**Standard Operation Procedure for Generation of SAP Data Snapshots & Related Search Algorithm**

**10 March, 2017**

**TS-TLP, HAESL**

Table of Contents

[1. Introduction 2](#_Toc476580348)

[2. Working Principle 3](#_Toc476580349)

[i. Extracting data from SAP 3](#_Toc476580350)

[ii. Searching through the dump data 4](#_Toc476580351)

[3. How to Use 6](#_Toc476580352)

[i. Importing the codes 6](#_Toc476580353)

[ii. Generating “Snapshots” 9](#_Toc476580354)

[iii. The Search Function 11](#_Toc476580355)

# Introduction

This Standard Operation Procedure (SOP) describes the inner workings and proper usage of generating SAP data backups and how to search the data collection generated using programs written using SAP GUI Scripting API running from MS Office Excel as host.

As HAESL has been using SAP as its primary Enterprise Resource Planning (ERP) system for years, enormous chunk of data is stored within the SAP system. By the nature of the work in Technical Services (TS) Department – especially for Task List Planning (TLP) Team, daily operations involve searching and manipulating the data which are related to Plant Maintenance Orders (PMOs). Searching the data needed thus comprises a large portion of our workings to deliver fast and accurate products.

Given, there are several ways to search through such data. This includes the SAP transaction “IA17” and “ZTEXT\_TASKLISTS” and etc. However, actual usage of such transactions has shown that transactions in SAP are failing to keep up with the ever growing data set and results in slow response time, measuring in hours, and often meets with crashes and timeouts. Therefore, new methods must be devised in order for the team to handle the workload effectively and without hindrance.

Fortunately, the SAP GUI Client that HAESL is currently using offers an Application Program Interface (API) for manipulating visual elements of SAP GUI. This scripting language, SAP GUI Scripting, presents an opportunity to automate a lot of repetitive works and works that require occasional response from users. This in turn frees us from manual workload on many operations such as generating a search result. Capitalizing the said opportunity, programs written in VBA are made to automatize various routines required in daily operations. Among them, programs for generating a searchable data collection from SAP and a related searching algorithm designed especially for searching the said generated data are produced.

This new method of generating/searching SAP data not only enables a much faster option to search through the data, but it also has several advantages against the traditional method:

* Provides an organized search result table showing details of the search hits, i.e. plan number, operation number, workcentre assigned, packages included & etc…
* Provides regularly generated data collection serving as backups, enabling backtracking and prevention of irreversible data loss from editing.
* Provide a set of complete and searchable data in case of network failure or SAP downtime.

