

**OAP Quick Look Program**

**A Brief Overview**

The OAP Quick Look Program is a tool that can present images of cloud particles onto a multipage PDF document. This program is written in IDL language. This program allows the user to manipulate the displayed particles by changing several aspects and variables. These include probe types, start times, end times, dates, diameters, particles touching an edge, and displaying of accepted or rejected particles. The user can elect to input these variables in one of two ways. The first is through custom entry, where, the user can put in one set of conditions and one PDF will be produced. The second method is through an input file. Through the file method the user can use an indefinite number of conditions and run them each one by one. The input file must be a comma-separated values (.CSV) file. This method will produce multiple PDFs. OAP Quick Look Program is a culmination of 5 functions.

**How to Use the OAP Quick Look Program**

To use the OAP Quick Look program the user needs to input variables into the main program and the OAP\_QUICKLOOK\_GETSETUP function. The inputs that are needed are input type, file directory, delete file, parameters (if custom data is entered), and file selection (if file data is entered). The user will also need to set up their file directories and set up the input CSV file (if that is the desired input method).

Firstly, the user needs to open the main program. At the begin of this program the user needs to select the file type. There are two options that the user can select: ‘file’ or ‘custom’. This entry needs to be a string entry, capitalization does not matter. If the user is using custom data, then they will be expected to enter their desired parameters into the OAP\_QUICKLOOK\_GETSETUP function. When this input method is used only one entry will be processed each time the program is ran. If the file input method is used, then the user will be prompted to pick a CSV file. The input CSV file needs to be set up in a specific manner which is described below.

If the file entry method is being used, then the user should input their path for the file selection. This should be done using the file directory variable. This should be a string input, but it can be left blank if need be. Capitalization does matter!!

The next variable that needs to be defined is the Delete file variable. This is also found at the top of the main program. The two options for this are: ‘on’ and ‘off’. Again, these need to be strings, but capitalization doesn’t matter. It is recommended that this variable is turned off. DELETE\_FILES is used to determine whether a previously existing file should be deleted. If set to on, then a matching file names will be deleted, and a new file created. If set to off, then matching file names will have a (#) added to them. Multiple files with matching file names can exist this way.

Once the user has defined the input type and custom data then they need to set up the input variables. If the input type is custom, then the user will need to input their parameters into the OAP\_QUICKLOOK\_GETSETUP function. The first if statement contains all the parameters that need to be filled out by the user. If the user has elected to use a CSV file, then they will need to format the file. The user can have as many rows in the CSV as they would like, each row should represent a different set of parameters. The user will be promoted to choose the correct CSV file when the program begins running.

The formatting for the CSV file should be as follows. The first row should contain the title information for each column. The order of columns should be as follows. A= PROC\_FILE, B= DIMG\_FILE, C=DATE, D=START\_TIME\_SEC, E=END\_TIME\_SEC, F=DISPLAY\_PARTICLES\_TOUCHING\_EDGE, G= DISPLAY\_REJECTED\_PARTICLES, H=minD, I=maxD, J= OUTPUT\_FILE\_PATH, K= OUTPUT\_FILE\_TYPE, L= READ ROW.

DISPLAY\_PARTICLES\_TOUCHING\_EDGE and DISPLAY\_REJECTED\_PARTICLES should be set to either ‘on’ or ‘off’. minD and maxD are the minimum and maximum diameters their possible range is 0 to 15000 micrometers. Do not include units in the CSV file. OUTPUT\_FILE\_PATH should be the full path for the output file. The name of the file will be added later in the OAP\_QUICKLOOK\_FILES function. OUTPUT\_FILE\_TYPE should be a .PDF. If the user is doing multiple runs of the same date, times, and probe type but changing diameters, particles touching an edge, or rejected particles then the file path and name will be the exact same. To avoid files being overwritten or deleted an identifier should be noted at OUTPUT\_FILE\_TYPE. For example, 1.PDF or rej.PDF. READ\_ROW is a variable that is meant to be set to either ‘run’ or ‘skip’. If READ\_ROW is set to ‘skip’ then the conditions described by that row will not be ran through the program, instead the program will attempt to read the next row.

The last task that the user needs to complete is setting up their directory for the output file(s). the directory needs to match the OUTPUT\_FILE\_PATH, otherwise the program will not work.

