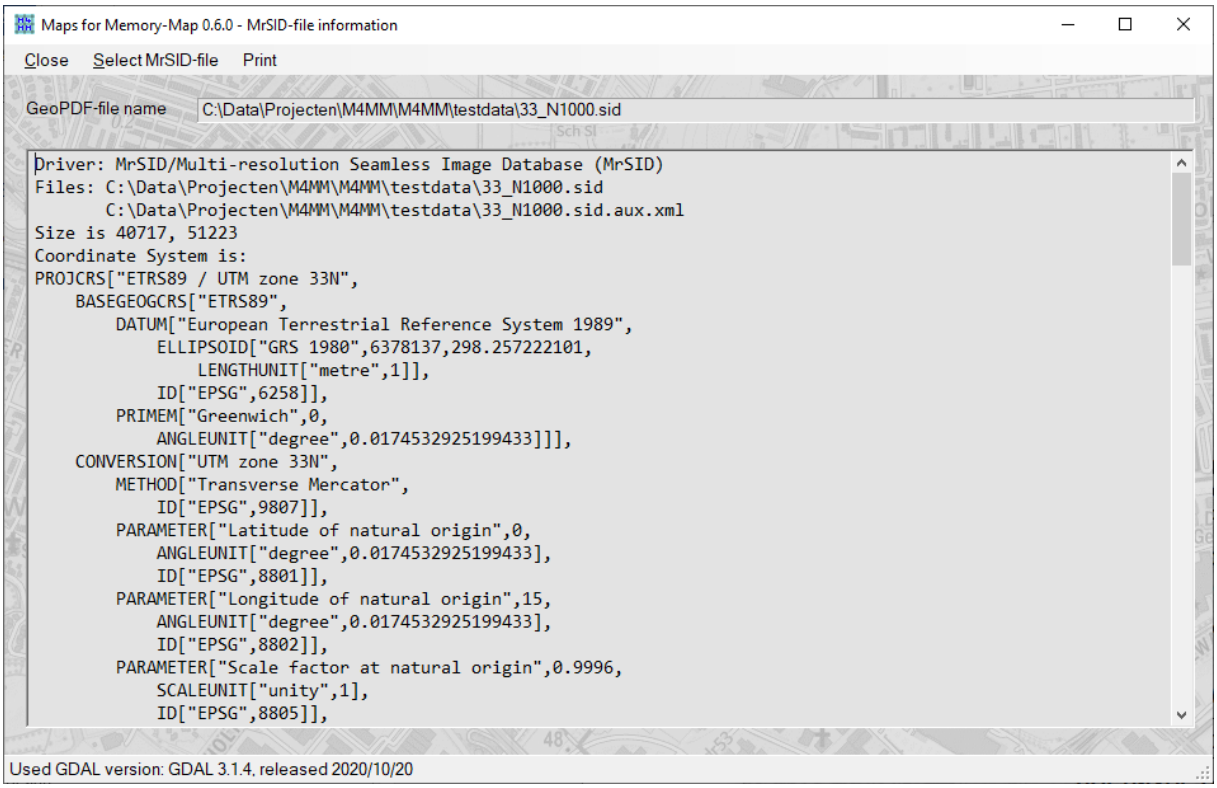
Variable structures
Form `TifInfo` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`SourceFile`	`FileNameStructure`		Represents the file name of the GeoTIFF-file.

Form SidInfo

Screen



Added Controls to the form: a MenuStrip: `TopStrip` and a StatusStrip: `BottomStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

Form `SidInfo` contains no functions.

Subroutines

`Subroutine SidInfo_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine SidInfo_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`

Executed:
On demand (by user)
Actions:

Selecting a MrSID-file.
Presenting the name of selected MrSID-file.
Presenting the WKT 2.0 of selected MrSID-file.
Presenting the GDAL version.

`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from MrSID-file using `PrintContent` function (StartMenu).

`Subroutine PrintDataNoClrItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from MrSID-file without color data using `PrintContent` function (StartMenu).

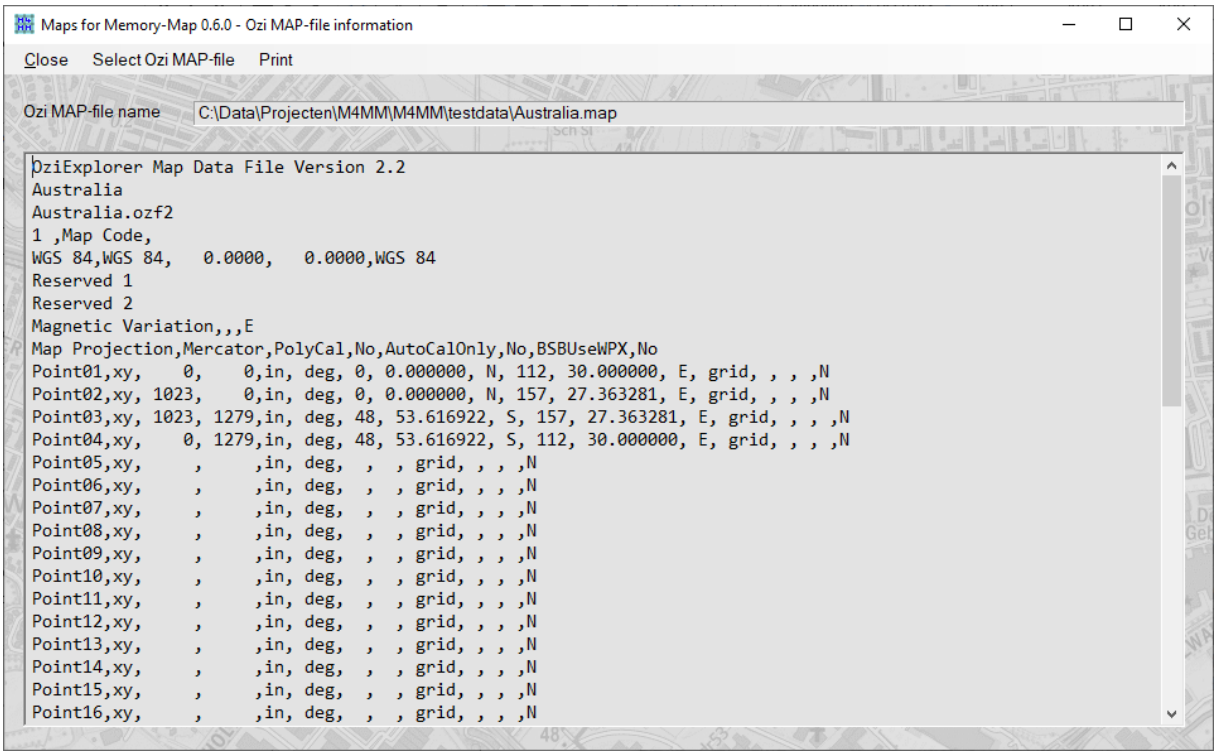
Variable structures
Form `SidInfo` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`SourceFile`	`FileNameStructure`		Represents the file name of the GeoTIFF-file.

Form OziInfo

Screen



Added Control to the form: a MenuStrip: `TopStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

Form `OziInfo` contains no functions.

Subroutines

`Subroutine OziInfo_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine OziInfo_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`

Executed:
On demand (by user)
Actions:
Selecting a Ozi MAP-file.

Presenting the name of selected Ozi MAP-file.
Presenting the content of selected Ozi MAP-file.

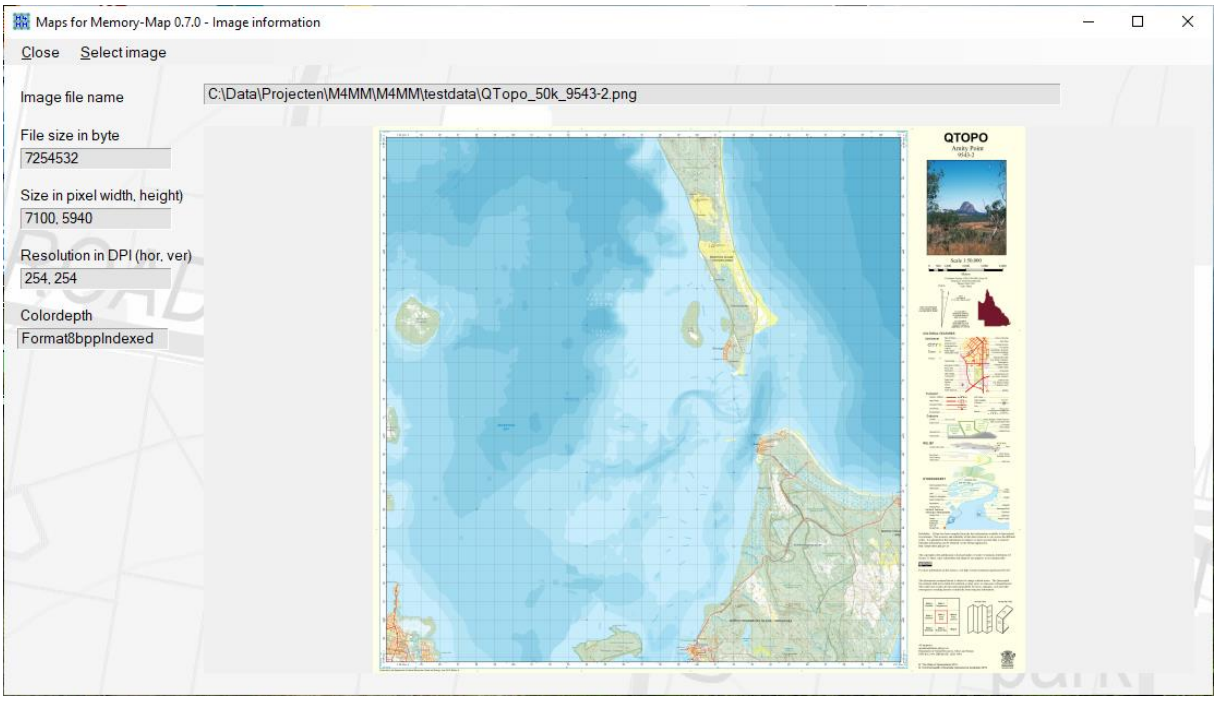
`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints content of OziExplorer MAP-file using `PrintContent` function (StartMenu).

Variable structures
Form `OziInfo` contains no declarations of variable structures.

Variables on form level
Form `OziInfo` contains no declarations on form level.

Form ImageInfo

Screen



Added Control to the form: a MenuStrip: `TopStrip`.

Functions

Form `ImageInfo` contains no functions.

Subroutines

`Subroutine ImageInfo_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Disabling `ViewItem`.
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine ImageInfo_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`

Executed:
On demand (by user)
Actions:
Selecting an image file.
Checking size of image file (about 2 G byte of image data based on BMP format).
If image file exists: The file name will be presented.

If Image data could extracted: the image, the height, the width, the color depth, the horizontal resolution and the vertical resolution of the selected image file will be presented.
Remarks:
If file doesn't exists: a "no file" image will be presented which informs user.
If file is too large to extract: the size of the file will be presented in red and a "large image" image will be presented which informs user.
If the color depth is to large: the description of the color depth will be presented in red.

