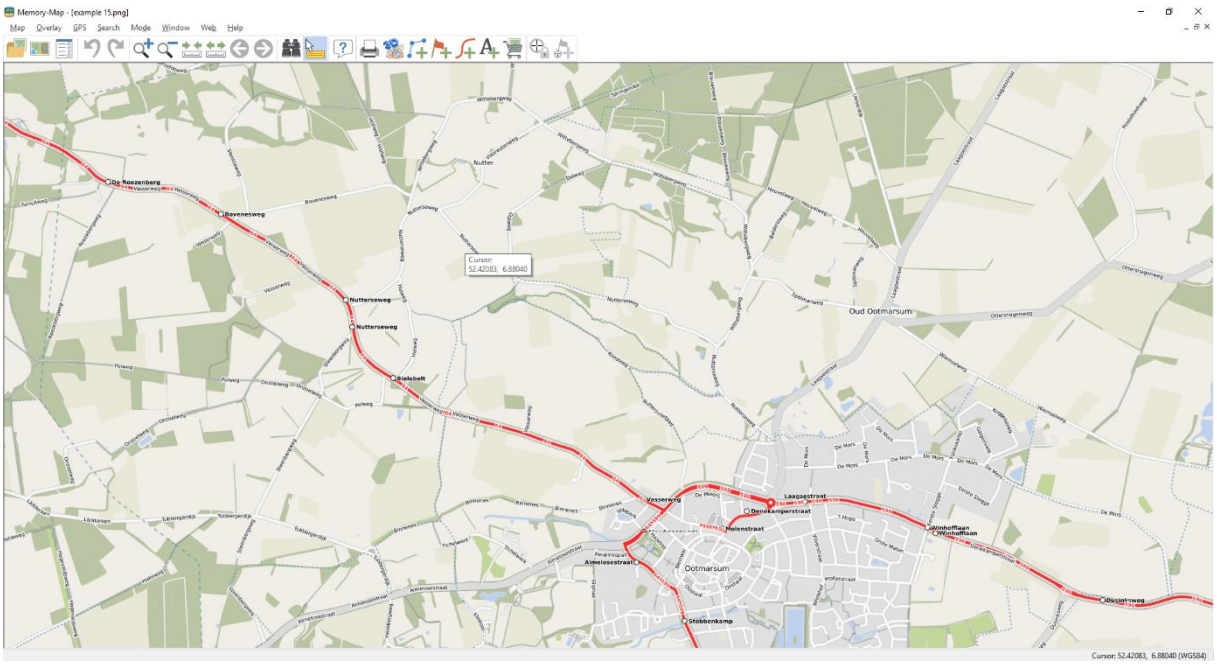
Project Maps for Memory-Map (M4MM)

Description of the tools

Map to JPR (M2J)
MOBAC to JPR (MOB2J)
GdalInfo to JPR (G2J)



Version: 0.2 (November 1, 2019)

(J.A. Kok, info@hzns. nl)

Inhoud

Change log

Version	Date	Remarks
0.1	October 1, 2019	Trial version
0.2	November 1, 2019	Initial version, partly replaces “Documentation MAP to JPR (v0.1)”

Introduction

Why (this project)

As a Memory-Map® user I am keen to collect maps which I can use with Memory-Map. Sources with direct to use maps are scarce. A second option is to calibrate maps by myself, which is a time consuming process with a lot of possibilities to make mistakes. For me, it is the last resource, if there is nothing else left. The third way is looking for maps which are already calibrated, but not in a proper format (to use with Memory-Map) and convert these data to an expectable format. This option to convert exiting calibration data was the start of the project "Map for Memory-Map" (M4MM); to develop conversion tools. Although projects primary goal is Memory-Map, FUGAWI users aren't excluded.

Strolling the internet, I found more or less three convertible map sources:

- Maps in a graphics format (tiff/jpeg/png/gif/bmp) with a OziExplorer® calibration file (MAP),
- GeoTiff maps and
- GeoPDF maps.

I also found Mobile Atlas Creator (MOBAC®) which can create .PNG files with an associated MAP-file (OziExplore option).

For this moment I focus on these four options.

A simple conversion tool is more or less: "translate A in B" and that's it. Mostly it isn't that simple. A is not exactly the same as B or there isn't always a B or there isn't an A for every B or vice versa, etc., etc. The best way describing a conversion tool is "a mix of tubes, pipes and flexible hoses".

Juggling with maps and building small applications are some of my hobbies. I'm just an amateur. Although this document is a "kind of" acknowledgement, it hasn't structure of an information jungle like a good professional application. It is more or less a story.

