GSAK 301!

Automating GSAK: Getting Started with Macros

This is a basic guide for those who want to use GSAK automation, or “macros”. All underlined text in this guide is a link to a web site – click on the underlined text to go directly to that website.

Please note: this guide covers automating GSAK. It’s designed to be read from start to finish (although not necessarily in one sitting!), and each section builds on the last *If you skip to topic headings without reading through the earlier material, you may miss key points and end up hopelessly confused.* In addition, this guide assumes that you’re familiar with GSAK features and functions. If you’re not familiar with GSAK, please read **GSAK 101 - Getting Started** and **GSAK 201 - Customizing GSAK.**

Many GSAK features are not addressed here – for information on those features, see the GSAK help file (in GSAK, go to the “Help” menu, then click on “Contents”, or press the F1 key on your computer keyboard). For help on Macros, in GSAK go to the “Macro” menu and click on “Help”. Those with computer programming experience may find this guide too basic – if you’re comfortable with scripting, you can dive right in, using the GSAK help file as your guide.

WHAT ARE “MACROS”?

No, we’re not talking about pasta. “Macro” is technospeak for a series of instructions that **automate a function or feature**. GSAK has automation or “macro” support that allows users to write a series of GSAK instructions to a text file. Users can create a macro file using any text editor, including GSAK's built in macro editor (in GSAK, go to Macro>Edit/Create to open the macro editor).

WHY USE MACROS?

Macros are used to automate frequently-used features (e.g. replace a series of user actions with one) or to enable more powerful features not available in GSAK’s “point and click” user interface.

Let’s say you frequently download a Pocket Query, load the GPX file into GSAK, sort the file by “Last GPX” date, check to see if caches that were not updated are archived, filter by distance from your home, and export the closest 500 caches to a mapping program and your GPS receiver. Rather than going through each of these steps manually each time you receive a Pocket Query, you can use a macro to do all of this with a single mouse click (i.e. automate a series of actions).

Or, let’s say you want to simultaneously sort by multiple GSAK columns (e.g. cache owner, date placed, last found), or use a piece of information in GSAK for which there is no “special tag”. These things can *only* be done through a macro (i.e. enable more powerful features).

ARE MACROS HARD TO USE?

Macros are easy to use - Clyde (GSAK's author) and GSAK users have created many macros and made them freely available to all users in the GSAK Macro Library.

The Macro Library is organized into general categories based on the end result of the macro from the user's perspective. If you can't find what you're looking for in a given category or don't want to browse the categories, use the forum search function to search the entire macro library.

Note: There were significant changes in GSAK macros in version 6.6, and additional macro changes in GSAK version 7. If you're not using the latest version of GSAK, please download the upgrade from the GSAK home page and install it before attempting to run these macros. This guide assumes you're using GSAK Version 7 or above.

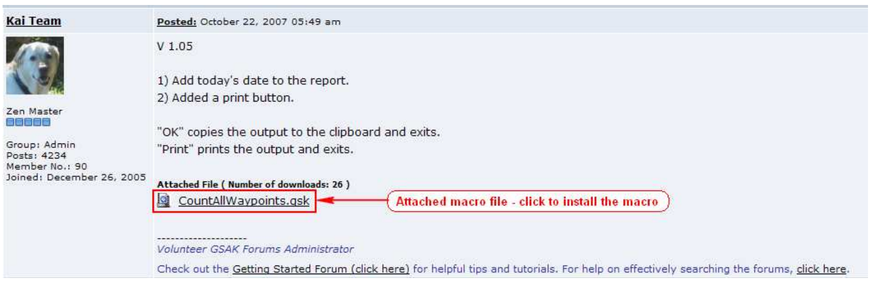
To use a macro from the Macro Library:

1) **Read the Description**: Click on a topic that interests you and read what the macro does. If you're interested in that macro,

2) **Check the Macro for Any Required Customization**: Most macros don't require any adjustment, but some do (e.g. create a saved filter before running the macro for the first time). The description in the Macro Library forum will tell you if you need to make any changes to the macro for your unique setup. If you want to use the macro,

3) **Install the Macro (GSAK Version 7)**: The latest version of the macro is in the last post in the topic, and may contain additional features or fixes compared to earlier versions.