Variable structures

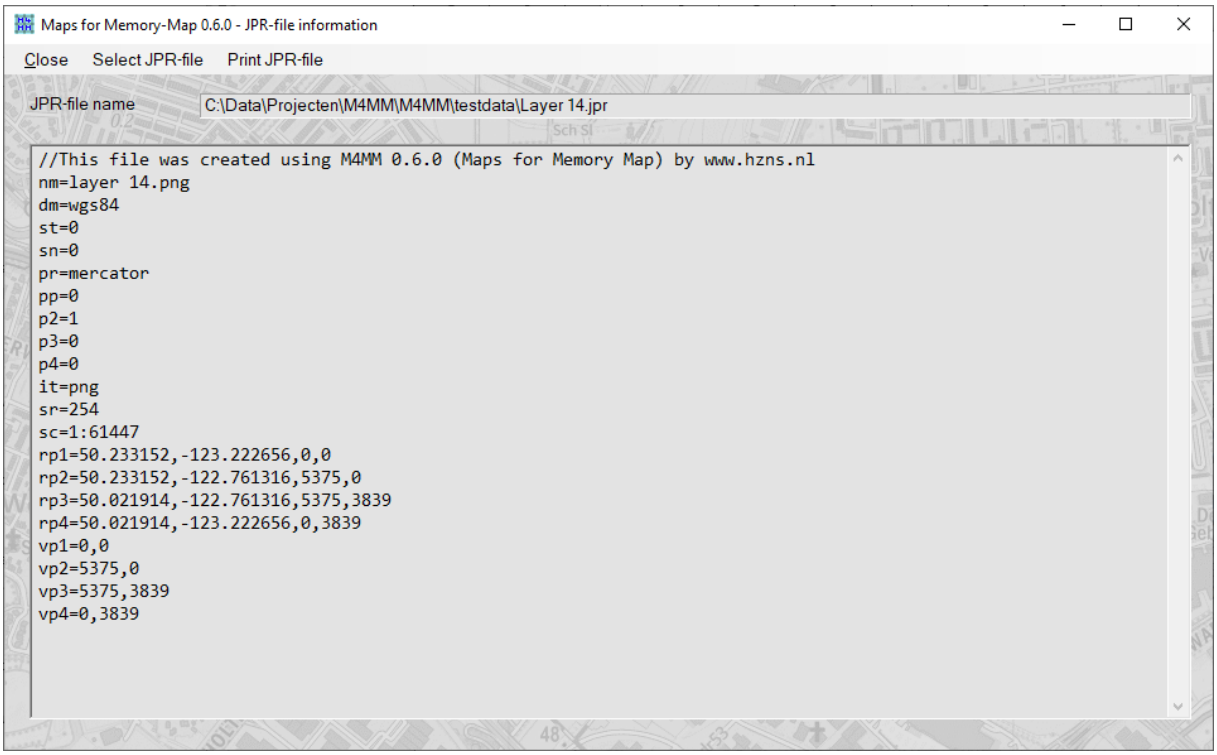
Form `JprInfo` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`MapImage`	Image object		Creates an image object for extracting data from image file

Form JprInfo

Screen



Added Control to the form: a MenuStrip: `TopStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

Form `JprInfo` contains no functions.

Subroutines

`Subroutine JprInfo_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine JprInfo_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`

Executed:
On demand (by user)
Actions:

Selecting a JPR-file.
Presenting the name of selected JPR-file.
Presenting the content of selected JPR-file.

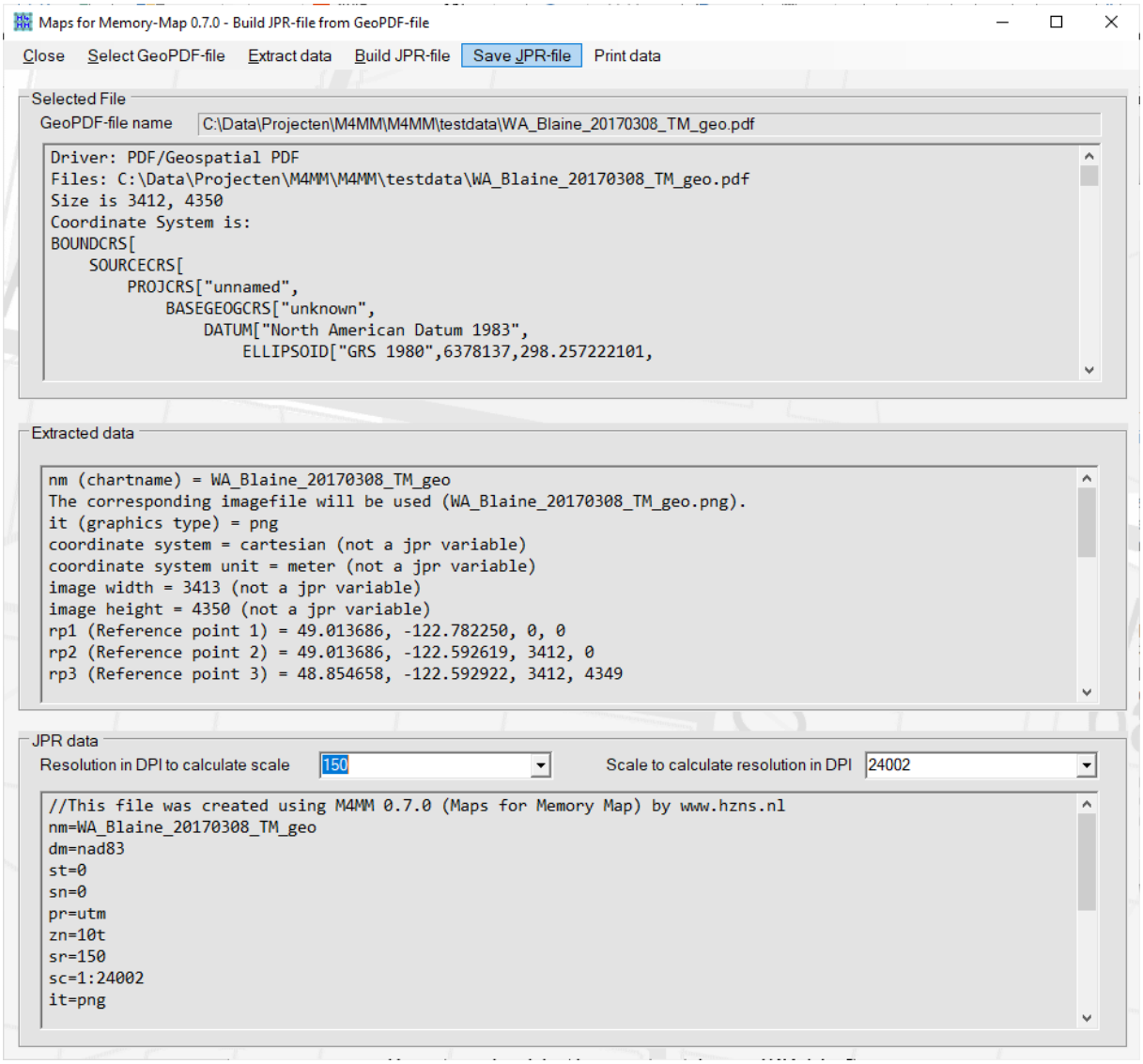
`Subroutine PrintDataItem_Click`
Executed:
On demand (by user).
Actions:
Prints content of JPR-file using PrintContent function (StartMenu).

Variable structures
Form `JprInfo` contains no declarations of variable structures.

Variables on form level
Form `JprInfo` contains no declarations on form level.

Form PdfBuild

Screen



Added Control to the form: a MenuStrip: `TopStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

`Function BuildJprContent(Trigger)`

Function builds the content for a JPR-file.
Input: `Trigger`; indication which subroutine called the function.
`ini`: `BuildItem_Click` called
`dpi`: `DpiBox_LostFocus` or `DpiBox_KeyPress` called
`scale`: `ScaleBox_LostFocus` or `ScaleBox_KeyPress` called
Output: a string with the content for the JPR-file.

Subroutines

`Subroutine PdfBuild_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Loading values into `DpiBox`.

Disabling `TopStrip` items except `CloseItem` and `SelectItem`.
Disabling groups.
Disabling `StartMemu`.

Subroutine `CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

Subroutine `PdfBuild_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

Subroutine `SelectItem_Click`

Executed:
On demand (by user)
Actions:
Selecting and opening a GeoPDF-file.
Creating values for variables: `SourceFile`, `JprFile`, `ImageFile`, `GeoData`
Retrieving WKT from GeoPDF-file.
Presenting `SourceFile`.
Presenting `GeoData`.
Checking for a related TIFF- or PNG-file.
Checking WKT for georeferenced data.
Enabling form elements for next step.

Subroutine `ExtractItem_Click`

Executed:
On demand (by user) if `GeoData` contains georeferenced data.
Actions:
Breaking `GeoData` up into array `GeoDataLine(#)`
Breaking `GeoData` up into blocks (`Block(#)`).
Retrieving `JprData`.
Retrieving `ReferencePoint(#)`.
Retrieving `VertexPoint(#)` (boundary).
Handling NEATLINE in relation with `VertexPoint(#)`.
Presenting extracted data.
Enabling form elements for next step.

Subroutine `BuildItem_Click`

Executed:
On demand (by user).
Actions:
Enabling `DpiBox` if enough data available.
Building content for the JPR-file by using function `BuildJprContent`.
Presenting content for the JPR-file.
Enabling form elements for next step.

Subroutine `SaveItem_Click`

Executed:
On demand (by user).
Actions:
Saving content for JPR-file into JPR-file.

Subroutine ClearData
Executed:
On demand (by ExtractItem_Click).
Actions:
Clearing all JPR related data.

Subroutine SplitNeatLine
Executed:
On demand (by ExtractItem_Click).
Actions:
Extracting the grid coordinates from a NEATLINE and writing then into VertexPoint(#) Array.

Subroutine NeatToVertex
Executed:
On demand (by ExtractItem_Click).
Actions:
Converting the grid coordinates (from a NEATLINE) in a VertexPoint(#) array into pixel coordinates and adding them to the VertexPoint(#) array.

Subroutine DpiBox_LostFocus
Executed:
On event lost focus.
Actions:
Recalculating Scale (JprData.sc).

Subroutine DpiBox_KeyPress
Executed:
On event key press.
Actions:
Recalculating Scale (JprData.sc) if Enter button is used.

Subroutine ScaleBox_LostFocus
Executed:
On event lost focus.
Actions:
Recalculating resolution (JprData.sr).

Subroutine ScaleBox_KeyPress
Executed:
On event key press.
Actions:
Recalculating resolution (JprData.sr) if Enter button is used.

Subroutine PrintItem_Click
Executed:
On demand (by user)
Actions:
Enables/disables PrintDataItem_Click, PrintDataNoClrItem_Click, ExtractedDataItem_Click and JprDataItem_Click (no data in related textbox)
Condition: PrintItem.CheckOnClick must be True.

Subroutine PrintDataItem_Click
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from PDF-file using PrintContent function (StartMenu).

`Subroutine PrintDataNoClrItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from PDF-file without color data using `PrintContent` function (StartMenu).

`Subroutine ExtractedDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints Extracted data from GeoPDF-file using `PrintContent` function (StartMenu).

`Subroutine JprDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints JPR-data from GeoPDF-file using `PrintContent` function (StartMenu).