The conversion tools will be build Visual Studio 2019 (Visual Basic) from Microsoft®. The best reason: I am familiar with this program.

Structure of the documentation

When complete the documentation for this project will of contain:

- Decryption of the tools of the Project Maps for Memory-Map (M4MM) (this document).
- Appendix A with some tables.
- Appendix B with the Visual-Basic.Net code (Visual Studio 2019) for each of the tools.
- Appendix C1 to C3 with workflows and manuals for the separate applications.
- Appendix D1 to D3 with Quick starts for the separate applications.

Development stages of the tools.

I divided the development in five stages:

- The first development stage was "getting a clear picture" of the receiving side: data structure, logical and real names and aloud values of the variables in the JPR-file.
- The second stage was nearly the same excise, but for the sending side (OziExplorer MAP-file, GeoTIFF and GeoPDF).
- The third stage was comparing both sides to determine the crossover, the designing the conversion rules ("designing the tubes, pipes and flexible hoses").
- The fourth stage was the coding en testing ("constructing and fitting the tubes, pipes and flexible hoses").
- The last stage was the documentation (the most annoying but necessary job, nobody likes).

Sources of information

The receiving side

Memory-Map uses two kinds of maps. The first one are the proprietary maps the extensions .QCT or .QC3. The structure is disclosed. These files were not use for the development of the tools.
The second kind are graphics files of maps with an associated JPR-file. The JPR-file was developed by FUGAWI®. It is a readable and editable file. FUGAWI published a paper "JPR file format Version 1.18" which describes the structure, variables and aloud values. Memory-Map uses a subset of these structure, variables an values. This paper is the base for de receiving side of the tool.

Additional the documentation of PathAway®, taken from their website, and the Memory-Map helpfiles were used to verify and deepening the FUGAWI paper. Memory-Map, FUGAWI and PathAway are commercial products. Most of their detailed documentation is disclosed and you will not find it on the internet. No part of the software of these three programs were reverse-engineered, decompiled or disassembled to create the tools (sorry about this legal statement).

The sending side OziExplorer (and MOBAC)

To analyze the structure, variables and values of the OziExplorer MAP-files I used the OziExplorer help files, the MAP-files which came with the installation of OziExplorer, maps (with MAP-file) I found on the internet and some maps I created with MOBAC.

The sending side GeoTIFF and GeoPDF files

Some years ago, I found the program QGIS® which had the possibility to view/extract the Geographical data form GeoTIFF and GeoPDF files. With a sorrow look under the bonnet I found a command line program gdalinfo.exe which did the trick. With the document '[Geographic information - Well known text representation of coordinate reference systems]', the online documentation of Gdal® and a lot of GeoTIFF and GeoPDF maps form the internet I was able to draw a picture of the structure, variables and values in GeoTIFF and GeoPDF files.

Map used testing the tools.

The maps which were used to test the tools were taken from:

- The website of eTopo® (http://www.etopo.ca/)
- Oziexplorer® (www.oziexplorer.com) which came with the installation
- The website of Kortforsyningen Denmark (https://download.kortforsyningen.dk/content/geodataprodukter)
- The website of OSGB OpenData (https://www.ordnancesurvey.co.uk/opendatadownload/products.html)
- The website of USGS/US Topo (National map) (https://viewer.nationalmap.gov/basic/)
- The website of Natural Resources Canada (http://ftp.geogratis.gc.ca/pub/nrcan_rncan/raster/topographic/)
- The website of OSGeo https://download.osgeo.org/geotiff/samples/

Data Structures

FUGAWI/Memory-Map JPR data structure, variables and values

In the table below you will find the JPR data structure, variables and aloud values, as described by FUGAWI. This table is the base for the conversion tools. By experience (trial and error) I know Memory-Map uses a subset. For example vertex polygon defined by

geographical coordinates isn't supported. I chose to convert as must information as possible.

Variable	Description	Variable type	Structure/aloud value(s)	Remarks
TFW data	First 6 Lines based on TFW (optional if reference points provided).			Will not be used
vr	Jpr version number	Alphanumeric	$$$	Will not be used
sc	Scale	Numeric, entered as the denominator	###	
mc	Longitude at centre of map	Numeric	(-)###.##	Will not be used
pr	Projection	Alphanumeric:	UTM, Mercator, Lambert Conformal Conic, Polyconic, Transverse Mercator, Equirectangular, Cassini, Finnish KKJ, Finnish YKJ	
pp	Central Meridian if Lambert, Polyconic, Cassini and Transverse Mercator.	Numeric	(-)###.##	
p1	Latitude of origin if Lambert, Transverse Mercator and Cassini; mid-latitude if Mercator	Numeric	(-)###.##	
p2	Scale factor if Transverse Mercator	Numeric	###.##	
p3	False northing if Transverse Mercator	Numeric	###.##	
p4	False easting if Transverse Mercator	Numeric	###.##	
p5	Standard Parallel 1 if Lambert	Numeric	(-)###.##	
p6	Standard Parallel 2 if Lambert	Numeric	(-)###.##	
zn	Zone UTM number	Alphanumeric	##$ ## = zone number $ = t for the northern hemis-phere and j for the southern hemisphere.	
dm	Datum	Alphanumeric	Appendix A, table 1: Supported datums by FUGAWI	