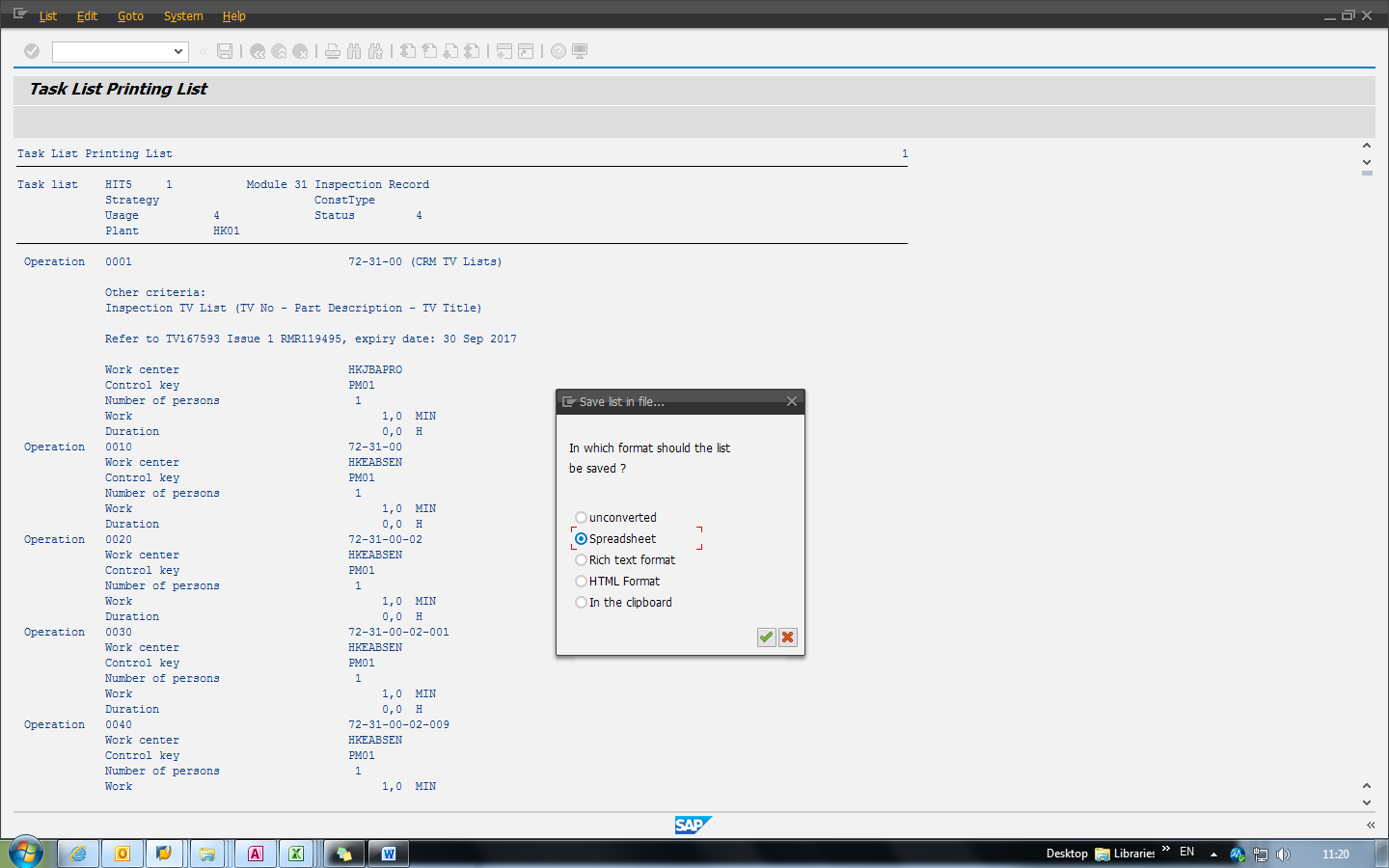
# Working Principle

The aforementioned programs can be divided into two parts in terms of functionality. A program first generates (or “dump”) the data collection utilising SAP GUI Scripting and uses it to control SAP GUI to generate Excel Workbook (.xlsx) files into local storage (“Local” refers to outside SAP system, it is possible to dump the files to a shared, online directory). The data files will be grouped together under a folder.