Variable structures
Form `PdfBluid` contains no declarations of variable structures.

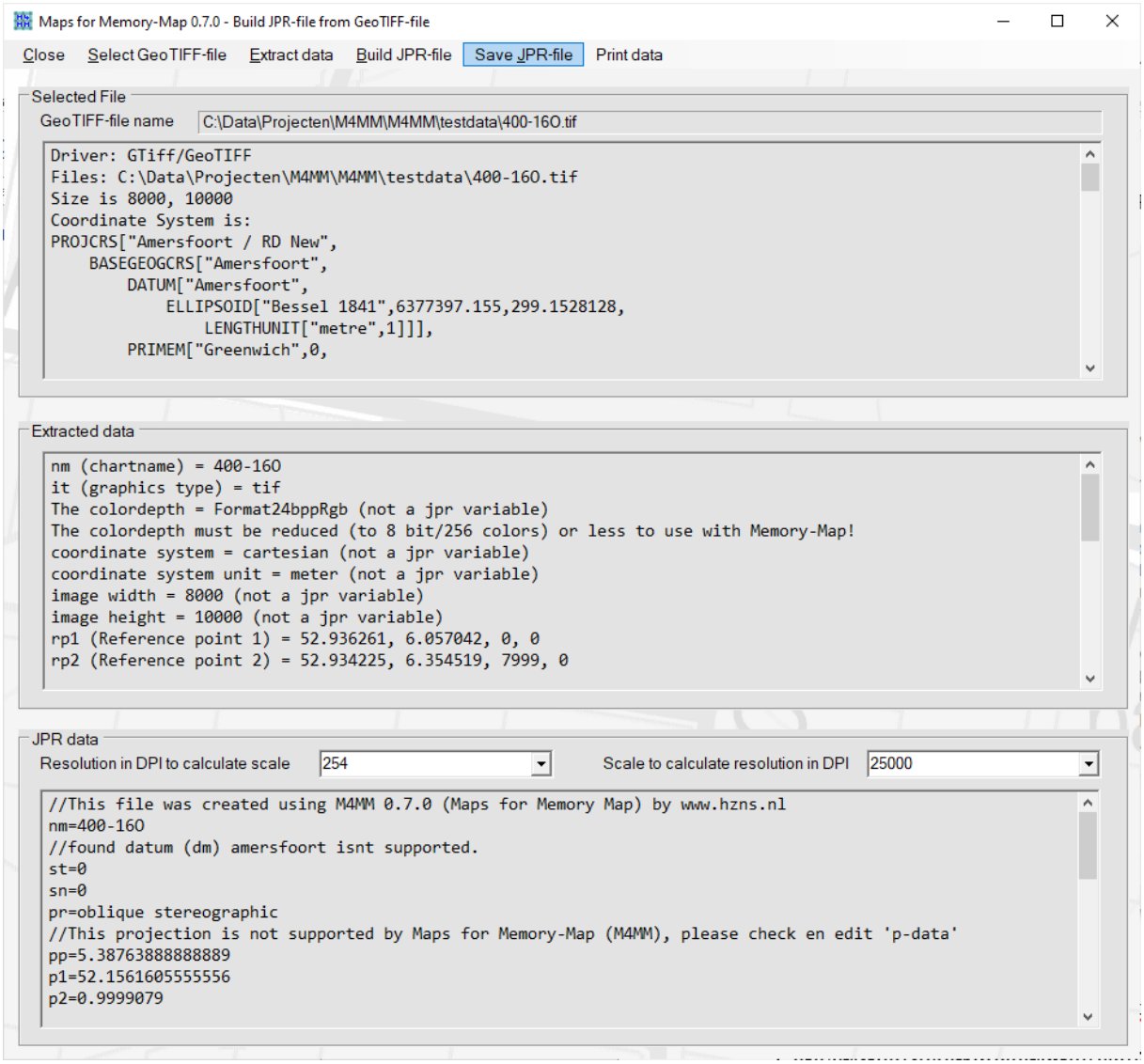
Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`SourceFile`	`FileNameStructure`		Represents the file name of the GeoPDF-file.
`ImageFile`	`FileNameStructure`		Represents the file name of the file containing the image of a map (TIFF- or PNG-file.
`JprFile`	`FileNameStructure`		Represents the file name of the JPR-file
`GeoData`	String		Represents the WKT 2.0 content of the source-file
`GeoDataLen`	integer		represents the Length of `GeoData`-string.
`JprData`	`JprStructure`		Represents the `JprData` needed to build a JPR-file.
`ResData`	`ResStructure`		Represents the resolution data of an image.
`ReferencePoint(#)`	Array of `Coordinate`		Array representing the georeferenced points of a map.
`VertexPoint(#)`	Array of `Coordinate`		Array representing the boundary points of a map (= `vertexpoint` by FUGAWI).
`VertexAmount`	Integer		Represents the amount of `VertexPoint`.

`NeatLine`	String		Represents the NEATLINE as found in the WKT 2.0
`ImageData`	`ImageDataStructure`		Represents extracted data from `ImageFile`

Form `TiffBuild`

Screen



Added Control to the form: a MenuStrip: `TopStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

`Function BuildJprContent(Trigger)`

Function builds the content for a JPR-file.
Input: `Trigger`; indication which subroutine called the function.
`ini`: `BuildItem_Click` called
`dpi`: `DpiBox_LostFocus` or `DpiBox_KeyPress` called
`scale`: `ScaleBox_LostFocus` or `ScaleBox_KeyPress` called
Output: a string with the content for the JPR-file.

Subroutines

`Subroutine TiffBuild_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Loading values into `DpiBox`.

Disabling `TopStrip` items except `CloseItem` and `SelectItem`.
Disabling groups.
Disabling `StartMemu`.

Subroutine `CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

Subroutine `TiffBuild_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

Subroutine `SelectItem_Click`

Executed:
On demand (by user)
Actions:
Selecting and opening a GeoTIFF-file.
Creating values for variables: `SourceFile`, `JprFile`, `GeoData`
Retrieving WKT from GeoTIFF-file.
Presenting `SourceFile`.
Presenting `GeoData`.
Checking WKT for georeferenced data.
Enabling form elements for next step.

Subroutine `ExtractItem_Click`

Executed:
On demand (by user) if `GeoData` contains georeferenced data.
Actions:
Breaking `GeoData` up into array `GeoDataLine(#)`
Breaking `GeoData` up into blocks (`Block(#)`).
Retrieving `JprData`.
Retrieving `ReferencePoint(#)`.
Retrieving `VertexPoint(#)` (boundary).
Handling NEATLINE in relation with `VertexPoint(#)`.
Presenting extracted data.
Enabling form elements for next step.

Subroutine `BuildItem_Click`

Executed:
On demand (by user).
Actions:
Enabling `DpiBox` if enough data available.
Building content for the JPR-file by using function `BuildJprContent`.
Presenting content for the JPR-file.
Enabling form elements for next step.

Subroutine `SaveItem_Click`

Executed:
On demand (by user).
Actions:
Saving content for JPR-file into JPR-file.

`Subroutine ClearData`
Executed:
On demand (by `ExtractItem_Click`).
Actions:
Clearing all JPR related data.

`Subroutine SplitNeatLine`
Executed:
On demand (by `ExtractItem_Click`).
Actions:
Extracting the grid coordinates from a NEATLINE and writing then into `VertexPoint(#)` Array.

`Subroutine NeatToVertex`
Executed:
On demand (by `ExtractItem_Click`).
Actions:
Converting the grid coordinates (from a NEATLINE) in a `VertexPoint(#)` array into pixel coordinates and adding them to the `VertexPoint(#)` array.

`Subroutine DpiBox_LostFocus`
Executed:
On event lost focus.
Actions:
Recalculating Scale (`JprData.sc`).

`Subroutine DpiBox_KeyPress`
Executed:
On event key press.
Actions:
Recalculating Scale (`JprData.sc`) if `Enter` button is used.

`Subroutine ScaleBox_LostFocus`
Executed:
On event lost focus.
Actions:
Recalculating resolution (`JprData.sr`).

`Subroutine ScaleBox_KeyPress`
Executed:
On event key press.
Actions:
Recalculating resolution (`JprData.sr`) if `Enter` button is used.

`Subroutine PrintItem_Click`
Executed:
On demand (by user)
Actions:
Enables/disables `PrintDataItem_Click`, `PrintDataNoClrItem_Click`, `ExtractedDataItem_Click` and `JprDataItem_Click` (no data in related textbox)
Condition: `PrintItem.CheckOnClick` must be `True`.

`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from GeoTIFF-file using `PrintContent` function (StartMenu).

`Subroutine PrintDataNoClrItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from GeoTIFF-file without color data using `PrintContent` function (StartMenu).

`Subroutine ExtractedDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints Extracted data from GeoTIFF-file using `PrintContent` function (StartMenu).

`Subroutine JprDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints JPR-data from GeoTIFF-file using `PrintContent` function (StartMenu).