A. Click on the attached file:



Kai Team

Zen Master

Group: Admin
Posts: 4234
Member No.: 90
Joined: December 26, 2005

Posted: October 22, 2007 05:49 am

V 1.05

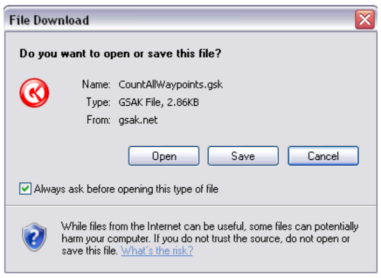
1) Add today's date to the report.
2) Added a print button.

"OK" copies the output to the clipboard and exits.
"Print" prints the output and exits.

Attached File (Number of downloads: 26)
CountAllWaypoints.gsk

Volunteer GSAK Forums Administrator
Check out the Getting Started Forum (click here) for helpful tips and tutorials. For help on effectively searching the forums, click here.

B. The following download screen will appear – click on "Open":



File Download

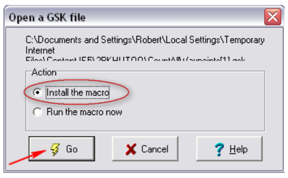
Do you want to open or save this file?

Name: CountAllWaypoints.gsk
Type: GSAK File, 2.86KB
From: gsak.net

✓ Always ask before opening this type of file

While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

C. GSAK will start (if it's not already running) and the following screen will appear. "Install the macro" should be selected. Click on the "Go" button:



C:\Documents and Settings\Robert\Local Settings\Temporary Internet

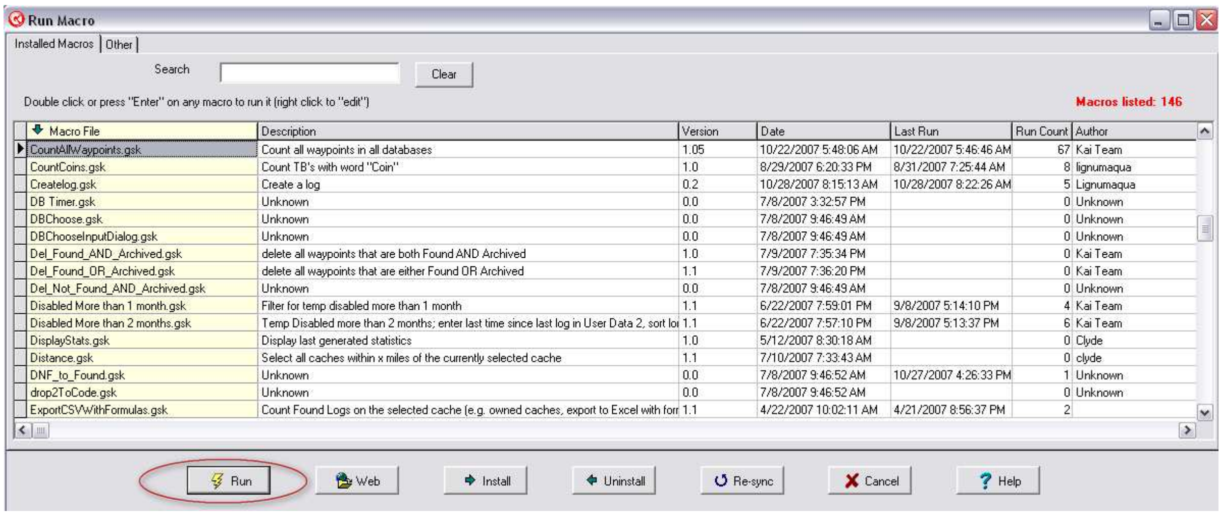
Action

[✓ Install the macro] Run the macro now

D. The macro will be installed.

5) **Customize (if necessary - see Step 2)**: If you're required to change any macro settings (e.g. saved filters, file names or folder paths), open GSAK and go to the Macro>Edit/Create menu. Click on File>Open in the macro editor and choose the macro you just installed. Make any changes required and save the file.