st	Latitude datum shift in degrees	Numeric	(-)###.## (Don't use minutes and seconds)	
sn	Longitude datum shift in degrees	Numeric	(-)###.## (Don't use minutes and seconds)	
un	units for grid in lines 5 and 6 of TFW portion of file		meters (default value if un = line missing), feet	Will not be used
cr	Copyright statement	Alphanumeric	$$$	Will not be used
nm	Name of map	Alphanumeric	$$$	
cu	Contour line or altitude units	Alphanumeric	Meters, feet	Will not be used
ci	Contour interval	Numeric	###.##	Will not be used
du	Depth Units	Alphanumeric:	Meters, Feet, fathoms	Will not be used
rp1 . . . . rpN	Reference point (latitude, longitude, x, y)	Numeric, numeric, numeric, numeric	(-)###.##, (-)###.##, ###, ### (Latitude and longitude in degree, don't use minutes and seconds)	
vp1 . . . . vpN	Pixel of boundary vertex polygon	Numeric, numeric	###, ###	
vg1 . . . . vgN	Latitude and longitude of boundary vertex polygon	Numeric, numeric	(-)###.##, (-)###.## (Latitude and longitude in degree, don't use minutes and seconds)	This vertex polygon isn't sup-ported by Memory-Map
su	Data source	Alphanumeric	$$$	Will not be used
ed	Edition number	Numeric	###	Will not be used
et	Edition date	Date	dd/mm/yyyy	Will not be used
dt	Date of last correction to map	Date	dd/mm/yyyy	Will not be used
sr	Scan resolution	Numeric in dots per inch	### (Most common: 72, 127, 150, 254, 300, 508, 600)	
sk	Skew angle in degrees	Numeric	(-)###.## (Don't use minutes and seconds)	Will not be used
it	Image type	Alphanumeric	gif, tif, jpg, png, bmp	
sd	Datum for original scanned paper source	Date	dd/mm/yyyy	Will not be used

Table 1: Structure of JPR-file

OziExplorer MAP-data structure, variables and values.

To get the picture of the MAP data structure, variables and aloud values I analyzed the Ozi Explorer help file and a lot of example MAP-files. The first 40 lines have a strict structure and a strict order. The following lines do have a strict structure but aren't obligate. As the conversion tool is one way from MAP-file to JPR-file, I searched for the corresponding data for all necessary JPR variables an equivalent in the .MAP file content. The result you will find in the table below.

Variable	Description	Remarks/Relation with MAP-file
sc	Scale	Related to line MM1B of MAP-file
pr	Projection	Line 9 of MAP-file
pp	Central Meridian if Lambert, Polyconic, Cassini and Transverse Mercator.	Line 40 of MAP-file
p1	Latitude of origin if Lambert, Transverse Mercator and Cassini; mid- latitude if Mercator	Line 40 of MAP-file
p2	Scale factor if Transverse Mercator	Line 40 of MAP-file
p3	False northing if Transverse Mercator	Line 40 of MAP-file
p4	False easting if Transverse Mercator	Line 40 of MAP-file
p5	Standard Parallel 1 if Lambert	Line 40 of MAP-file
p6	Standard Parallel 2 if Lambert	Line 40 of MAP-file
zn	Zone UTM number	Line 10 to 39 of MAP-file
dm	Datum	Line 5 of MAP-file
st	Latitude datum shift in degrees	Line 5 of MAP-file
sn	Longitude datum shift in degrees	Line 5 of MAP-file
nm	Name of map	Line 2 of MAP-file
rp1 . . . . rpN	Reference point (latitude, longitude, x, y)	Line 10 to 39 of MAP-file in case latitude and longitude are given. Alternative: Lines MMPXY and MMPLL and in MAP-file
vp1 . . . . vpN	Pixel coordinate of boundary vertex polygon	Lines MMPXY in MAP-file
vg1 . . . . vgN	Latitude and longitude of boundary vertex polygon	Lines: MMPLL in MAP-file (Memory-Map doesn't support this type of vertex polygon)
sr	Scan resolution	Related to MM1B MAP-file
it	Image type	Line 3, based on file extension

Table 2: Confrontation JPR-file/OziExplorer MAP-file

Remarks

Resolution and scale

Neither resolution nor scale aren't a variable in the MAP-file. The only usable variable is MM1B. It represents the size of a dot in reality in meter. Based on this information and a given resolution the scale (sr) can be calculated. `sc = (sr/2.54) *100 * MM1B)`. The scale is always an estimate value.