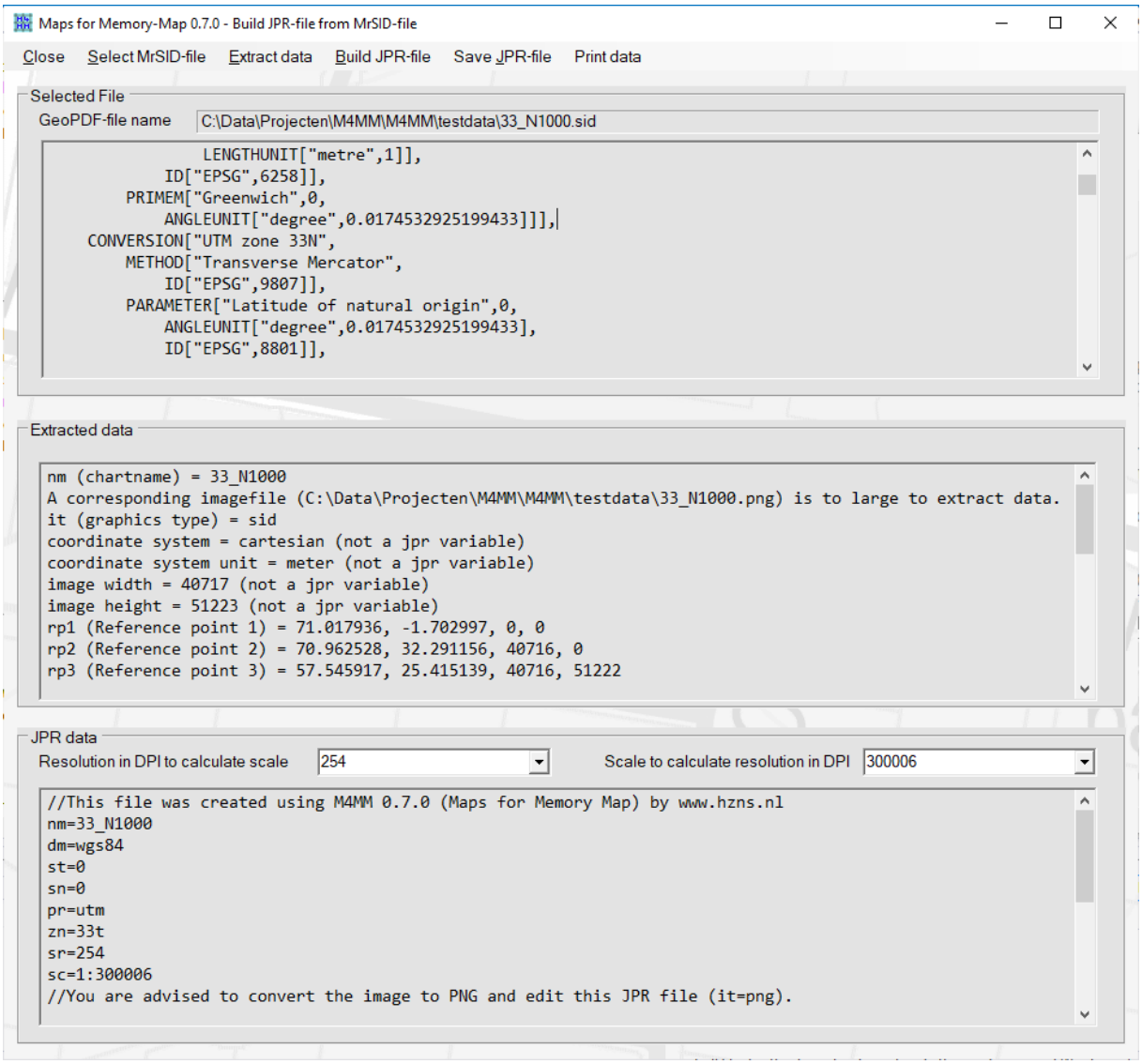
Variable structures
Form `PdfBluid` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`SourceFile`	`FileNameStructure`		Represents the file name of the GeoTIFF-file.
`JprFile`	`FileNameStructure`		Represents the file name of the JPR-file
`GeoData`	String		Represents the WKT 2.0 content of the source-file
`GeoDataLen`	integer		represents the Length of `GeoData`-string.
`JprData`	`JprStructure`		Represents the `JprData` needed to build a JPR-file.
`ResData`	`ResStructure`		Represents the resolution data of an image.
`ReferencePoint(#)`	Array of `Coordinate`		Array representing the georeferenced points of a map.
`VertexPoint(#)`	Array of `Coordinate`		Array representing the boundary points of a map (= `vertexpoint` by FUGAWI).
`VertexAmount`	Integer		Represents the amount of `VertexPoint`.
`NeatLine`	String		Represents the NEATLINE as found in the WKT 2.0
`ImageData`	`ImageDataStructure`		Represents extracted data from `SourceFile`

Form SidBuild

Screen



Added Control to the form: a MenuStrip: `TopStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

`Function BuildJprContent(Trigger)`

Function builds the content for a JPR-file.
Input: `Trigger`; indication which subroutine called the function.
`ini`: `BuildItem_Click` called
`dpi`: `DpiBox_LostFocus` or `DpiBox_KeyPress` called
`scale`: `ScaleBox_LostFocus` or `ScaleBox_KeyPress` called
Output: a string with the content for the JPR-file.

Subroutines

`Subroutine SidBuild_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Loading values into `DpiBox`.

Disabling `TopStrip` items except `CloseItem` and `SelectItem`.
Disabling groups.
Disabling `StartMemu`.

Subroutine `CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

Subroutine `SidBuild_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

Subroutine `SelectItem_Click`

Executed:
On demand (by user)
Actions:
Selecting and opening a GeoPDF-file.
Creating values for variables: `SourceFile`, `JprFile`, `ImageFile`, `GeoData`
Retrieving WKT from GeoPDF-file.
Presenting `SourceFile`.
Presenting `GeoData`.
Checking for a related TIFF- or PNG-file.
Checking WKT for georeferenced data.
Enabling form elements for next step.

Subroutine `ExtractItem_Click`

Executed:
On demand (by user) if `GeoData` contains georeferenced data.
Actions:
Breaking `GeoData` up into array `GeoDataLine(#)`
Breaking `GeoData` up into blocks (`Block(#)`).
Retrieving `JprData`.
Retrieving `ReferencePoint(#)`.
Retrieving `VertexPoint(#)` (boundary).
Presenting extracted data.
Enabling form elements for next step.

Subroutine `BuildItem_Click`

Executed:
On demand (by user).
Actions:
Enabling `DpiBox` if enough data available.
Building content for the JPR-file by using function `BuildJprContent`.
Presenting content for the JPR-file.
Enabling form elements for next step.

Subroutine `SaveItem_Click`

Executed:
On demand (by user).
Actions:
Saving content for JPR-file into JPR-file.

`Subroutine ClearData`
Executed:
On demand (by `ExtractItem_Click`).
Actions:
Clearing all JPR related data.

`Subroutine DpiBox_LostFocus`
Executed:
On event lost focus.
Actions:
Recalculating Scale (`JprData.sc`).

`Subroutine DpiBox_KeyPress`
Executed:
On event key press.
Actions:
Recalculating Scale (`JprData.sc`) if `Enter` button is used.

`Subroutine ScaleBox_LostFocus`
Executed:
On event lost focus.
Actions:
Recalculating resolution (`JprData.sr`).

`Subroutine ScaleBox_KeyPress`
Executed:
On event key press.
Actions:
Recalculating resolution (`JprData.sr`) if `Enter` button is used.

`Subroutine PrintItem_Click`
Executed:
On demand (by user)
Actions:
Enables/disables `PrintDataItem_Click`, `PrintDataNoClrItem_Click`, `ExtractedDataItem_Click` and `JprDataItem_Click` (no data in related textbox)
Condition: `PrintItem.CheckOnClick` must be `True`.

`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from MrSID-file using `PrintContent` function (StartMenu).

`Subroutine PrintDataNoClrItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from MrSID-file without color data using `PrintContent` function (StartMenu).

`Subroutine ExtractedDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints Extracted data from MrSID-file using `PrintContent` function (StartMenu).

`Subroutine JprDataItem_Click`

Executed:
On demand (by user)
Actions:
Prints JPR-data from MrSID-file using `PrintContent` function (StartMenu).

Variable structures

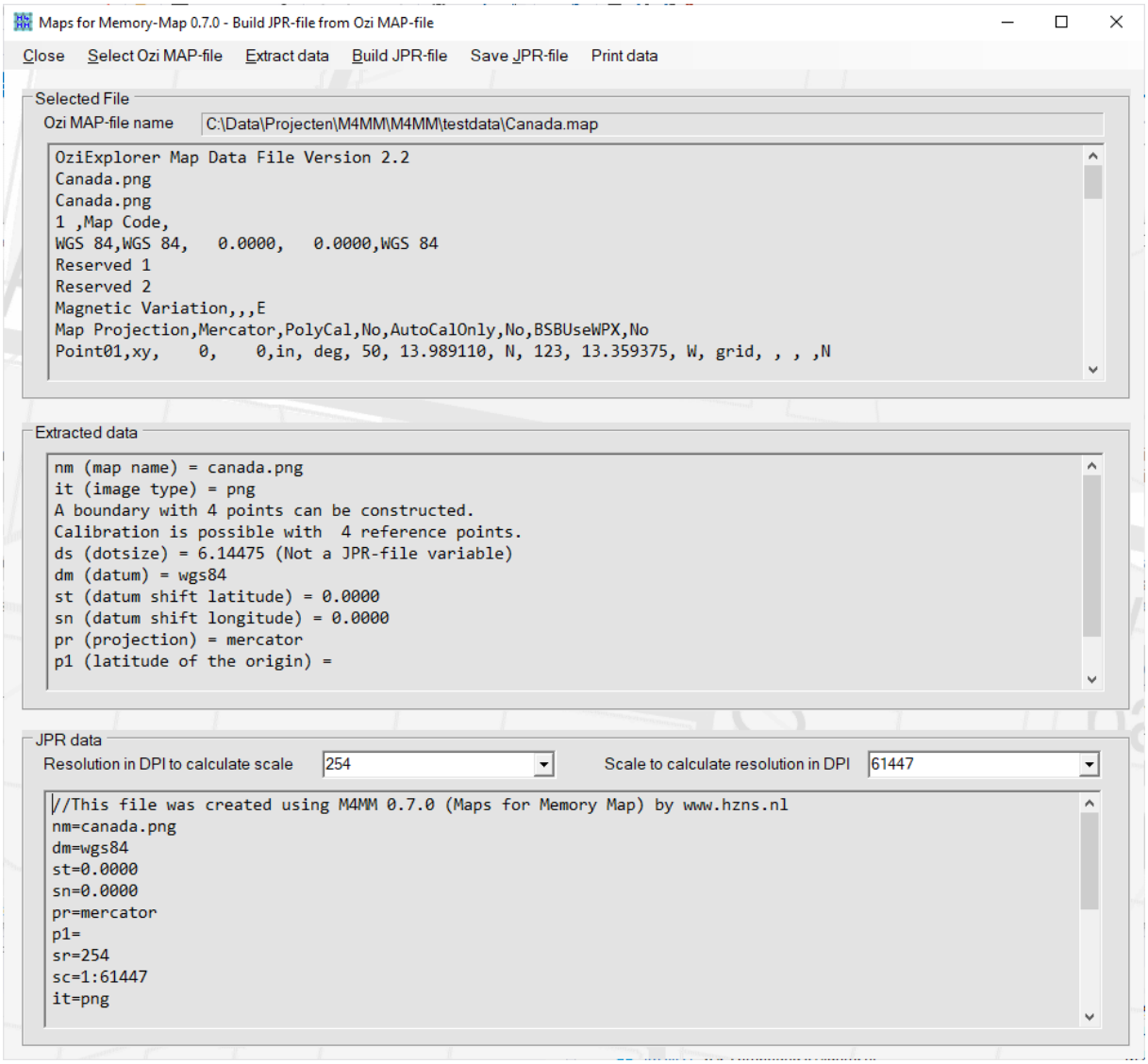
Form `PdfBluid` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`SourceFile`	`FileNameStructure`		Represents the file name of the MrSID-file.
`ImageFile`	`FileNameStructure`		Represents the file name of the file containing the image of a map (TIFF- or PNG-file.
`JprFile`	`FileNameStructure`		Represents the file name of the JPR-file
`GeoData`	String		Represents the WKT 2.0 content of the source-file
`GeoDataLen`	integer		represents the Length of `GeoData`-string.
`JprData`	`JprStructure`		Represents the `JprData` needed to build a JPR-file.
`ResData`	`ResStructure`		Represents the resolution data of an image.
`ReferencePoint(#)`	Array of `Coordinate`		Array representing the georeferenced points of a map.
`VertexPoint(#)`	Array of `Coordinate`		Array representing the boundary points of a map (= `vertexpoint` by FUGAWI).
`VertexAmount`	Integer		Represents the amount of `VertexPoint`.
`ImageData`	`ImageDataStructure`		Represents extracted data from `ImageFile`

Form `OziBuild`

Screen



Added Control to the form: a MenuStrip: `TopStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

`Function BuildJprContent(Trigger)`

Function builds the content for a JPR-file.
Input: `Trigger`; indication which subroutine called the function.
- `ini`: `BuildItem_Click` called
- `dpi`: `DpiBox_LostFocus` or `DpiBox_KeyPress` called
- `scale`: `ScaleBox_LostFocus` or `ScaleBox_KeyPress` called

Output: a string with the content for the JPR-file.