6) **Run the Macro**: To use the macro, go to Macro>Run/Manage in the GSAK menus (or press Ctrl-M on your keyboard). The "Run Macro" dialog will open, showing a list of all installed macros:



Double click or press "Enter" on any macro to run it (right click to "edit")

Macros listed: 146

Macro File	Description	Version	Date	Last Run	Run Count	Author
CountAllWaypoints.gsk	Count all waypoints in all databases	1.05	10/22/2007 5:48:06 AM	10/22/2007 5:46:46 AM	67	Kai Team
CountCoins.gsk	Count TB's with word "Coin"	1.0	8/29/2007 6:20:33 PM	8/31/2007 7:25:44 AM	8	lignumaqua
Createlog.gsk	Create a log	0.2	10/28/2007 8:15:13 AM	10/28/2007 8:22:26 AM	5	Lignumaqua
DB Timer.gsk	Unknown	0.0	7/8/2007 3:32:57 PM		0	Unknown
DBChoose.gsk	Unknown	0.0	7/8/2007 9:46:49 AM		0	Unknown
DBChooseInputDialog.gsk	Unknown	0.0	7/8/2007 9:46:49 AM		0	Unknown
Del_Found_AND_Archived.gsk	delete all waypoints that are both Found AND Archived	1.0	7/9/2007 7:35:34 PM		0	Kai Team
Del_Found_OR_Archived.gsk	delete all waypoints that are either Found OR Archived	1.1	7/9/2007 7:36:20 PM		0	Kai Team
Del_Not_Found_AND_Archived.gsk	Unknown	0.0	7/8/2007 9:46:49 AM		0	Unknown
Disabled More than 1 month.gsk	Filter for temp disabled more than 1 month	1.1	6/22/2007 7:59:01 PM	9/8/2007 5:14:10 PM	4	Kai Team
Disabled More than 2 months.gsk	Temp Disabled more than 2 months; enter last time since last log in User Data 2, sort lo	1.1	6/22/2007 7:57:10 PM	9/8/2007 5:13:37 PM	6	Kai Team
DisplayStats.gsk	Display last generated statistics	1.0	5/12/2007 8:30:18 AM		0	Clyde
Distance.gsk	Select all caches within x miles of the currently selected cache	1.1	7/10/2007 7:33:43 AM		0	clyde
DNF_to_Found.gsk	Unknown	0.0	7/8/2007 9:46:52 AM	10/27/2007 4:26:33 PM	1	Unknown
drop2ToCode.gsk	Unknown	0.0	7/8/2007 9:46:52 AM		0	Unknown
ExportCSVWithFormulas.gsk	Count Found Logs on the selected cache (e.g. owned caches, export to Excel with for	1.1	4/22/2007 10:02:11 AM	4/21/2007 8:56:37 PM	2	

Click on the macro you want to run under "Macro File" and click on the "Run" button at the bottom of the dialog (or just double click the macro file name). The macro will run and perform whatever magic it was designed to perform. For more information on Run/Manage dialog features, please see the GSAK help file (Help>Contents>Using GSAK>Macros (automating GSAK)>Managing and Running).

GSAK users who post macros also maintain them. Please remember that these are fellow users who voluntarily offer their work to others. If you have questions please post in the Macros support forum, with a link or reference to the posted macro.

Note: If this is a macro you'll use often, you may want to add it to your tool bar. Now it will only be one mouse click away!

ARE MACROS HARD TO CREATE?

Whether you'll find creating a macro easy or hard depends on the complexity of the tasks that you want to automate. It also depends on how much attention you pay to details, your patience and your aptitude for logical thinking.

Many macro writers start off automating a fairly easy task (we'll give examples below), or they begin by studying, copying pieces of, or "tweaking" a macro someone else wrote. All of the macros in the GSAK Macro Library may be freely copied and edited. A great way to learn how macros work is to look at how other users have done things in *their* macros.

WRITING YOUR FIRST MACRO

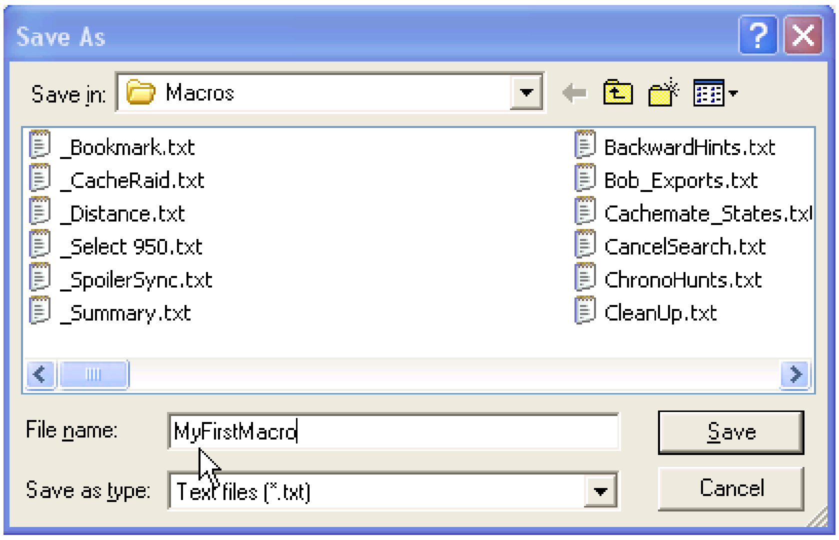
Basic Macros

In its simplest form, a macro can be a single line. Just to get your feet wet, let's write and run a simple macro. This macro will tell GSAK to pause what it's doing and display a dialog window with a message we've created (in this case, the message will be "My first macro is a success!").