Another program thus makes use of Excel’s innate search function to produce a search result table from the dumped data files. The search result is then saved as a Workbook at one directory higher than the data collection.

## Extracting data from SAP

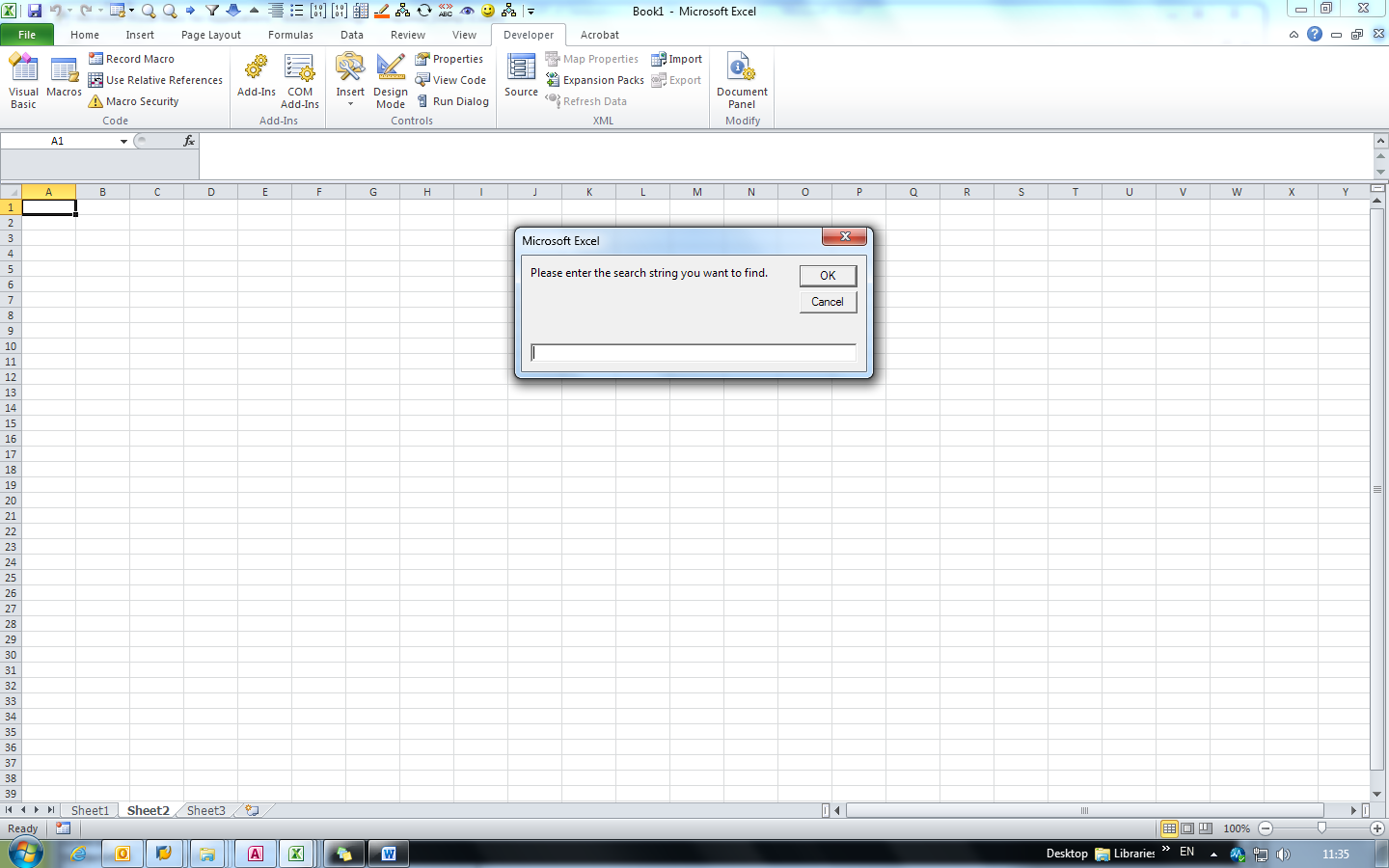
The program uses SAP GUI Scripting to control SAP GUI, generating Excel Workbooks from transaction “IA17” and SAP’s default function to save the data in the format of .xls workbooks, organized under the same folder. Then the program converts the format to .xlsx, reducing the file size and greatly improving the search time needed.