MOBAC data structure, variables and values.

In the context of this project MOBAC is a special case of OziExplorer MAP-file. The MAP-file generated by MOBAC is very basic. From the information on the MOBAC forum I concluded MOBAC uses a Spherical Mercator projection (https://mobac.sourceforge.io/forum/viewtopic.php?t=769). Knowing this, I was able to fill the necessary gaps. In the table below You will find the result.

Variable	Description	Remarks/Relation with MAP-file
sc	Scale	Related to line MM1B of MAP-file
pr	Projection	Value is mercator
pp	Central Meridian	Value is 0
p2	Scale factor	Value is 1

p3	False northing	Value is 0
p4	False easting	Value is 0
dm	Datum	Value is wgs84
st	Latitude datum shift in degrees	Value is 0
sn	Longitude datum shift in degrees	Value is 0
nm	Name of map	Line 2 of MAP-file
rp1 . . . . rpN	Reference point (latitude, longitude, x, y)	Line 10 to 39 of MAP-file in case latitude and longitude are given. Alternative: Lines MMPXY and MMPLL and in MAP-file
vp1 . . . . vpN	Pixel coordinate of boundary vertex polygon	Lines MMPXY in MAP-file
vg1 . . . . vgN	Latitude and longitude of boundary vertex polygon	Lines: MMPLL in MAP-file (Memory-Map doesn't support this type of vertex polygon)
sr	Scan resolution	Related to MM1B MAP-file
It	Image type	Line 3, based on file extension

Table 3: Confrontation JPR-file/MOBAC MAP-file

GeoTIFF/GeoPDF data structure, variables and values.

Contrary to the JPR- and MAP-files documentation the line by line/field by field documentation for GeoTIFF/GeoPDF files is scattered. Most information in I found on the Gdal website. Additionally I collected al lot of different GeoTIFF/GeoPDF files to analyze the output. The result you will find in the table below. I'm sure the table isn't complete, but gives, for the moment, enough information to build a conversion tool. Gdalinfo uses the Well Know Text (WKT) as standard output.

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, Polyconic, Cassini and Transverse Mercator.	Line: PARAMETER "central_meridian" Gdalinfo generated this value for all Mercator projections.
p1	Latitude of origin if Lambert, Transverse Mercator and Cassini; 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	Line: PARAMETER "standard_parallel_1"
p6	Standard Parallel 2 if Lambert	Line: PARAMETER "standard_parallel_2"
zn	Zone UTM number	Line: PROJCS
dm	Datum	Line: DATUM

st	Latitude datum shift in degrees	Value = 0 (correlation not found yet)
sn	Longitude datum shift in degrees	Value = 0 (correlation not found yet)
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 vertex polygon in Pixels	If the map hasn't a collar: line: Size is for pixel coordinates, if it has and there is a NEATLINE given the vertex polygon may be calculated (in one of the next versions).
vg1 . . . . vg4	Boundary vertex polygon in latitude and longitude	If the map hasn't a collar: lines: Upper Left Lower Left, Upper right and Lower Right. (Memory-Map doesn't support this type of vertex polygon, FUGAWI does!)
sr	scan resolution	Lines: TIFFTAG_XRESOLUTION, TIFFTAG_YRESOLUTION and TIFFTAG_RESOLUTIONUNIT
it	image type	Line: Files: use filename extension

Table 4: Confrontation JPR-file/Gdalinfo data

Remarks

Calibration data

The calibration data (vp1 to vp5) are a combination of the size of the image (Size is), latitude and longitude of the corners and the center of map. The line CENTER will also be used to in increase the accuracy of the calibration.

Resolution and scale

The resolution can only be used if TIFFTAG_XRESOLUTION and TIFFTAG_YRESOLUTION have the same value. TIFFTAG_RESOLUTIONUNIT must be taken in account. Value 2 means Dot Per Inch (DPI), value 3 means Dot per Centimeter an Value 1 means No unit.
The scale of the map isn't available as a variable, but can be calculated based on TIFF_XRESOLUTION, TIFF_YRESOLUTION, TIFF_RESOLUTIONUNIT and Pixel Size. This is only possible if these variables are present, TIFF_XRESOLUTION and TIFF_YRESOLUTION have the same value and the x- and y-pixel size have nearly the same value.
The calculation: sc = (RESOLUTION IN DPI/2.54) *100 * Pixel Size). The scale is always an estimate value and based on the presumption the RESOLUTION is correct.