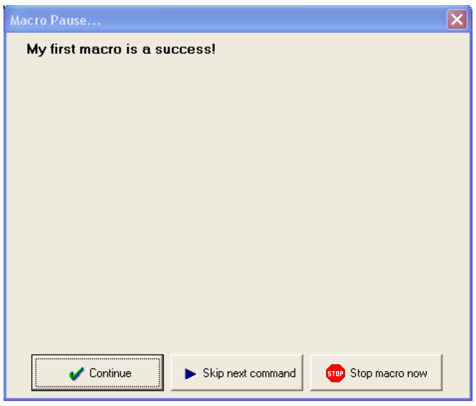
First, in GSAK, go to Macro>Edit/Create.   Now type (or copy and paste) the pink text exactly as shown here (it won't be pink in the macro editor – that's OK):

```
PAUSE Msg="My first macro is a success!"
```

Now go to the File menu in the macro editor and choose "Save and Run".   A dialog box will pop up asking you to name your macro. Type "MyFirstMacro" into the File Name box (in place of "New Macro") and click on the "save" button:



GSAK will save and run the macro, and a dialog window will pop up, showing our message:



My first macro is a success!

Click on the "Continue" button to finish the macro. You're now officially a GSAK macro writer!

Planning a more complex macro – the logical process

The first step in writing a more complex macro is to clearly define what it is you want to do. This means not only the final outcome, but also the sequence of steps in logical order – i.e. "first things first".

If you want to find a geocache (the final outcome), you first need to get in your car and drive a specific route, with various turns along the way (the steps). In planning a cache hunt, you wouldn't start by thinking about where you'll look for the cache once you got there. First, you think about how to get from where you are to the cache and only then do you think about where you'll look for the cache.

Writing a macro involves a similar, step-by-step process. If you miss a turn along the drive you won't end up at the cache, and if you leave a step out of a macro you won't get the desired outcome!

To get started, we're going to plan and write a relatively simple but useful macro. The outcome of this macro is to select the 200 caches closest to a given point for viewing, sending to a GPS receiver, or exporting to a file or other program (e.g. a mapping program or Cachemate). The logical steps to get this result are:

1) Select the database that contains the desired caches,
2) Sort the caches by their distance from your selected location
3) Select the first 200 caches (those closest to your location)
4) Filter for the selected caches.

This macro is similar to performing the following actions from the GSAK menus:

1. Select a database (Database>Select)
2. Sort the database by distance from the current location (click the "Miles" or "Kms" column heading)
3. Clear all user flags (User Flags>Clear All User Flags)
4. Set user flags for next nn (User Flags>Set for Next nn)
5. Set a filter on user flags = set

However, one big difference is that the macro uses the "macro flag" to set and filter on, rather than the user flag. This means that your previously set user flags are not affected by running the macro.

The GSAK Macro Editor

Although you can use any plain text editor to write a macro (e.g. Windows Notepad), GSAK has a built-in macro editor that's convenient to use because it's integrated with GSAK. You can test run your macros from the macro editor (File>Save and Run or CTRL-R) and, if the macro produces an error, you can press a button to return to the macro editor at the line where the error occurred.

To create a new macro in the GSAK macro editor, in GSAK go to Macro>Edit/Create.

Writing a macro – using the macro help files

When writing a macro, the GSAK macro help file is like a foreign language dictionary – it tells you how to translate what you want to do into terms that GSAK understands.

To access the macro help file, in GSAK go to Macro>Help.

Just like any language, there are several categories of terms in the macro language. These are highlighted as links at the top of the Macro Help, and clicking on one of the highlighted links will take you to the list of available terms, their definitions, and their syntax (the options associated with that term).