##### Fig.1 & 2. Dumping files through SAP transaction “IA07” then convert & organize.

## Searching through the dump data

The data files generated will be in the format similar to that shown in “IA17”, which makes little sense for untrained eyes and difficult to navigate and extract anything meaningful given the file often contains hundreds of thousands of lines. A program is then written for extracting search hits using user-defined keywords then use them to search related lines in Excel. The result is then saved as .xlsx file.



##### Fig. 3 & 4. User-defined keyword input prompt & search table generated.

# How to Use

To use the programs, the user must have a macro-enabled MS Office Excel to act as host to the codes. Then for the data dumping part, make sure SAP GUI is available.

There are two modules containing the codes necessary to perform the functions:

* Snapshot\_Dump\_Standalone.bas
* Snapshot\_Search\_Standalone.bas

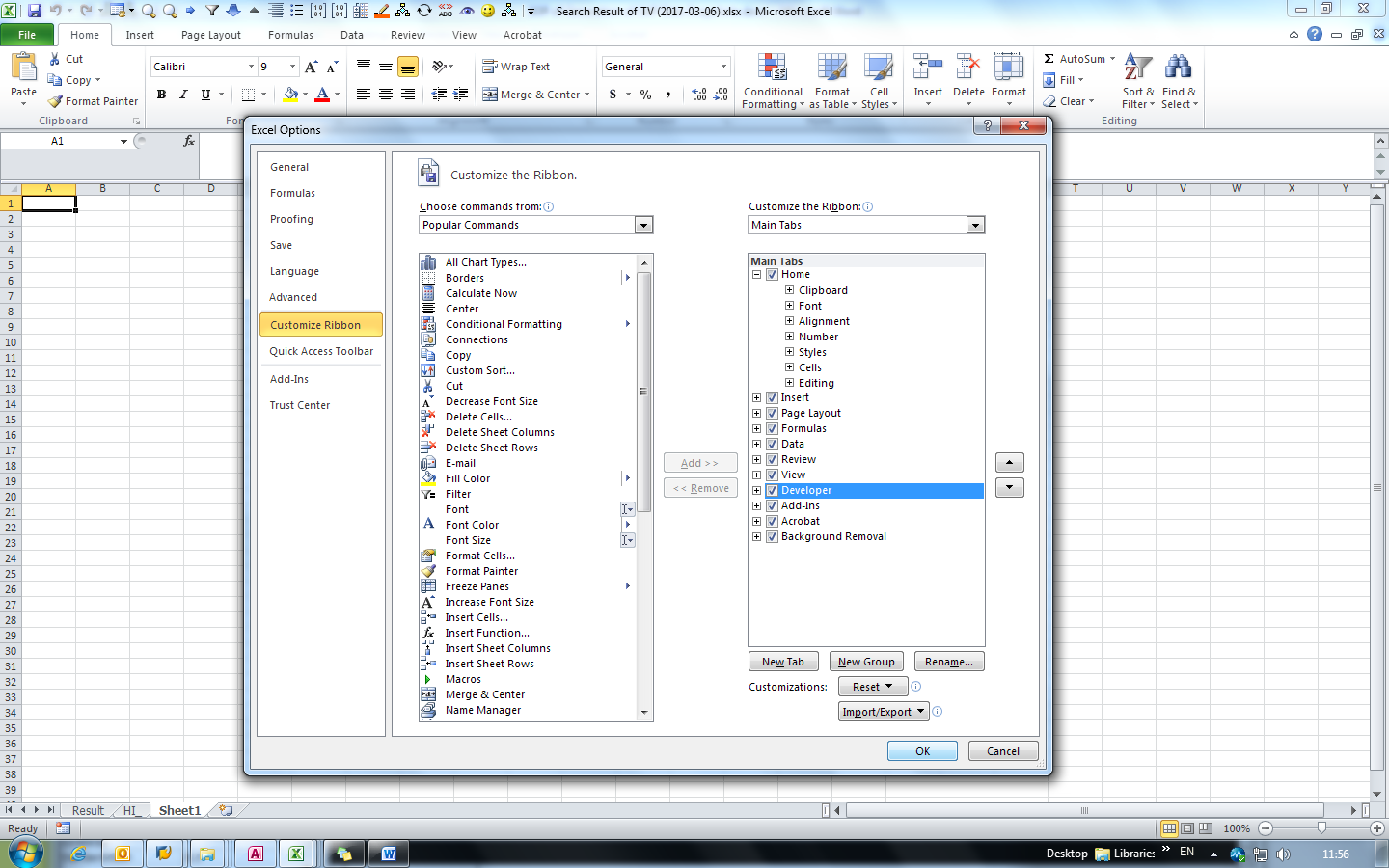
They are stored in [V:\Technical Services\5. Task List Planning\5.2 Team Projects\Snapshots](file:///V:\Technical%20Services\5.%20Task%20List%20Planning\5.2%20Team%20Projects\Snapshots). Both are stand-alone, meaning that each of them can be executed on its own.

## Importing the codes

The codes will need to be imported before being able to run by Excel. This requires importing Visual Basic Exported Module (.bas) files to Excel’s library. The library can be opened in “Developer” Tab in the Excel ribbons which is hidden by default. To make the tab visible, right-click the ribbon and select “Customize the ribbon…”.