Subroutines

`Subroutine OziBuild_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Loading values into `DpiBox`.

Disabling `TopStrip` items except `CloseItem` and `SelectItem`.
Disabling groups.
Disabling `StartMemu`.

Subroutine `CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

Subroutine `OziBuild_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

Subroutine `SelectItem_Click`

Executed:
On demand (by user)
Actions:
Clearing all data in textboxes in form.
Selecting and opening a Ozi MAP-file.
Creating values for variables: `MapFile`, `GeoData`.
Presenting `MapFile`.
Presenting `GeoData`.
Enabling form elements for next step.

Subroutine `ExtractItem_Click`

Executed:
On demand (by user).
Actions:
Checking if the MAP-file is a OZI-file
Breaking `GeoData` up into lines (`GeoDataLine(#)`).
Retrieving `JprData`.
Creating value for variable: `JprFile`.

Retrieving `ReferencePoint(#)`.
Retrieving `VertexPoint(#)` (boundary).
Presenting extracted data and comment on these data.
Enabling form elements for next step.

Subroutine `BuildItem_Click`

Executed:
On demand (by user).
Actions:
Enabling `DpiBox` if enough data available.
Building content for the JPR-file by using function `BuildJprContent`.
Presenting content for the JPR-file.
Enabling form elements for next step.

Subroutine `SaveItem_Click`

Executed:
On demand (by user).
Actions:
Saving content for JPR-file into JPR-file.

`Subroutine DpiBox_LostFocus`
Executed:
On event lost focus.
Actions:
Recalculating Scale (`JprData.sc`).

`Subroutine DpiBox_KeyPress`
Executed:
On event key press.
Actions:
Recalculating Scale (`JprData.sc`) if `Enter` button is used.

`Subroutine ScaleBox_LostFocus`
Executed:
On event lost focus.
Actions:
Recalculating resolution (`JprData.sr`).

`Subroutine ScaleBox_KeyPress`
Executed:
On event key press.
Actions:
Recalculating resolution (`JprData.sr`) if `Enter` button is used.

`Subroutine SplitLine`
Executed:
On demand (by `ExtractItem_Click`).
Actions:
Spliting a line (string) with comma separated fields into max 20 fields.

`Subroutine PrintItem_Click`
Executed:
On demand (by user)
Actions:
Enables/disables `PrintDataItem_Click`, `ExtractedDataItem_Click` and `JprDataItem_Click` (no data in related textbox)
Condition: `PrintItem.CheckOnClick` must be `True`.

`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from MAP-file using `PrintContent` function (StartMenu).

`Subroutine ExtractedDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints Extracted data from MAP-file using `PrintContent` function (StartMenu).

`Subroutine JprDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints JPR-data from MAP-file using `PrintContent` function (StartMenu).

Variable structures

Form `Dummy` contains no declarations of variable structures.

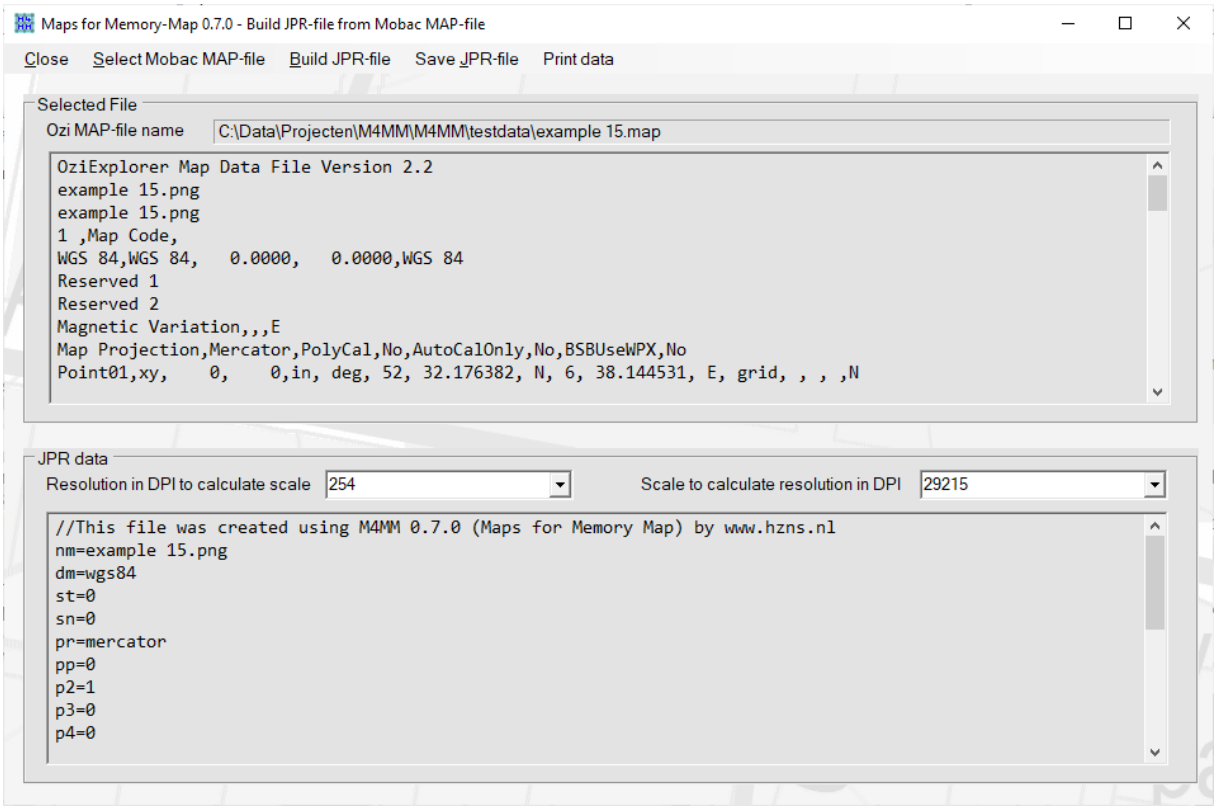
Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`MapFile`	`FileNameStructure`		Represents the file name of the Ozi MAP-file.
`ImageFile`	`FileNameStructure`		Represents the file name of the file containing the image of a map (graphics or OZF-file.
`JprFile`	`FileNameStructure`		Represents the file name of the JPR-file
`GeoData`	String		Represents the content of the MAP-file
`GeoDataLen`	integer		Represents the Length of `GeoData`-string.
`GeoDataLine(#)`	Array of strings		Array representing the content of each line in `GeoData`-string.
`GeoDataLineCount`	Integer		Represents the amount of lines in `GeoData`-string.
`GeoDataLineField(#)`	Array of strings		Array representing the content of each field in a `GeoDataLine`.
`JprData`	`JprStructure`		Represents the data needed to build a JPR-file.
`ResData`	`ResStructure`		Represents the resolution data of an image.
`ReferencePoint(#)`	Array of `Coordinate`		Array representing the georeferenced points of a map.
`ReferencePointCounter`	Integer		Represents the amount of `ReferencePoint`.
`ValidReferencePoints`	Integer		Represents the amount of valid `ReferencePoint`.
`VertexPoint(#)`	Array of `Coordinate`		Array representing the boundary points of a map (= `vertexpoint` by FUGAWI).
`VertexAmount`	Integer		Represents the amount of `VertexPoint`.
`ValidVertexPoints`	Integer		Represents the amount of valid `VertexPoint`.
`VertexMax`	Integer		Represents the maximum amount of `VertexPoint`.

`VertexPointLLCounter`	Integer		Represents the amount of `VertexPoint` based on latitude/longitude coordinate.
`VertexPointXYCounter`	Integer		Represents the amount of `VertexPoint` based on X/Y pixel coordinates.

Form MobacBuild

Screen



Added Control to the form: a MenuStrip: `TopStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

`Function BuildJprContent(Trigger)`

Function builds the content for a JPR-file.
Input: `Trigger`; indication which subroutine called the function.
- `ini`: `BuildItem_Click` called
- `dpi`: `DpiBox_LostFocus` or `DpiBox_KeyPress` called
- `scale`: `ScaleBox_LostFocus` or `ScaleBox_KeyPress` called

Output: a string with the content for the JPR-file.

Subroutines

`Subroutine MobacBuild_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Loading values into `DpiBox`.
Disabling `TopStrip` items except `CloseItem` and `SelectItem`.
Disabling groups.
Disabling `StartMemu`.

Subroutine `CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

Subroutine MobacBuild_Closing

Executed:
On closing form.
Actions:
Enabling StartMemu.

Subroutine SelectItem_Click

Executed:
On demand (by user)
Actions:
Clearing all data in textboxes in form.
Selecting and opening a Mobac MAP-file.
Creating values for variables: MapFile, GeoData.
Presenting MapFile.
Presenting GeoData.
Enabling form elements for next step.

Subroutine ExtractItem_Click

Executed:
On demand (by user).
Actions:
Checking if the MAP-file is a OZI-file
Breaking GeoData up into lines (GeoDataLine(#)).
Retrieving JprData.
Creating value for variable: JprFile.

Retrieving ReferencePoint(#).
Retrieving VertexPoint(#) (boundary).
Presenting extracted data and comment on these data.
Enabling form elements for next step.

Subroutine BuildItem_Click

Executed:
On demand (by user).
Actions:
Enabling DpiBox if enough data available.
Building content for the JPR-file by using function BuildJprContent.
Presenting content for the JPR-file.
Enabling form elements for next step.

Subroutine SaveItem_Click

Executed:
On demand (by user).
Actions:
Saving content for JPR-file into JPR-file.

Subroutine DpiBox_LostFocus

Executed:
On event lost focus.
Actions:
Recalculating Scale (JprData.sc).

Subroutine DpiBox_KeyPress

Executed:
On event key press.
Actions:
Recalculating Scale (JprData.sc) if Enter button is used.

`Subroutine ScaleBox_LostFocus`
Executed:
On event lost focus.
Actions:
Recalculating resolution (`JprData.sr`).

`Subroutine ScaleBox_KeyPress`
Executed:
On event key press.
Actions:
Recalculating resolution (`JprData.sr`) if `Enter` button is used.

`Subroutine SplitLine`
Executed:
On demand (by `ExtractItem_Click`).
Actions:
Spliting a line (string) with comma separated fields into max 20 fields.

`Subroutine PrintItem_Click`
Executed:
On demand (by user)
Actions:
Enables/disables `PrintDataItem_Click` and `JprDataItem_Click` (no data in related textbox)
Condition: `PrintItem.CheckOnClick` must be `True`.

`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints WKT 2.0 from MAP-file using `PrintContent` function (StartMenu).

`Subroutine JprDataItem_Click`
Executed:
On demand (by user)
Actions:
Prints JPR-data from MAP-file using `PrintContent` function (StartMenu).

Variable structures
Form `MobacBuild` contains no declarations of variable structures.