Don't worry about memorizing all of this information now (there's so much you can do in GSAK macros that it can make your head spin). We'll give you an overview of the general terms here, but you can always refer back to the Macro Help if you forget what something is. We'll also explain how to actually use some of these terms in our example (further below). That said, here are the types of things you'll find in the macro help (your macro language dictionary):

> **Commands & Functions** – tell GSAK to take action! These are the "verbs" of the macro language (computer programmers will tell you there's a difference between commands and functions and they're right, but we won't get into that here). To see all of the commands available in GSAK, click here and then click on "command summary". For a list of functions available in GSAK, click here and then click on "function summary".
>
> **Variables**: are the things you take action on (the "nouns" of the macro language). There are several types of variables:
>
> > **User Created Variables:** are terms that you define to store information, either for use in the macro, or for use in the result. All user created variables start with a dollar sign ($) and can be named almost anything you want (e.g. $MyVariable is a valid user created variable), as long is it starts with $. The only exception to freely naming user created variables is when that name is reserved for a database or system variable:
> >
> > **Database Variables:** are predefined terms that represent the information (data) stored in GSAK. All Database variables begin with "$d_". For example, the GC or other unique code for a cache or waypoint is stored in the database variable $d_Code. For a list of database variables available in GSAK, click here. You cannot use $d_ to name a *user created* variable because that prefix is reserved for database variables.
> >
> > **System Variables:** are predefined terms that represent information that exists in your computer system when GSAK is running, but outside of the database itself (i.e. it's not waypoint or cache information). All system variables start with "$_". For example, the currently selected database is stored in $_CurrentDatabase, and the place on your computer where you installed GSAK is stored in the $_Install system variable. For a list of all available system variables, click here and then click on "system variables". Again, you cannot use the $_ prefix for a *user created* variable because it's reserved for system variables.

Expressions: Expressions allow to you perform mathematical or logical actions or tests on variables and data. For example, expressions allow you to add, subtract, multiply, compare (equal to, not equal to, greater than, less than) or logically combine (AND, OR, NOT) different terms. To see the expressions available in GSAK, click here and then scroll down to "Variable Operands", "Relational Operators" and "Logical Operators" (don't worry, you don't have to be a mathematician, despite the esoteric names!).

Enough already! Let's write a macro:

OK, let's start with the example we mentioned in "planning a macro", above. Remember that we want to show the nearest 200 caches to our location, and the steps are:

1) Select a database,
2) Sort the database by distance from the current location,
3) Select the first 200 caches (those closest to the location),
4) Filter for the selected caches.

If you haven't already done so, open the GSAK macro editor (in GSAK, Macro>Edit/Create) and let's get started. Note: The actual "macro code" (instructions) are shown in this guide in pink to make them easy to distinguish from the accompanying explanations. You can enter each line as we go along, or you can copy and paste the final macro (shown at the end of this discussion) into the Macro Editor.

Step One: It is good practice to include comments in your macros. A comment is any line that starts with the # character, and any line that starts with # is ignored by GSAK. So, the first line of our macro is a comment that reminds us what this macro does:

```
# Macro to select a number of caches that are closest to a location
```

You can use comments anywhere in the macro that you like, and we highly recommend it! Macros that are "well documented" (have good comments) help you to "debug" (fix errors in) your macro, and they help you and others understand out what the macro is doing (it's amazing how quickly you can forget what you did, or why you did it, when you look back at a macro you wrote).

Step Two: Now that we know what this macro is intended to do, we need to select a database that has the caches we're interested in. For the purpose of this lesson, we're going to select the "Default" database because GSAK always has a "Default" database. The command to select a database is:

```
DATABASE Name="Default" Action=Select
```

The first part of that line (DATABASE) is the command, and tells GSAK that we want to do something with a database. The second part (Name=) tells GSAK *which* database you want to act on. The third part (Action=) tells GSAK *what* you want to do with that database. In this case, we're telling GSAK to select, or switch to, the database named "Default".

If you go to the macro help file (Macro>Help, or online click here), click on the "Command Summary" link, and then click on "DATABASE", you'll see the following "syntax" for the DATABASE command:

DATABASE <Name="Database"> [<Action=select|create|delete>]

In the Macro Help files, the command is given first, usually in capital letters. "Tokens" supplement the command, and are surrounded by <> (do not include these <> when creating a macro) and items in square brackets ([..]) are optional. Valid options for tokens are separated by a | (a vertical bar). All commands, tokens, and parameters are *not* case sensitive (i.e. capitalization is unimportant).