Vertex polygon

If the map hasn't a collar the vertex polygon can defined by Size of the image (Size is). With a collar the only option is to use the NEATLINE (if available and if the line is based on meter based coordinate system. The NEATLINE is defined in map coordinates (northing, easting) instead of pixel coordinates. This means they must be translated. For small scale maps with Mercator like projections it can be done with some strait forward vector calculations (shifting, scaling, rotating) instead of the more complicated geographical transformations. These NEATLINE calculations will be implement in a later version.

Conversion tools

OziExplorer and MOBAC MAP-file (M2J and MOB2J)

Analysing difference between MAP- and JPR-file

From tables 2 and 3 we can learn that, on logical level, only one variable is missing, the resolution (DPI/JPR-variable `sr`) of the map. This aspect must be addressed in the tool (adding a possibility for input). Analysing the aloud values for two variables need attention: the map datum (JPR-variable `dm`) and the map projection (JPR-variable `pr`). In Annex A table 2 and 3 you will find a comparison made between OziExplorer and Fugawi. There is a great discrepancy. In the first versions of M2J the map datum will be reduced to WGS 84, NAD 27 and NAD 83 and the projection to Mercator, Transverse Mercator, UTM and Lambert Conformal Conic, in a late stadium more datums and projections will be added.

Every map projection has its specific variables. For each projection there will be a set of rules.

OziExplorer MAP-File

JPR variable	MAP variable or value	Mercator	Transverse Mercator	Universal Transverse Mercator	Lambert Conformal Conic
`//Comment`					
`nm=`	Line 2 of MAP-file	X	X	X	X
`dm=`	Line 5, field 1	X	X	X	X
`st=`	Line 5, field probably 3, value mostly 0,	X	X	X	X
`sn=`	Line 5, field probably 4, value mostly 0	X	X	X	X
`pr=`	Line 9, field 2, `WGS84/NAD27/NAD83`	X	X	X	X
`pp=`	Line 40, field 3		X		X
`p1=`	Line 40, field 2	X	X		X
`p2=`	Line 40, field 4		X		
`p3=`	Line 40, field 6		X		
`p4=`	Line 40, field 5		X		
`p5=`	Line 40, field 7				X
`p6=`	Line 40, field 8				X
`zn=`	mask ##$ Line 10, field 14 and t for the northern hemisphere and j for the southern hemisphere			X	
`it=`	Line 3, file extension	X	X	X	X
`sr=`	DPI, must be selected	X	X	X	X
`sc=`	To be calculated based on line `MM1B` of MAP-file and sr	X	X	X	X
`rpN where N = 1 to max ∞`	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	X	X	X	X

	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 where N = 1 to max ∞	Mask ###, ### Field 3: Line MMPXY, N, field 3 of MAP-file Field 4: Line MMPXY, N, field 4 of MAP-file	X	X	X	X
vgN where N = 1 to max ∞	Mask ###, ### Field 3: Line MMPLL, N, field 3 of MAP-file Field 4: Line MMPLL, N, field 4 of MAP-file Not supported by Memory-Map	X	X	X	X

Table 5: OziExplorer MAP-file conversion rules

MOBAC MAP-file

The MAP-File MOBAC generates is based on spherical Mercator projection and WGS84 as datum. Conversion rules are shown in the table below.

JPR variable	MAP variable or value
//Comment	
nm=	Line 2 of MAP-file
dm=	wgs84
st=	0
sn=	0
pr=	Mercator
pp=	0
p2=	1
p3=	0
p4=	0
it=	Line 3, file extension
sr=	DPI, must be selected
sc=	To be calculated based on line MM1B of MAP-file and sr
rpN where N = 1 to max ∞	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 where N = 1 to max ∞	Mask ###, ### Field 3: Line MMPXY, N, field 3 of MAP-file Field 4: Line MMPXY, N, field 4 of MAP-file
vgN where N = 1 to max ∞	Mask ###, ### Field 3: Line MMPLL, N, field 3 of MAP-file Field 4: Line MMPLL, N, field 4 of MAP-file Not supported by MEMORY-MAP

Table 6: MOBAC conversion rules

GDalInfo data

GdalInfo generates data about Projection, Datum, Calibration and Coordinate systems in Well Known Text. The conversion rules are based on the analyse shown in the table below.