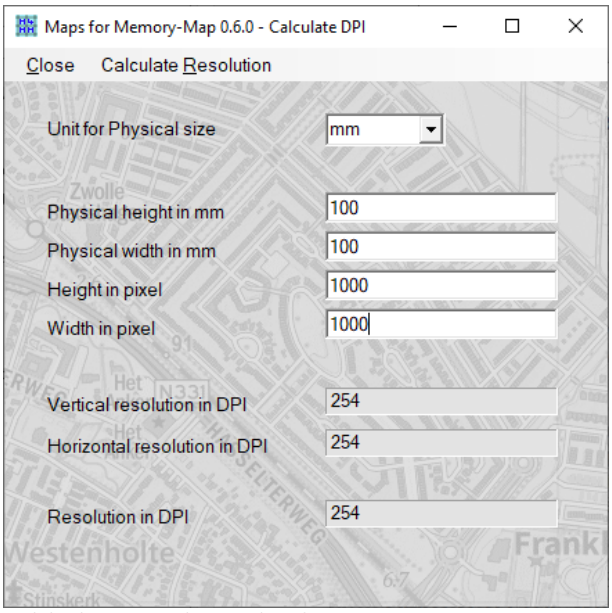
Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`MapFile`	`FileNameStructure`		Represents the file name of the MOBAC MAP-file.
`JprFile`	`FileNameStructure`		Represents the file name of the JPR-file
`GeoData`	String		Represents the content of the MAP-file
`GeoDataLen`	integer		Represents the Length of `GeoData`-string.

GeoDataLine(#)	Array of strings		Array representing the content of each line in GeoData-string.
GeoDataLineCount	Integer		Represents the amount of lines in GeoData-string.
GeoDataLineField(#)	Array of strings		Array representing the content of each field in a GeoDataLine.
JprData	JprStructure		Represents the data needed to build a JPR-file.
ReferencePoint(#)	Array of Coordinate		Array representing the georeferenced points of a map.

Form DpiCalc

Screen



Added Controls to the form:
A MenuStrip: `TopStrip`.

Functions

Form `DpiCalc` contains no functions.

Subroutines

`Subroutine DpiCalc_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Loading values into `UnitBox`.
Setting Values for `PhyHeightLabel.Text` and `PhyWidthLabel.Text`
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine DpiCalc_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine PhyHeightValue_KeyPress`

Executed:
On keypress on textbox `PhyHeightValue`
Actions:
Handles digits, control, dot and enter keystrokes

Subroutine PhyWidthValue_KeyPress
Executed:
On keypress on textbox PhyWidthValue
Actions:
Handles digits, control, dot and enter keystrokes

Subroutine PixHeightValue_KeyPress
Executed:
On keypress on textbox PixHeightValue
Actions:
Handles digits, control, dot and enter keystrokes

Subroutine PixWidthValue_KeyPress
Executed:
On keypress on textbox PixWidthValue
Actions:
Handles digits, control, dot and enter keystrokes

Subroutine CalculateItem_Click
Executed:
On demand (by user)
Actions:
Calculates VerResValue, HorResValue and ResValue if PhyHeightValue, PhyWidthValue, PixHeightValue and PixWidthValue are filled. Takes value of UnitBox in account.

Subroutine PhyHeightValue_TextChanged
Executed:
On TextChanged on textbox PhyHeightValue
Actions:
Calling subroutine ClearResolutions and ChangeUnit.

Subroutine PhyWidthValue_TextChanged
Executed:
On TextChanged on textbox PhyWidthValue
Actions:
Calling subroutine ClearResolutions and ChangeUnit.

Subroutine PixHeightValue_TextChanged
Executed:
On TextChanged on textbox PixHeightValue
Actions:
Calling subroutine ClearResolutions and ChangeUnit.

Subroutine ClearResolutions
Executed:
Called by PhyHeightValue_TextChanged, PhyWidthValue_TextChanged, PixHeightValue_TextChanged and PixWidthValue_TextChanged,
Actions:
Resetting HorResValue, VerResValue and ResValue.

Subroutine ChangeUnit
Executed:
Called by PhyHeightValue_TextChanged, PhyWidthValue_TextChanged, PixHeightValue_TextChanged, PixWidthValue_TextChanged, and UnitBox_TextChanged
Actions:

Handling/Changing labels `PhyHeightLabel` and `PhyHeightLabel` based on value of `UnitBox`.

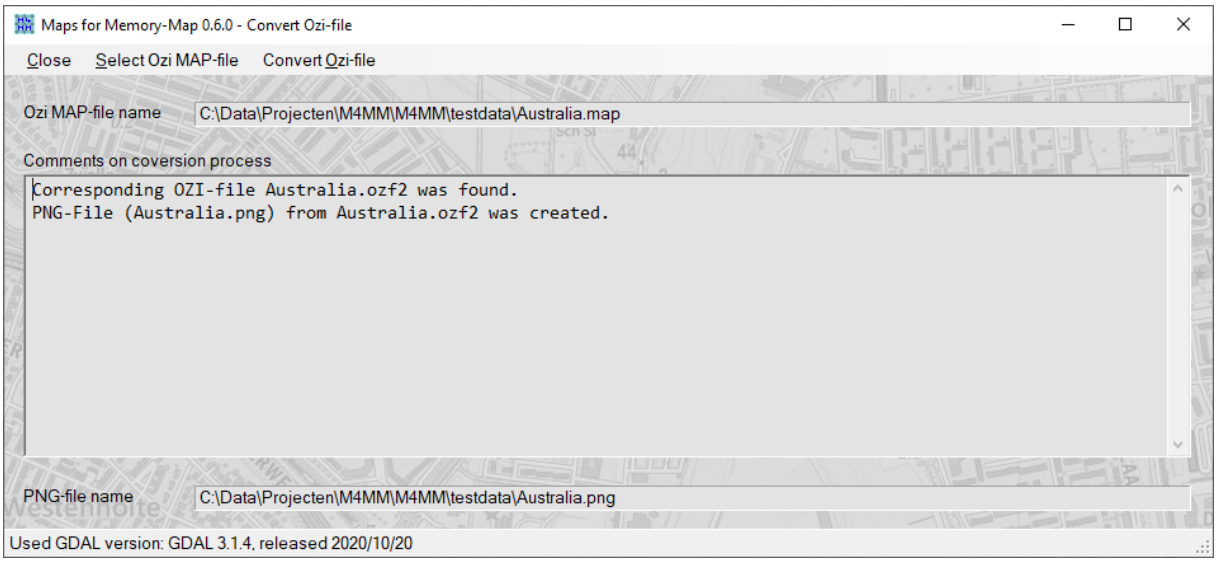
`Subroutine UnitBox_TextChanged`
Executed:
On change of value of `UnitBox` by user.
Actions:
Calling `CalculateItem_`Click and `ChangeUnit`.

Variable structures
Form `CalcDpi` contains no declarations of variable structures.

Variables on form level
Form `CalcDpi` contains no declarations of variables.

Form OziConv

Screen



Added Controls to the form:
A MenuStrip: `TopStrip`.
A StatusStrip: `BottomStrip`.

Functions

Form `OziConv` contains no functions.

Subroutines

`Subroutine OziConv_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Setting `MapFileValue`, `TargetFileValue` and `CommentBox` to “”
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine OziConv_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`

Executed:
On demand (by user)
Actions:
Clearing all data in textboxes in form.
Selecting and opening a OziExplorer MAP-file.
Checking for a corresponding OZF2/OZFX3-file
Creating values for variables: `SourceFile`, `TargetFile`.
Enabling form elements for next step.

`Subroutine ConvertItem_Click`
Executed:
On demand by user
Actions:
Creating `CommandLine` which contains the shell command to translate MAP/OziExplorer file.
Executing `CommandLine`.
Retrieving GDAL version.

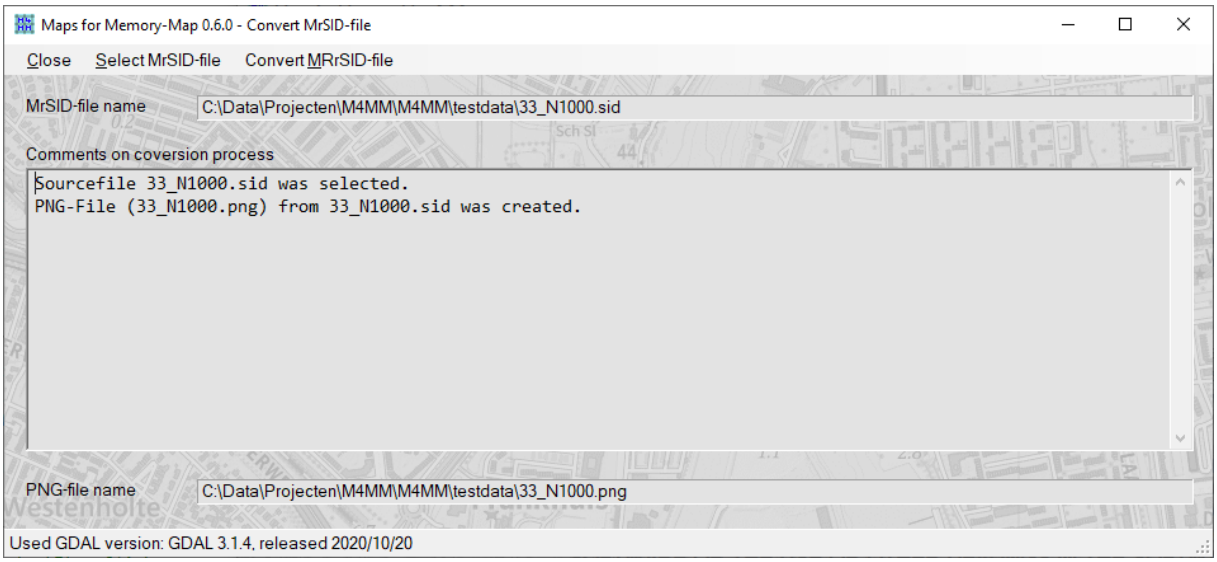
Variable structures
Form `OziConv` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`MapFile`	`FileNameStructure`		Represents the file name of the OziExplorer MAP-file.
`SourceFile`	`FileNameStructure`		Represents the file name of the OziExplorer OZF2-/OZFX3-file.
`TargetFile`	`FileNameStructure`		Represents the file name of the result file of a file conversion.

Form SidConv

Screen



Added Controls to the form:
A MenuStrip: `TopStrip`.
A StatusStrip: `BottomStrip`.

Functions

Form `SidConv` contains no functions.

Subroutines

`Subroutine SidConv_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Setting `SourceFileValue`, `TargetFileValue` and `CommentBox` to ""
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Initializing closing of this form.

`Subroutine SidConv_Closing`

Executed:
On closing form.
Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`

Executed:
On demand (by user)
Actions:
Clearing all data in textboxes in form.
Selecting and opening a SourceFile MrSID-file.
Creating values for variables: `SourceFile`, `TargetFile`.
Enabling form elements for next step.

`Subroutine ConvertItem_Click`

Executed:
On demand by user
Actions:
Creating `CommandLine` which contains the shell command to translateMrSID-file.
Executing `CommandLine`.
Retrieving GDAL version.