In this case, the command is DATABASE and must be included. NAME is a required token and "database" (the name of your actual database) must be included in double quotes (which is why it's quoted in the help file). Action (in the square brackets) is optional – you can include it or not. If Action is not included, GSAK will assume "select" – this assumed default for an optional token is always underlined in the help file. Because the default action for the DATABASE command is "Select", we could have left the Action= token off and gotten the same result.

Step Three: Now that we've told GSAK which database we want the macro to work with, we have to sort the caches in that database by distance from the current location, from closest to farthest. To do that, we enter:

```
SORT By="distance" Sequence=A
```

SORT is the command, By= is the GSAK column we want to sort on (see the SORT command in the help file for a list of column names) and Sequence= tells GSAK whether we want to sort from lowest to highest (i.e. Ascending = A) or highest to lowest (i.e. Descending = D). Sequence= is optional, and the default is ascending (A), so we could have left that token off and gotten the same result.

Step Four: OK, we're in the right database and we've sorted by distance. Now we're going to ask the user how many caches they want to select. The next line in our macro is:

```
INPUT Msg="How many waypoints do you want to select" Default="200"
```

The INPUT command allows the user to "input" (type in) information. It's a very handy command when you want to allow for different answers to a question. Msg= is the message token - whatever you put after Msg= will appear on the screen when the macro is run – i.e. it tells the user what they're being asked (just like the PAUSE command in the first macro you wrote, above, except that INPUT allows the user to respond). Default= is optional. Default= tells GSAK what value to use if the user just presses the enter key without typing an answer. In this case, we're telling GSAK to use 200 if the user doesn't enter a number. Another option for the input command is "BROWSE", which allows the user to browse (use point and click) for either a file or a folder, but we're just asking for a number here, so we won't use that option.

Step Five: The INPUT command collects information typed from the keyboard and creates a default variable called $Result (you can tell the input command to assign the answer to some other variable, using the VarName option, but we don't need to do that here).

Because it's accepting information typed from the keyboard, the INPUT command assumes that the result is what is called a "string" variable. A string variable is any combination of letters, numbers and symbols and is treated as *text* by GSAK. A computer can't count or perform other math on text, so the next step in our macro is to tell GSAK to convert that text "200" to the *number* 200, and to assign it to a user created variable we've called $Number (this could be called $Cucumber if you like, but it makes sense to name variables to represent what they are). The next line in our macro is:

```
$Number = Val($Result)
```

There are two parts to this line. The first part, $Number =, tells GSAK to create a variable called $Number and store whatever comes after the equal sign in that variable. The second half of the line, Val($Result) is using the "Value" function to convert a string (text) into a number (value). In this case we're saying "covert the variable $Result (which came from the INPUT command) into a number.

Step Six: Since we want to count the caches that are closest to our location, we now need to tell GSAK to move the "pointer" (the thing that tells GSAK which cache it's on in your database) to the top of the sorted list of caches (the cache nearest our location). The next line is:

```
GOTO Position=Top
```

The GOTO command tells GSAK to move the "pointer" around in a database. The options are: "top" (the first cache in the current list), "bottom" (the last cache in the current list), "next" (the very next cache below where the pointer is now) and "nn" (some number of caches below the current cache – e.g. "10" would move the pointer down 10 caches from the current spot).

Step Seven: As mentioned above, we're going to use the "Macro Flag", which functions just like user flags, but only within a macro. Using the Macro Flag leaves any previously set user flags untouched. Before we set the macro flags, we have to make sure that there aren't any macro flags already set, so our next line is:

```
MacroFlag Type=Clear Range=All
```

The command is MacroFlag, Type= is the token that tells GSAK what we want to do with the macro flag (in this case, clear it), and Range= tells GSAK which macro flags we want to clear (in this case, all of them).

Step Eight: Now that we know that the macro flags are all unset, we want to set them for the number of caches that the user requested (via the INPUT command). Our next line is:

```
MacroFlag Type=Set Range=$number
```

This is the same command as in Step Seven, but now we're telling GSAK to Set the macro flags for the range that's equal to the variable $number (remember that $number is the numeric value of the answer the user gave in response to the INPUT command!).

Step Nine: Finally, we want to set a filter for those caches that have been marked with the macro flag:

```
MFILTER If=$d_MacroFlag
```

MFILTER is the command that tells GSAK to set a filter from within a macro. It's similar to going to Search>Filter in the GSAK menus, except that you can filter on *any* database variable (see the Macro>Help for a list of all *database variables* you can use). In this case, we're telling GSAK to filter on the database variable $d_MacroFlag, which is the place where GSAK stores the information about whether or not a cache is flagged.