##### Fig. 5. Selecting “Customize the ribbon…”

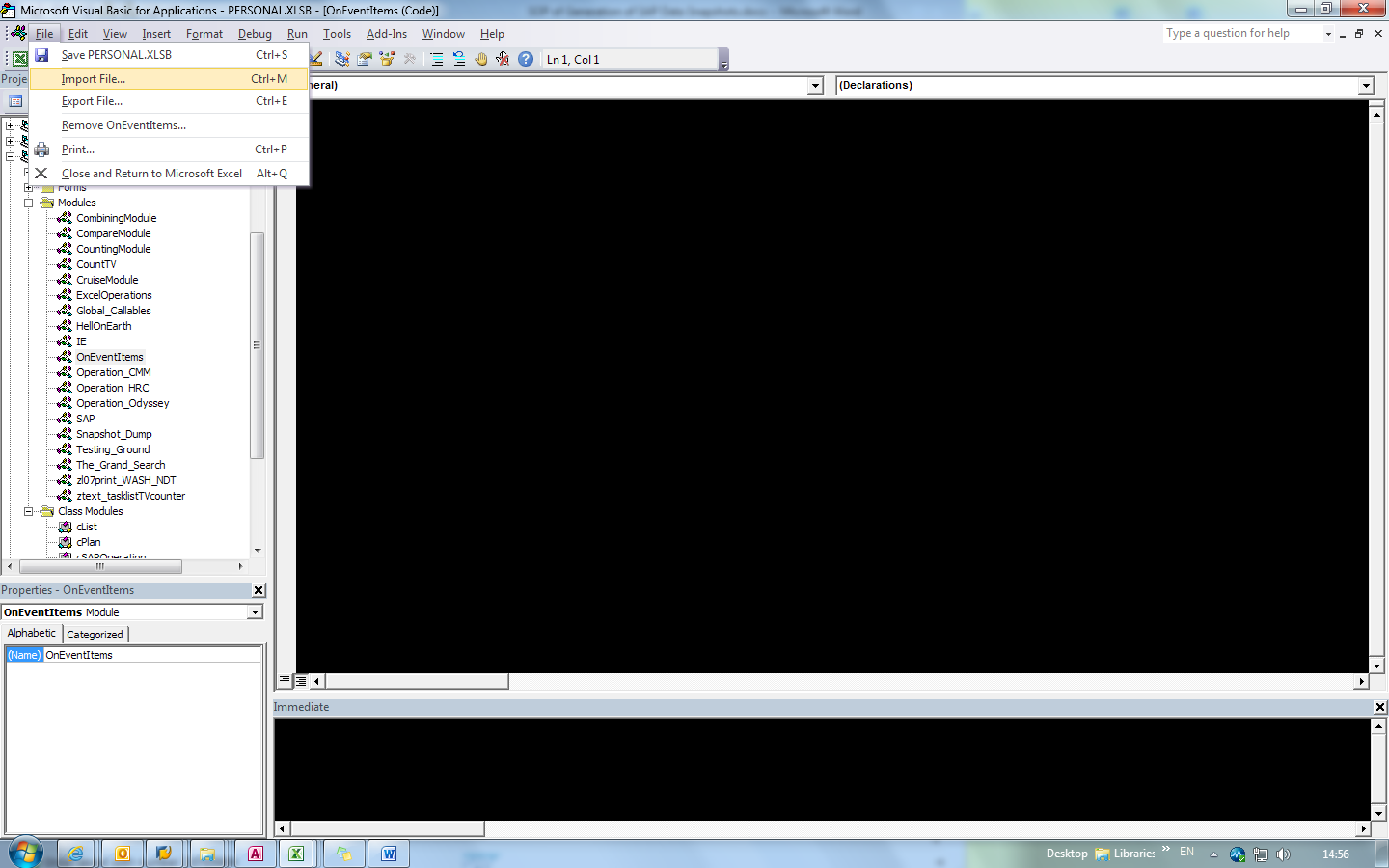
A pop-up window will show up containing configurations for the ribbon. Make sure the “Developer” is ticked and after clicking “OK” the tab “Developer” will show up in the ribbon bar.

Fig. 6. Make sure “Developer” is ticked.

From here, an Integrated Development Environment (IDE) for Visual Basic for Application (VBA) will be accessible. Importing files will be done by using the said IDE. To open, enter the “Developer” tab and click “Visual Basic” at the left most of the bar. It will show a separate window, showing VBA modules.

|  |
| --- |
| Optional: As Excel imports VBA modules to the active workbook, it will store the codes of the imported module at the currently opened workbook only. This means if you want to use the code on different workbooks, you must store it in the personal marco workbook called PERSONAL.XLSB which Excel will open each time whenever it starts up. To import the codes into the said workbook, you can first open it manually by the following address:  C:\Users\<USER>\AppData\Roaming\Microsoft\Excel\XLSTART\PERSONAL.XLSB (Replace <USER> with appropriate user name.)  Then left-clicking the “VBA Project (PERSONAL.XLSB)” at the left hand side panel to make it the active workbook. |