JPR variable	GDalInfo variable or value

//comment	
nm=	Line: Files: use filename without path and extension
dm=	Line: DATUM, the value must be translated to a corresponding JPR-value
st=	Value is 0
sn=	Value is 0
pr=	Line: PROJECTION, the value must be translated to a corresponding JPR-value
pp=	Value of line: PARAMETER "central meridian"
p1=	Value of line: PARAMETER "latitude of origin"
p2=	Value of line: PARAMETER "scale factor"
p3=	Value of line: PARAMETER "false northing"
p4=	Value of line: PARAMETER "false easting"
p5=	Value of line: PARAMETER "standard_parallel_1"
p6=	Value of line: PARAMETER "standard_parallel_2"
zn=	Extracted from line: "PROJCS"
it=	Value of line: Files: use extension of file and add a comment line in case the value isn't png, jpg, tif, gif or bmp (for example pdf)
sr=	Lines: TIFFTAG_XRESOLUTION, TIFFTAG_YRESOLUTION and TIFFTAG_RESOLUTIONUNIT TIFFTAG_XRESOLUTION must be equal to TIFFTAG_YRESOLUTION and TIFFTAG_RESOLUTIONUNIT must be 2 (Dot per inch) or 3 (Dot per centimetre). In this case the resolution must be multiplied by 2.54.
sc=	Must be calculated based on JPR variable sr and line: Pixel Size. Because the scale (sc) is an estimate, the absolute value of the X and Y pixel may have a small difference. The average of the absolute values is used.
rpN where N = 1 to 5	Mask (-)###.##, (-)###.##, ###.#, ###.# (Longitude, Altitude, Pixel X, Pixel Y) Lines: Upper Left Lower Left, Upper right, Lower Right and Center for latitude and longitude Line: Size is for pixel coordinates JPR field 1: Longitude corner coordinate (third value) JPR field 2: Latitude corner coordinate (fourth value) JPR field 3: Pixel X coordinate JPR field 4: Pixel Y coordinate rp1: upper left corner; pixel coordinate = (0,0) rp2: upper right corner; pixel coordinate = (width – 1,0) rp3: lower right corner; pixel coordinate = (width – 1, height – 1) rp4: upper left corner; pixel coordinate = (0, height - 1) rp5: upper left corner; pixel coordinate = (½ width, ½ height) Degrees must be decimal (no minutes and/or seconds); conversion is needed.
vpN where N = 1 to 4	If map doesn't has a collar Mask ###, ### (Pixel X, Pixel Y) Line: Size is JPR field 1: Pixel X coordinate JPR field 2: Pixel Y coordinate vp1: upper left corner; pixel coordinate = (0,0) vp2: upper right corner; pixel coordinate = (width – 1,0) vp3: lower right corner; pixel coordinate = (width – 1, height – 1) vp4: upper left corner; pixel coordinate = (0, height - 1)
vgN where N = 1 to 4	If map doesn't has a collar Mask (-)###.##, (-)###.## (Longitude, Altitude)

	Lines: `Upper Left, Lower Left, Upper right` and `Lower Right` for latitude and longitude JPR field 1: Longitude corner coordinate (third value) JPR field 2: Latitude corner coordinate (fourth value) `vg1`: upper left corner `vg2`: upper right corner `vg3`: lower right corner `vg4`: upper left corner Degrees must be decimal (no minutes and/or seconds); conversion is needed. Not supported by MEMORY-MAP

Table 7 Gdal conversion rules

GdalInfo tool

Before G4J can be used, the application must "know" where to find the "OSGeo4W/Gdal" environment. This "finding" must be addressed in the application.

Building en testing the tools

In general

In the descriptions below you will find the main structure of the tools. All the details you will find in the code. Comment lines in the code are used for explanation.

The difference between a comma and a dot in the values used by a JPR-file is essential. The tools do have a vale/save function to prevent the wrong use.

Visual Studio 2019 Basic

The tools were built as a Windows Forms App (.NET Framework) in Visual Studio 2019 Basic. The code is depending on features of this template (I know by trial and error).

Building M2J

1. The first step is to find the MAP-file. If a MAP-file is selected the content of the file will be shown. If the file doesn't contain georeferenced information the procedure stops.
2. The second step is extract the JPR-data. Comments and the converted data will be presented to the user.
3. The third step is to build the content for the JPR-file. This content will be presented to the user.
4. The forth step is selecting/changing the `resolution`. De default value is 254 dpi (=100 dot per cm). After every change the `scale` will recalculated.
5. The last step is the saving van JPR-data in a file in the same directory of de MAP-file with the same name but with the extension .jpr.

The conversion tools will be based on the strict structure of the MAP-file. Editing this file before conversion may have unpredictable results. For more detail read the code in appendix B. Below an image of the application.