That's it! The entire macro looks like this (you can copy and paste this into the macro editor):

```
# Macro to select the first nn caches in a filter

DATABASE Name="Default" Action=Select
SORT By="distance" Sequence=A
INPUT Msg="First nn waypoints to select" Default="200"
$Number = Val($Result)
Goto Position=Top
MacroFlag Type=Clear Range=All
MacroFlag Type=Set Range=$number
MFILTER If=$d_MacroFlag
```

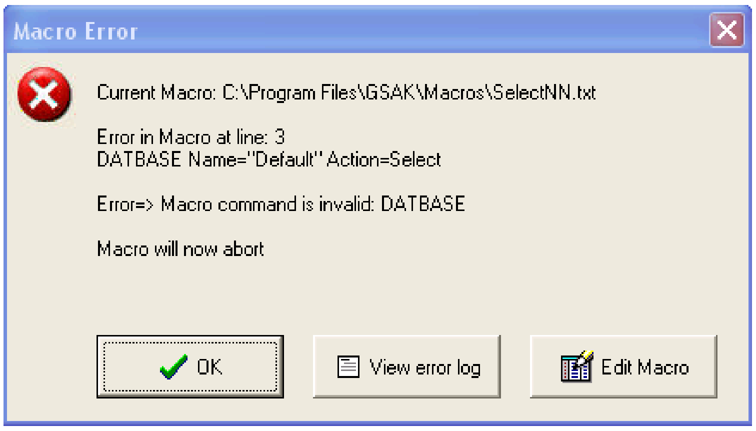
Once you have all of that in the macro editor, go to File>Save and Run. Give the macro a name (how about "SelectNN", without the quotes) and click the "Save" button. GSAK will automatically run the macro and when it finishes, your database will be sorted by the caches closest to your selected location, and filtered for the first 200, ready to export or send to your GPS receiver!

Testing and "Debugging a Macro": No matter how much experience you have writing macros, the odds are that there will be "bugs" in the macro when you first write it (usually these are typographical errors, or they could be errors in the use of a command, function or variable). GSAK reviews the macro code before it starts to run it, and is smart enough to catch these kinds of errors and alert you to the problem. This is called "debugging" the macro, and is a necessary step in the macro creation process.

As an example, let's say that we made a typographical error when entering our macro and entered the first line (after the comment) like this (the second A is missing from the DATABASE command):

```
DATBASE Name="Default" Action=Select
```

In testing our Macro, GSAK will discover this error and pop up an error message that looks like this:



```
Current Macro: C:\Program Files\GSAK\Macros\SelectNN.txt

Error in Macro at line: 3
DATBASE Name="Default" Action=Select

Error=> Macro command is invalid: DATBASE

Macro will now abort
```

GSAK is telling us that our current macro ("SelectNN.txt") has an error in line 3. It shows us the line with the error ("DATBASE Name=…"). It tells us what the error is (DATBASE is not a valid command, because the valid command is DAT**A**BASE). And it stops the macro from running ("Macro will now abort!") so that our invalid command doesn't cause any problems.

The beauty of using the built in macro editor is that, if you click on the "Edit Macro" button, GSAK will take you to the macro editor and place your cursor on the line with the error, so that you can easily make the correction, save and rerun the macro for a second test. This may not seem like a big deal with an error in the third line, but if the error is in line 73, this is indeed a treat!

The DEBUG command: GSAK also has a built in feature to help trace errors in macros – the DEBUG command. The Help file syntax looks like this:

DEBUG <STATUS=on|off> [<Height=nnn>] [<Width=nnn>] [<Left=nnn>] [<Top=nnn>]

The DEBUG command allows you to step through a macro, one command at a time. Each command in the macro will display a dialog showing the command and the value of all variables, just before the command is run. From this dialog you have the option to accept the command, skip the command, or stop the macro. You can turn DEBUG on and off anywhere in the macro by using the corresponding STATUS token ("on" or "off").

Height, Width, Left and Top are optional and allow you to change the size and position of the debug dialog. All measurements (nnn) are in screen pixels.

It's a good idea, especially when you're first learning macros, to include DEBUG Status=on at the beginning of any new block of code you write. Once you have the macro working as expected, you should delete the DEBUG command (or change the status to "off") so that your macro doesn't stop at every step!