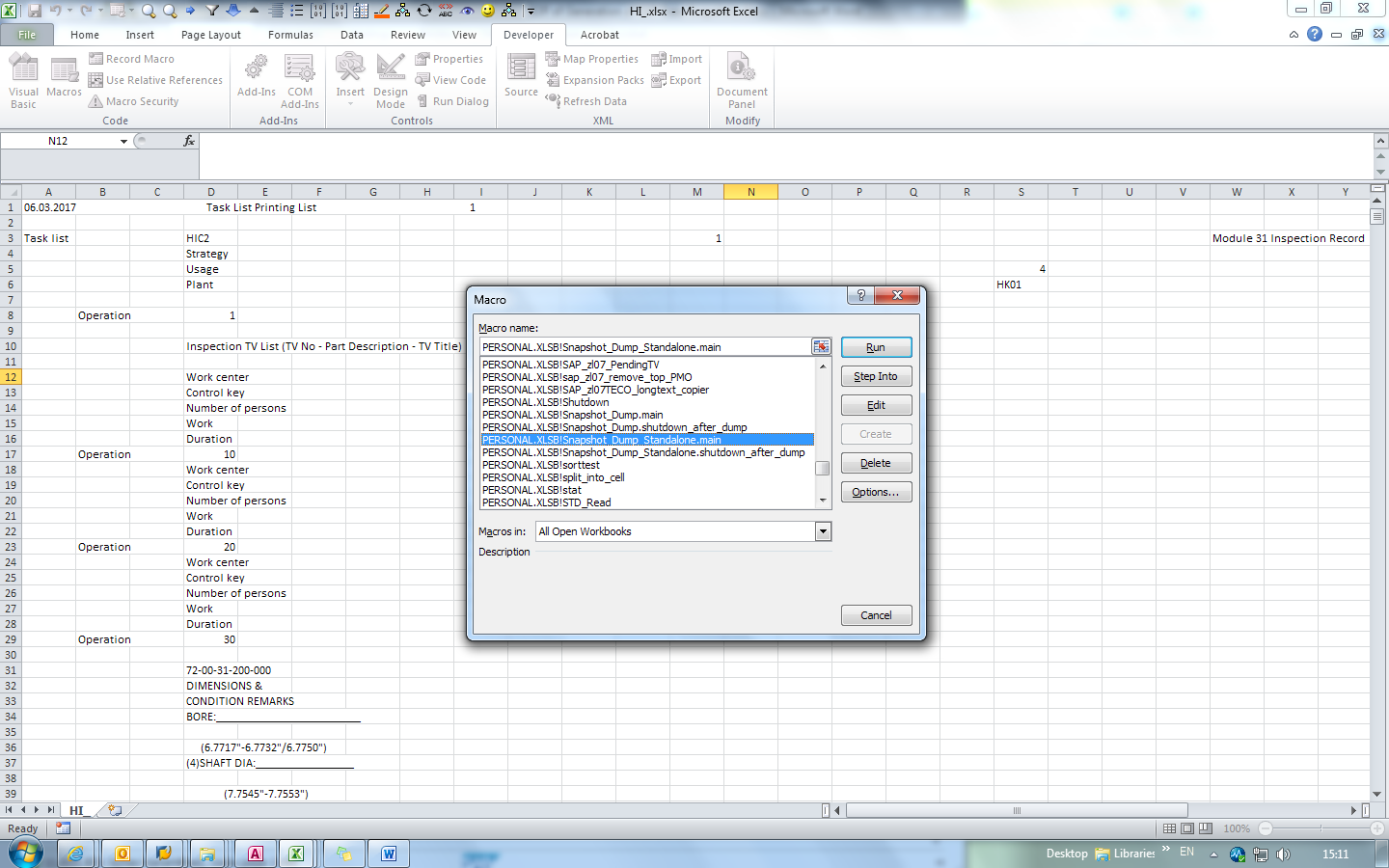
Import the .bas files via “File -> Import File…”.

Fig. 7. Import File under File tab.

Once imported, the modules should be visible at the left hand panel. This indicates the codes have been properly imported and ready to use.

## Generating “Snapshots”

To generate a “snapshot”, simply run the “main()” function inside “Snapshot\_Dump\_Standalone” module. This can be done directly via VBA’s IDE window pressing F5, or indirectly by pressing a “Call Marco” shortcut in Excel (Alt Key + F8) and select “Snapshot\_Dump\_Standalone.main” and click “Run”.

Fig.8. Alt Key + F8, select the marco and click “Run”.

Immediately after clicking, a window will prompt the user to specify a folder for the program to automatically dump the files. Any folder of choice is acceptable as long as the user has Read & Write privileges. After confirming the action, the program should automatically start controlling SAP GUI and generate data files inside the folder, which will later be converted into .xlsx files.

Once completed, the code will open a new Explorer window to show the folder with the readied files before terminating itself.

By default, this will dump all HI & H0 plans in use. This includes:

* All inspection plans (HI)
* All T800 repair plans (H00)
* All T700 repair plans (H01)
* All 524GHT repair plans (H03)
* All 535E4 repair plans (H04)
* All T900 & Misc. repair plans (H07)
* All T500 repair plans (H08)
* All XWB repair plans (HX0)

A collection of snapshots generated in the past are stored online. Please refer to the following link:

[V:\Technical Services\5. Task List Planning\5.2 Team Projects\Snapshots\SNAPSHOTS](file:///V:\Technical%20Services\5.%20Task%20List%20Planning\5.2%20Team%20Projects\Snapshots\SNAPSHOTS)

## The Search Function

The search function is stored within the module “Snapshot\_Search\_Standalone”. To initiate a search, simply run “main()” within the module (“Snapshot\_Search\_Standalone.main”). This can be done directly via VBA’s IDE window pressing F5, or indirectly by pressing a “Call Marco” shortcut in Excel (Alt Key + F8) and select “Snapshot\_Search\_Standalone.main” and click “Run”.

After running the program, it will show a dialogue for user to specify the folder to be used. Please choose the “xlsx” folder by double clicking the folder (Once entered the folder will be shown as empty, this is normal as the program has been instructed to only show folders, not files) or it will return “no xlsx files is present” and terminating itself due to no .xlsx file to execute. Press “OK” to continue.