Variable structures

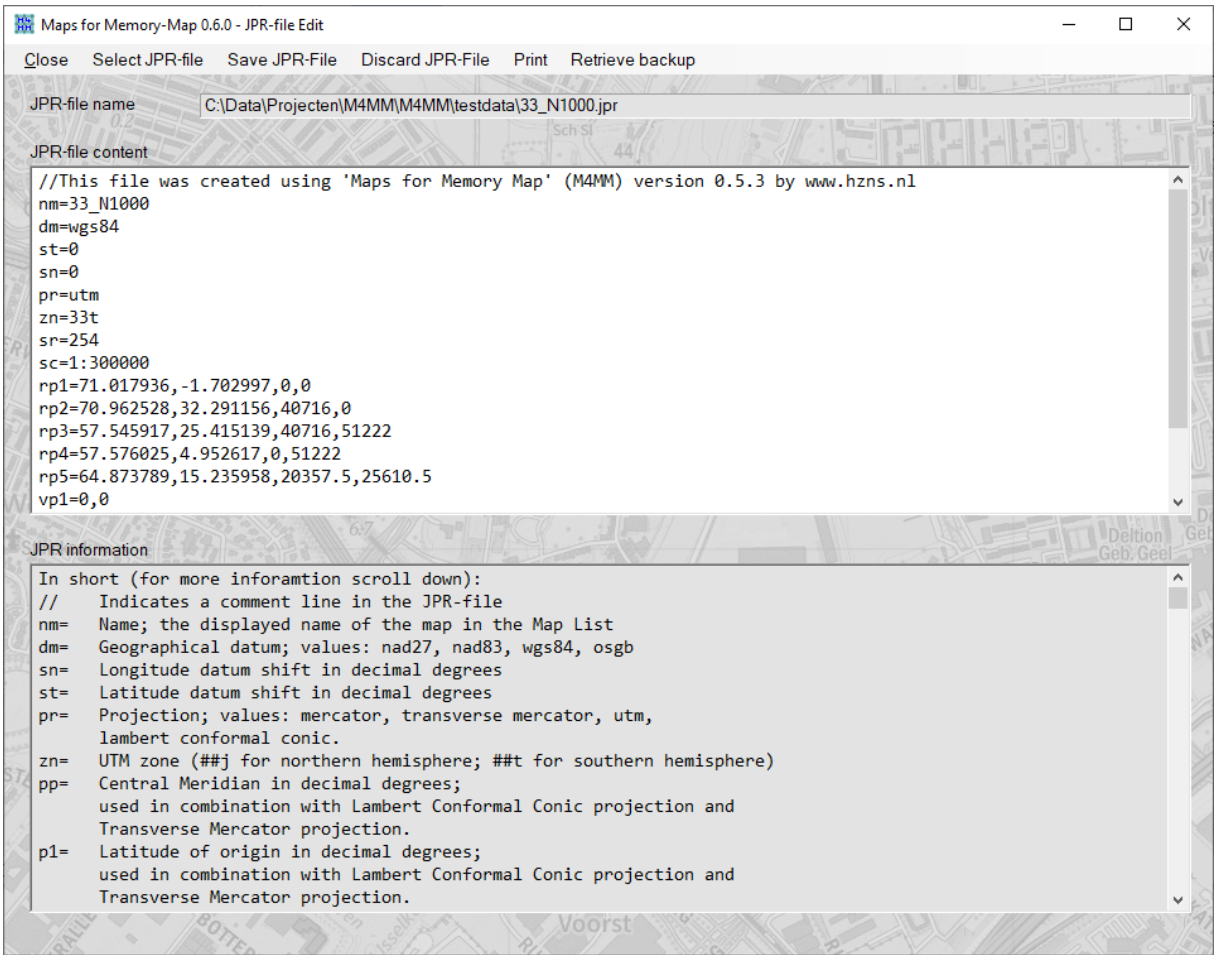
Form `SidConv` contains no declarations of variable structures.

Variables on form level

Name of variable	Type of variable	Value of variable	Description/Remarks
`SourceFile`	`FileNameStructure`		Represents the file name of the MrSID-file.
`TargetFile`	`FileNameStructure`		Represents the file name of the result file of a file conversion.

Form JprEdit

Screen



Added Control to the form: a MenuStrip: `TopStrip`.
Value `PrintItem.CheckOnClick` is `True`.

Functions

Form `JprEdit` contains no functions.

Subroutines

`Subroutine JprEdit_Load`

Executed:
On loading form
Actions:
Enables and sets the tooltips for this form.
Disabling `TopStrip` items except `CloseItem` and `SelectItem`
Disabling `StartMemu`.

`Subroutine CloseItem_Click`

Executed:
On demand (by user)
Actions:
Checking if changes in `DataBox` are saved and do so if requested
Initializing closing of this form.

`Subroutine JprEdit_Closing`

Executed:
On closing form.

Actions:
Enabling `StartMemu`.

`Subroutine SelectItem_Click`
Executed:
On demand (by user)
Actions:
Clearing all data in textboxes in form.
Selecting and opening a JPR-file.
Creating values for variables: `JprFile`, `BakFile` and `StartText`.
Enabling form elements for next step.

`Subroutine SaveItem_Click`
Executed:
On demand (by user)
Actions:
Creating a backup file of the original JPR data.
Saving edited data to a new JPR-file.
Resetting `StartText`.

`Subroutine DiscardItem_Click`
Executed:
On demand (by user)
Actions:
Clearing all values in form.

`Subroutine RetrieveItem_Click`
Executed:
On demand (by user)
Actions:
Selecting BAK-file
Clearing all values in form
Creating values for variables: `JprFile`, `BakFile` and `StartText`.
Enabling all form elements.

`Subroutine PrintDataItem_Click`
Executed:
On demand (by user)
Actions:
Building pages based on status `LandscapeItem` (`Checked = true/false`).
Selecting printer using `DialogToPrint`.
Printing page by page using `DocumentToPrint`.

`Subroutine DocumentToPrint_PrintPage`
Executed:
By `PrintDataItem_Click`
Actions:
Parsing text and parameters to `DocumentToPrint` to print a page.

`Subroutine LandscapeItem_Click`
Executed:
On demand (by user).
Actions:
Changing `Checked` status.

Variable structures
Form `JprEdit` contains no declarations of variable structures.

Build

“The cookbook”

“The cookbook” was created using Microsoft Word 2013 and saved as a PDF-file with bookmarks to navigate through the content. The intention is that can be read from paper and from screen.

The application M4MM

The application was built with Microsoft Visual Studio Community 2019 and ported to Visual Studio Community 2022. The coding was done with Visual basic Windows Forms Application (.NET Framework, version 4.8.04161).

The application will be compiled with the newest release of Microsoft Visual Studio Community 2022 (Version 17.2.5) at the moment of publication July 1, 2022).

The following parameters were set:

- Assembly option: Enable application framework: on
- Assembly option: Save My.Settings on shutdown: on
- Compile option: Option explicit: on
- Compile option: Option strict: off
- Compile option: Option infer: on
- Compile option: Option compare: binary
- Compile option: Target CPU: X64

In this part the code will not be explained line by line. The solution files are included in the project. In all forms and the module `common` contain comment lines with explanations.

To view/read these files You do not need to install Visual Sudio, you only need a simple editor. Sophisticated code editors (like Notepad++ or Code Writer) do have some features which make it essayer. What can you expect? The explanation of some specific features.

In the project the same functionality aren't sometime implemented in the same way. For example sometimes controls are added by using `dim <name> as new control` and sometimes controls are added to the form.

The creating of the application M4MM was (and is) an iterative process; designing > building > testing. In some cases there was a lot of trial and error to find a right solution. An effort has been made to write readable and understandable code. In some cases the code isn't "fully Visual Basic" because it is "re-used" from code snippets ("do not change functioning code to much!).

The following "features" are described:

- File operations
- Extracting data from image file
- Printing the content of a textbox
- Splitting a string into substrings
- Using a shell command
- Special directories
- From NEATLINE to boundary
- Relation between Scale, Resolution and Pixel size

File operations

One of the main "Activities" in M4MM is handling files.

- Selecting a file
- Reading data text from a file
- Creating/Saving a file

Selecting a File

Before handling a file, the file must be selected. To do so a Select a `OpenFileDialog` must be added to the project:

```
Dim SelectFile As New OpenFileDialog()
```

After adding some parameters must be added:

```
SelectFile.Filter = "Graphics (*.png,*.tif)|*.png; *.tif;
   *.tiff|PNG-Files (*.png)|*.png|TIFF-files (*.tif)|*.tif; *.tiff"
SelectFile.Multiselect = False
```

The `Filter` is set to narrow the search for file and `Multiselect` is set to `False` to make sure only one file can be selected.

Next step is to activate the `OpenFileDialog`:

```
...SelectFile.ShowDialog()
```

The result if the dialog is a filename (`SelectFile.FileName`).

Remark:

- Be aware of handling the case no file is selected.

Reading data from a text file

To read text from a text file the `ReadAllText` method is used. Taken for the manual the basic structure for this method is:

```
My.Computer.FileSystem.ReadAllText(FileName)
```

In the application the method is implemented as followed to copy the text of the file to a textbox:

```
DataBox.Text = My.Computer.FileSystem.ReadAllText(SelectFile.FileName)
```

Creating/ Saving a file

To write text from a text file the `WriteAllText` method is used. Taken for the manual the basic structure for this method is:

```
My.Computer.FileSystem.ReadAllText(FileName, Text, Append)
```

In the application the method is implemented as followed to copy the text of a textbox to a file:

```
My.Computer.FileSystem.WriteAllText(TextFileName, DataBox.Text, False)
```

Remarks:

- The Parameter `Append` is set to `False` to overwrite te content of the text file is it already exists.
- In some cases the user is asked to confirm to allow overwriting (using a `MsgBox`).
- In some cases a backup file will be created (form `JprEdit`).

Extracting data from image file

Memory-Map can only handle images with a color depth of 8 bit or less. The purpose of form `ImageInfo` is to explore some parameters of an image of which color depth is one of them. To extract this information the image must be open by the form. Taken for the manual the basic structure for this method is:

```
Image.FromFile(String)
```

In the application the method is implemented as followed:

```
Public MapImage As Image
..
..
MapImage = Image.FromFile(ImageFileName)
```

Height, Width, VerticalResolution, HorizontalResolution and PixelFormat are properties of an Image. To Retrieve them:

```
Height = MapImage.Height
Width = MapImage.Width
ColorDepth = MapImage.PixelFormat.ToString
ResolutionX = MapImage.HorizontalResolution
ResolutionY = MapImage.VerticalResolution
```

Remarks:

- The color depth is retrieved as a (understandable) string (.ToString). Only images with Format8bppIndexed, Format4bppIndexed and Format1bppIndexed can be imported into Memory-Map.
- Based on some experiments the maximum size of about 2 G byte of pixel information (BMP). By using an error handling procedure M4MM avoid opening larger images.

```
Function ...

   '- Initiating error handling
      (If image to large the Error kicks in)
   On Error GoTo Errorhandler

   '- Loading image (if not loaded > to errorhandler)
   TempImage = Image.FromFile(ImageFileName)

   Exit function

Errorhandler:
   .
   .
   .
End Function
```

Printing the content of a textbox

In earlier days printing was simple, just a print command and the job was done. Now a days not any more. Although the implementation in M4MM is not the most elegant solution, it does its job. M4MM prints every page of a text as a single print job. The implication of this choice is that printing to a file (like a PDF-files) isn't preferable (you must assign a filename to every page).

The first step is adding a PrintDialog (SelectPrinter) to the module Common and a PrintDocument (PrintString) to the form StartMenu.

The second step is creating functionality to build pages with a specific maximum number of lines and a maximum number of characters (Function ConvertToMaxLineSize and Subroutine BuildPages, both in the form StartMenu).