Using a test database: The macro we used above is known as "non-destructive" because it doesn't make any permanent changes to your database. It's safe to test a non-destructive macro on "real" data because even if you mess it up, there's no harm done (in this case, just cancel the filter and everything is back the way you started).

However, the macro language is very powerful and can be used to do things that are "destructive" – replacing database variables (the cache data) or deleting caches or even entire databases! We therefore recommend that you create a new database in GSAK called "Test" and copy (don't move) some waypoints into the test database, then use this test database whenever you're debugging a macro. If your macro accidentally deletes all of the caches in your test database, there's no harm done, because that's their sole purpose in life. You can just copy some waypoints back into the test database and start again (imagine your horror if you accidentally delete all the waypoints in a "real" database)!

OTHER USEFUL MACRO CONCEPTS AND TIPS

Add a Macro to a button: In the Customize Speedbar window (see **GSAK 201 - Customizing GSAK** for more detailed information on customizing the GSAK SpeedBar), the next to the last category is "Macros". If you click on that category, you'll see a series of buttons labeled "All Macros" and Macro1, Macro2, etc. If you download or create a macro that you'll use often, remember that you add that macro to a button, then drag it to the Speedbar, and your customized instructions to GSAK are just one click away!

If you assign your macros to buttons and then add the "All Macros" button to the SpeedBar, the All Macros button acts like a drop down menu, allowing you to select any of the assigned macros from a single button! The All Macros button also shows more descriptive text for each macro than you can fit on a macro button.

Interacting with the user: We used two ways of interacting with the user (the PAUSE and INPUT commands) in our example macros, but there are several others:

> **SHOWSTOP** & **SHOWSTATUS**: The SHOWSTOP command shows a small stop dialog when a macro is running. You would usually place this as the first command in a macro with many commands. This allows you to terminate the macro by clicking on the stop button. If you don't use the DEBUG command, we recommend that you include the SHOWSTOP command at the beginning of any new macro you're writing, since it will give you a way out if you write your way into an infinite loop (where the computer keeps processing a command that has no end, and you have to terminate the program or restart your computer to get out of it!). Once the macro is debugged (functioning the way you want it to), you can delete the SHOWSTOP command.
>
> SHOWSTATUS is similar, but it shows you GSAK's progress in processing a large amount of data (i.e. doing something with a large database). See the Macro Help file for more information on using the SHOWSTATUS command.
>
> **CHOOSE**: CHOOSE is similar to INPUT except that it allows you to offer the user pre-defined choices. It's useful to avoid typographical errors and also to limit the user's choices to a set of options (i.e. the user can't type in something that doesn't exist or make sense).

SHOWFORM: The SHOWFORM function allows macro writers to create forms very much like those in GSAK. You can add text labels (instructions, prompts, questions, etc), various controls (buttons, check boxes, check list box, combo box, date box, edit box, file box, folder box, radio buttons, memo box), and images. You can change the colors of your form and control the layout of all elements. For more information on ShowForm, go to Macros>Help>Commands and Functions and scroll down to and click on "ShowForm".

Using conditional statements, loops and subroutines: There are several commands in GSAK that allow you to test something before proceeding, or to repeat an action or set of actions without retyping the same code over and over again:

IF/ELSE/EndIF: command allows you to create a conditional branch in your macro. IF something is true, do this. ELSE (if something is not true), do that.

WHILE/ENDWHILE: command allows you to continue to perform a set of actions *as long as* a condition is true. A common use for WHILE is to "walk" through a set of caches one-by-one, testing each cache (e.g. if it's been found), until you reach the end of the list. The syntax for that is (the system variable $_EoL tells GSAK when it's at the "End of List"):

```
GOTO Position=Top
WHILE not($_EoL)
    IF [condition]
      [do something]
    ELSE
      [do something else]
    EndIF
    GOTO Position=Next
EndWhile
```

GOSUB: The BeginSub, EndSub and GOSUB commands allow you to create a set of macro code (a subroutine) that you can use more than once in the macro. It also allows you to segregate a complex set of code from the rest of your macro, to make it easier to follow and debug.

Please see the Macro Help for more information on using the IF, WHILE and GOSUB commands.

That's enough for a "Getting Started" guide (some of you might say it's too much!). Macros can seem daunting at first, but like so many things in life, they make sense once you start using them.

Remember: start by automating a relatively easy task or by studying, copying pieces of, or "tweaking" a macro someone else wrote. All of the macros in the GSAK Macro Library may be freely copied and edited, and they are a great way to learn how macros work!