The program then will ask for keyword input. There are two types of input: Entering the keywords manually, or by selecting the cells containing keywords.

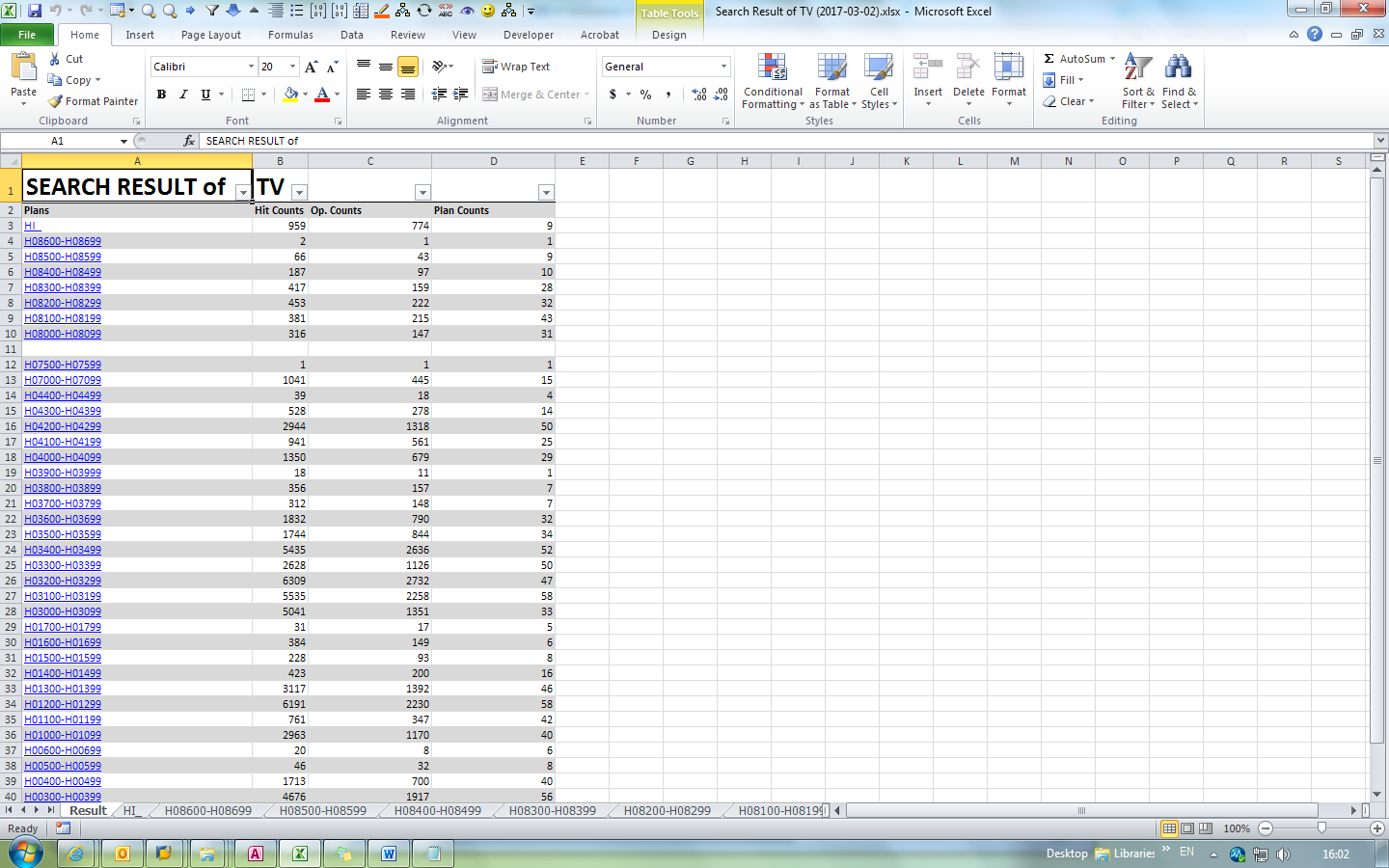
After entering keywords, it will prompt whether to combine the search results if more than one keyword is present. If the user chooses to combine, the program will combine all search results of different keywords ad will only output one single search result file. If the user chooses not to combine, the program will separate each search individually, producing number of files exactly equal to the number of keywords entered (e.g. 23 keywords entered will output 23 files).