The third step is adding a subroutine PrintContent (in the module Common) which initiates the printer selection using the PrintDialog (SelectPrinter) control and prints the text page by page (from step 2) using the PrintDocument (PrintString) control.

Splitting a string into substrings

One of the features which is used many times is splitting a string into substrings. This is done three different ways. The first one is using the method `Strings.Split`, the second is using the method `Strings.InStr` and the last is a "walk through" as string, character by character, form the first character forwards or from the last character backwards.

The Method `Strings.Split`

Taken for the manual the basic structure for this method is:

```
Strings.Split(String, String, Int32, CompareMethod)
```

In the application the method is implemented as followed:

```
Dim SubStringArray() As String = Split(SourceString, SplitString,)
```

In the manual is written the `SplitString` is a one character string. In the application is also a two character string used (Carriage Return (`Chr(13)`) and Line Feed (`Chr(10)`)) to separate text lines in a string. It worked well.

Remarks:

- Be sure the `SourceString` contains the `SplitString` to prevent errors (test for example with `Strings.InStr` method).
- The `SubStringArray()` starts with `SubStringArray(0)` and end with `SubStringArray(SubStringArray.Count - 1)`.
- To use the " character use `Chr(34)` as `SplitString`.
- If you want to use a `Split` method within a function or subroutine more than ones, create every time a new `SubStringArray()`.
- The `SplitString` isn't included in the `SubString`.

Method `Strings.InStr`

Taken for the manual the basic structure for this method is:

```
Strings.InStr(String, String, CompareMethod)
```

In the application the method is implemented as followed:

```
SplitPoint = InStr(SourceString, SplitString, CompareMethod.Binary)
```

The `SplitPoint` is the position in the `SourceString` where the `SplitString` begins. Use the method `Stings.Left` to create the string left of the `SplitPoint` and the method `Strings.Right` in combination of the method `Strings.Len()` to create the right string.

Remarks:

- Be sure the `SourceString` contains the `SplitString` to prevent errors (test for example with `Strings.InStr` method).
- To use the " character use `Chr(34)` as `SplitString`.

Method `Strings.Mid`

Taken for the manual the basic structure for this method is:

```
Strings.Mid(String, Integer, integer)
```

In the application the method is implemented as followed:

```
Strings.Mid(SourceString, CharacterPointer, 1)
```

This method is used in combination with a `For ... Next` loop.

```
For CharacterPointer = 0 to Strings.Len(SourceString) - 1
   If Strings.Mid(SourceString, CharacterPointer, 1) = SplitString then
      SplitPoint = CharacterPointer
      Exit For
   End if
Next
```

In the example above a `SplitString` is used with a length of 1.

Use the method `Stings.Left` to create the string left of the `SplitPoint` and the method `Strings.Right` in combination of the method `Strings.Len()` to create the right string.

Remarks:

- Be sure the `SourceString` contains the `SplitString` to prevent errors (test for example with `Strings.InStr` method).
- To use the " character use `Chr(34)` as `SplitString`.
- You can use a walk back (from end to start) through a string a reversed `For ... Next` loop.
  `For CharacterPointer = Strings.Len(SourceString)- 1 to 0 step - 1`

Using a shell command

Taken for the manual the basic structure for this method is:

```
Interaction.Shell(String, AppWinStyle, Boolean, Int32)
```

In the application the method is implemented as followed:

```
Shell(CommandLine, AppWinStyle.Hide, Wait:=True)
```

Or

```
Shell(CommandLine, AppWinStyle.NormalFocus, Wait:=True)
```

`Wait` is set to `True` as a precaution. The `Shell` command must be finished before it "hands over" the control to M4MM.

Normally the `AppWinStyle` is set to `Hide`. Only in those cases where the execution of the `Shell` command takes a noticeable period of time it is set to `NormalFocus` (to show the progress of the called external application).

When `GdalInfo` is used, the result will be written to a temporary file.

Remarks

- In M4MM parameters with a file path must be enclosed by ". To do so use `Chr(34)`. An example below:
  ```
  CommandLine = Chr(34) & BatFile.LongName & Chr(34) & " gdalinfo.exe
  -- version > " & Chr(34) & TempFile.LongName & Chr(34)
  ```
- Using Python with Python scripts isn't stable.

Special directories

M4MM uses temporary files to save data during executions of procedures. It also stores the location of `OSgeo4W.bat` in the file `m4mm.ini`, which is locate in users document directory.

To retrieve these directories M4MM uses the `SpecialDirectories` property.

Users document directory:

```
My.Computer.FileSystem.SpecialDirectories.MyDocuments.
```

Users temporary directory:

```
My.Computer.FileSystem.SpecialDirectories.Temp.
```

From NEATLINE to boundary

A Memory-Map boundary is based on pixel coordinates. GeoTIFF-files and GeoPDF-files may contain a NEATLINE, a polygon around a 'area of interest'. This NEATLINE can be used for the same purpose as boundary (not always). If the NEATLINE is defined in a Cartesian coordinate system it can be translated into a boundary.

Converting a coordinate from one to another Cartesian grid system is can be done using geometry or mathematics (More information).

The method used in M4MM is based on converting polar coordinates. Before the NEATLINE coordinates can be converted two important parameters musty be calculated.

- In the first step to convert the NEATLINE coordinates is to calculate two parameters. The first one is relation in the units of length in both coordinate systems. This is the pixel size (`JprData.ps`/size of a pixel in units of the coordinate system).
- The second parameter is the angle between both coordinate systems. by calculating the angle from the upper left reference point to the upper right reference point using the geographical coordinates. This angle in the pixel coordinate system is 0 (by definition).

The following step must be done for every NEATLINE coordinate except the last one (it is the same as the first one.

- The first step is to convent the NEATLINE coordinate to polar coordinates base on the geographical system. The result is an angle and a distance.
- The second step is to convert the geographical distance to the pixel distance using the pixel size.
- The third step is to convert the geographical angle to a pixel angle using the angle between the coordinate systems. Be aware the orientation of the pixel system is flipped compared to the geographical system.
- Step four. The result of the second and third step is a polar coordinate based on the pixel system. By converting this polar coordinate to pixel coordinates (X,Y) the NEATLINE coordinate has been converted to a boundary coordinate.

The calculation of the distance between two coordinates is based on the Pythagoras theorem. The angle is based on calculations using sine and cosine. To calculate X,Y coordinates from a polar coordinate the functions arcsine and arccosine is used.

The conversion from NEATLINE to boundary is use in the forms `PdfBuild` and `TiffBuild` (in the subroutine `NeatToVertex()`). In the module Common the function `PixelSize()` is found to calculate the size of pixel in units of the geographical coordinate system.

Relation between Scale, Resolution and Pixel size

The relation is already describe in "Relation between scale, resolution and pixel size".

The data structure `JprData` has two 'types' of dot size/pixel size:
The pixel size (`JprData.ps`) is the size of one pixel in units of a Cartesian coordinate system.
The dot size (`JprData.ds`) is the size of one pixel in meter.

In the forms `PdfBuild`, `TiffBuild`, `SidBuild`, `OziBuild` and `MobacBuild` there are comboboxes `DpiBox` and `ScaleBox` in which the DPI or the scale of the image can be changed. These comboboxes are interactive, when DPI changes scale changes and vice versa. The `DpiBox` and `ScaleBox` are only available when the dot size (`JprData.ds`) could be established.

In the module `Common` you will find the function `PixelSize` calculates the pixel size based on the pixel and grid coordinates of the image corners.

Although the resolution, the scale and the pixel size are calculated as accurate as possible they must be considered as estimated values, due of averaging and rounding. There for these values will be rounded to whole numbers.

Testing

The testing has two separated phases. The first one is testing as a part of the design, build test cycle. The second is the final testing.

"The cookbook" is tested by walking through all the described procedures including the procedure steps in the third party applications.

The application M4MM is tested using "The cookbook" as "test scenario" with the most important question "Does M4MM what is expected to do?"

The development of application M4MM (version 0.6.0) and so "The cookbook" (version 0.6.0) was frozen at July 15. 2021.

The development of application M4MM (version 0.7.0) and so "The cookbook" (version 0.7.0) was frozen at October 19. 2021.

The development of application M4MM (version 0.8.0) and so "The cookbook" (version 0.8.0) was frozen at November 21. 2021.

The development of application M4MM (version 0.9.0) and so "The cookbook" (version 0.9.0) was frozen at March 01. 2022.

The development of application M4MM (version 1.0.0) and so "The cookbook" (version 0.9.0) was frozen at June 13. 2022.

Testing "The cookbook"

During the development of "The cookbook" the mentioned procedures were tested using the following releases of third party applications.

Version of "The cookbook"	Application	Version application
0.6.0	Memory-Map	5.4.4 (build 1113) and 6.4.3 (build 1278)
	QGIS	3.10.14 (LTR), 3.16.8 (LTR), 3.18.3 and 3.20.0
	MOBAC	2.1.4 and 2.2.0
	Paint.Net	3.2.15 and 3.2.16
	PDF-Exchange Editor	9.0 (build 352.0) and 9.1 (build 355.0)
0.7.0	Memory-Map	5.4.4 (build 1113), 6.4.3 (build 1278)
	QGIS	3.16.11 (LTR) and 3.20.3
	MOBAC	2.2.0
	Paint.Net	4.3.2
	PDF-Exchange Editor	9.2 (build 357.0)
0.8.0	Memory-Map	5.4.4 (build 1113), 6.4.3 (build 1278)
	QGIS	3.16.11 (LTR) and 3.22.0
	MOBAC	2.2.0

	Paint.Net	4.3.2
	PDF-Exchange Editor	9.2 (build 357.0)
0.9.0	Memory-Map	5.4.4 (build 1113), 6.4.3 (build 1278)
	QGIS	3.22.04 (LTR) and 3.24.0
	MOBAC	2.2.2 (build 2628)
	Paint.Net	4.3.8
	GIMP	2.10.30
	PDF-Exchange Editor	9.2 (build 359.0)
1.0.0	Memory-Map	5.4.4 (build 1113), 6.4.3 (build 1278)
	Gdal	3.1.4 and 3.4.3
	QGIS	3.22.7 (LTR) and 3.24.3
	MOBAC	2.2.3.2 (build 2678)
	Paint.Net	4.3.11
	GIMP	2.10.30
	PDF-Exchange Editor	9.3 (build 361.0)

All described functions and procedures were tested.

Testing the application M4MM

Version of M4MM	Application	Version application
0.6.0	Gdal	3.0.4, 3.1.4 and 3.3.0
	QGIS	3.10.14 (LTR), 3.16.8 (LTR), 3.18.3 and 3.20.0
0.7.0	Gdal	3.3.1
	QGIS	3.16.11 (LTR) and 3.20.3
0.8.0	Gdal	3.3.2
	QGIS	3.16.11 (LTR), 3.22.0
0.9.0	Gdal	3.1.4 and 3.4.1
	QGIS	3.22.4 (LTR), 3.24.0
1.0.0	Gdal	3.1.4 and 3.4.3
	QGIS	3.22.7 (LTR) and 3.24.3

All described functions were tested and worked as intended. Following issues remain:

- Printing in “one go” instead of page by page.
- Handling character “)” in shell command.