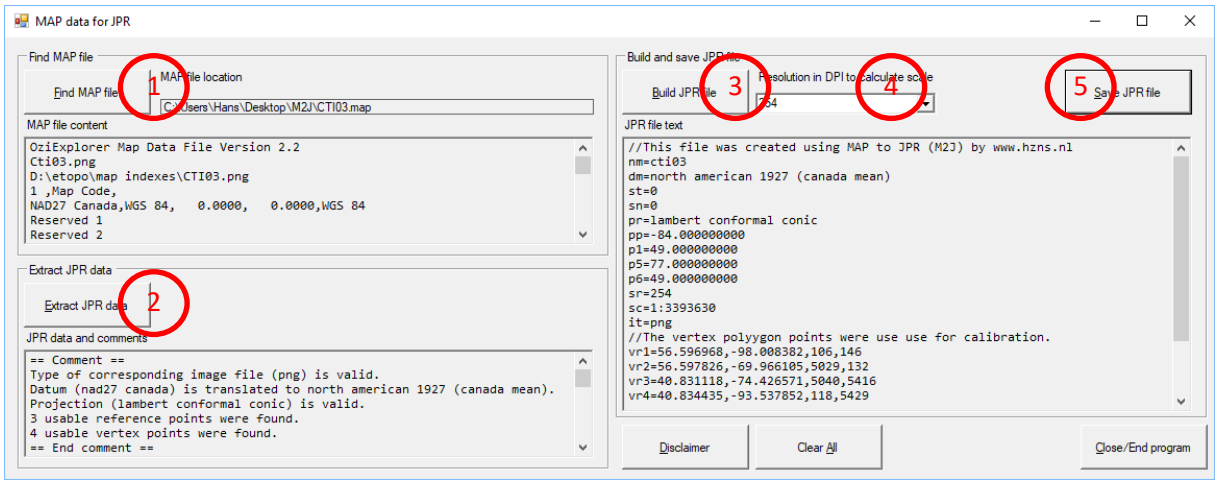


Image 1

Building MOB2J

1. The first step is to find the MAP-file. If a MAP-file is selected the content of the file will be shown. If the file doesn't contain georeferenced information the procedure stops.
2. The second step is convert these data to the JPF-format. The converted data will be presented to the user.
3. The third step is selecting/changing the `resolution`. De default value is 254 dpi (=100 dot per cm). After every change the `scale` will recalculated.
4. The last step is the saving van JPR-data in a file in the same directory of de MAP-file with the same name but with the extension .jpr.

The conversion tools will be based on the strict structure of the MAP-file. Editing this file before conversion may have unpredictable results. For more detail read the code in appendix B. Below an image of the application.