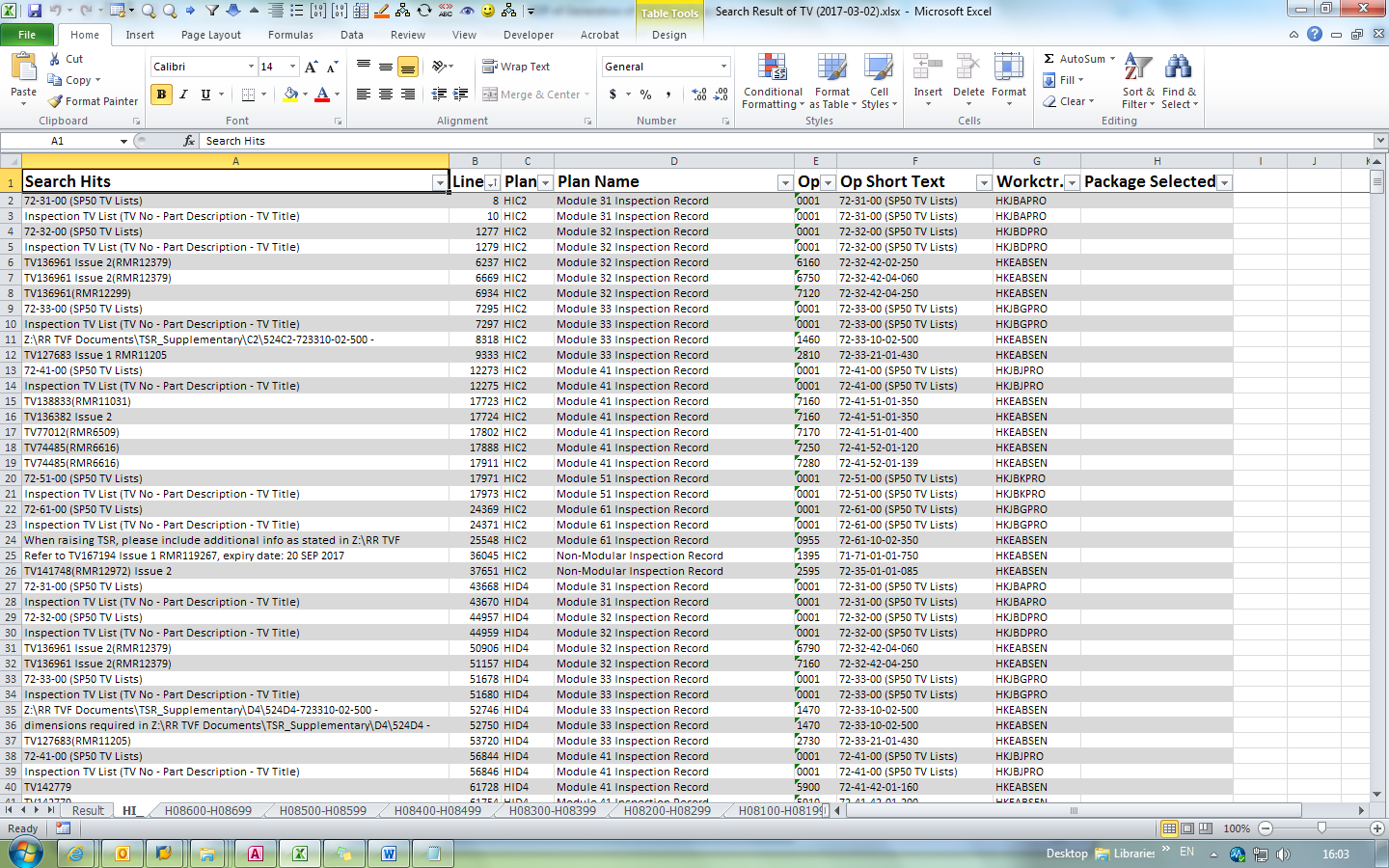
Average searching will take several minutes due to the sheer size of the data size. The time needed is dependent on:

* The size of the data collection
* The search hits found

After the search has been completed, the program will show the file location through Explorer.

Inside the file, a summary page named “Result” will be put at the top of the workbook. Other pages are the search hits in individual files.



 Fig.9 & 10. Search Summary “Result” and actual search hits.

In the page “Result”, the following information is shown:

* Files used to search with links
* Search hit count
* Total number of operations containing the search hits
* Total number of plans containing the search hits

While in other pages, the search hits are listed with related information:

* Actual “long text line” containing the search hits (Column 1)
* Line number in the raw data file containing the long text (Column 2)
* Maintenance plan code (Column 3)
* Maintenance plan title, Group Counter specific (Column 4)
* Operation number (Column 5)
* Operation Short Text (Column 6)
* Workcentre assigned (Column 7)
* Packages selected (Column 8)