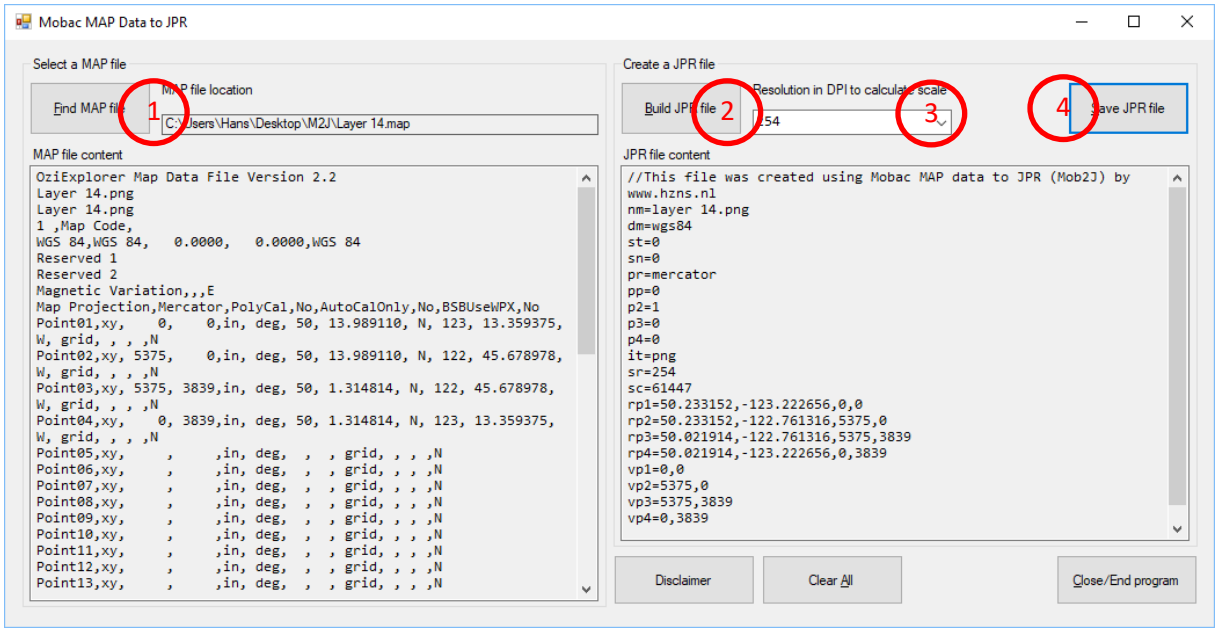


Image 2

Building G2J

1. The application uses the function GdalInfo of the OSGeo4W/Gdal environment. Before the application can be used, it must know where to find the batch file `OSGeo4W.bat`. (`sub(routine) CheckGdalPath`) The application offers a possibility to find (`sub FindChangeOS`). The location will be stored in a file `g2j.ini` in de `documents` directory

of the user. If a batch file isn't selected, the procedure stops. Using a button the user can change g2j.ini.

2. The second step is to find the TIF- or PDF-file of the map. If a TIF- or PDF-file is selected the relevant text content of the map will be shown. If the file doesn't contain georeferenced information the procedure stops.
3. The third step is extract the JPR-data and, if necessary, evaluate and/or change these data (converting to JPF-format). This the moment were the "magic" happens. Relevant information will be presented to the user.
4. The fourth step is the creation of content of the JPR-file and this information is presented to the user. The conversion of projections and datums to for a JPR-file acceptable value is done in separate functions (function `TranslateProjection()` and function `TranslateDatum()`). Although not necessary, it is convenient for future alterations.
5. The Fifth step is the possibility to change the resolution. In some cases the `TIFFTAG_XRESOLUTION` and `TIFFTAG_YRESOLUTION` aren't correct.
6. The last step is the saving van JPR-data in a file in the same directory of de TIF- or PDF-file with the same name but with the extension .jpr.

For more detail read the code in appendix B. Below an image of the application.

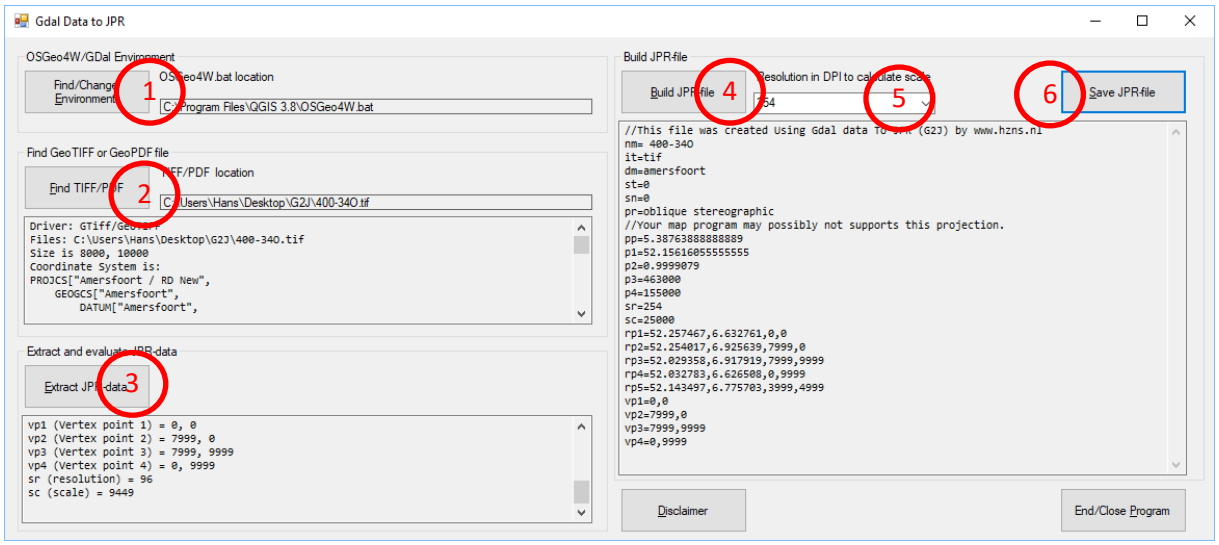


Image 3

Testing

The first phase of the test was done on the fly during the building of the tools. It was iterative, building some functionality an testing if before shifting to the next piece of code. In the second phase of testing was testing the tool as a whole, using real data/maps, sometime altered to generate exceptions and warnings. Although the testing was as carefully as possible, bugs may be emerge. Please let me know (info@hzns.nl).