Once the input type, delete files, parameters, and directories have all been set up and defined then the program is ready to run. The program will run till it either reaches an error it cannot overcome or until it is out of parameters to process. If there are known errors in given parameters, then the program will print the error to the IDL prompt or to the printed CSV file. These prompts and error types can be found in the error section.

**Table of Contents**

**Flow Chart**…………………………………………………………………………………………….. 4

**Full Program Description**…………………………………………………………………..5 - 12

**MAIN PROGRAM**…………………………………………………………………...5 - 8

**OAP\_QUICKLOOK\_GETSETUP**………………………………………………….8 - 9

**OAP\_QUICKLOOK\_FILES**………………………………………………...…….9 – 10

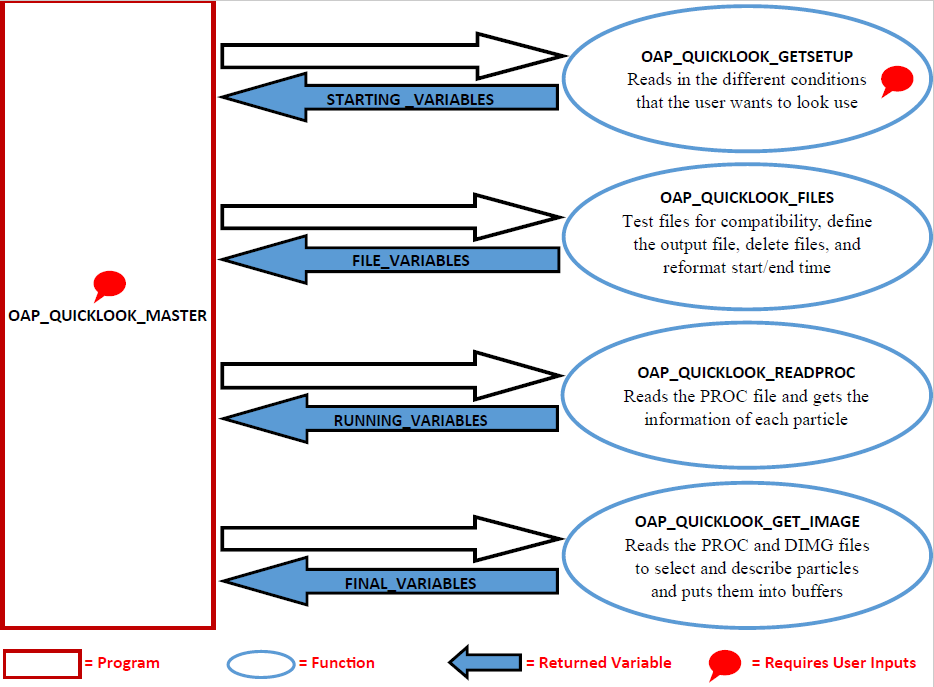
**OAP\_QUICKLOOK\_READPROC**.……………………………………………...10 - 11

**OAP\_QUICKLOOK\_GET\_IMAGE**……………………………………………..11 - 12

**The PDF File**.………………………………………………………………………………12 - 13

**Errors and Messages**…………………………………….………………………………...13 - 15

**Flow chart**



**Full Program Description**

The OAP Quick Look Program is a culmination of 5 different functions. The order for these is: the main program, OAP\_QUICKLOOK\_GETSETUP, OAP\_QUICKLOOK\_FILES, OAP\_QUICKLOOK\_READPROC, OAP\_QUICKLOOK\_GET\_IMAGE.

**MAIN PROGRAM**

The user first needs to input variables to dictate the input method and whether files should be deleted. The first prompt is INPUT\_TYPE. This variable needs to be defined as either ‘file’ or ‘custom’. If custom conditions are used, the user will need to be input their conditions into the OAP\_QUICKLOOK\_GETSETUP function. If a ‘file’ is being used, then the file type must be a comma separated values file (.CSV). The proper format for the CSV file can be found under the OAP\_QUICKLOOK\_GETSETUP section.

File directory is the next variable to be defined. This variable can be used to shorten the search for the input CSV file. This variable needs to contain the path for the CSV file that is being used. Capitalization does matter for this variable due to this being a string variable. This variable can be left with a blank string (‘ ‘), although this will likely result in too broad of a search. This variable does not matter for the custom data entry method.

The next variable that needs to be defined is DELETE\_FILE. This needs to be defined as either ‘on’ or ‘off’. This variable is used during the OAP\_QUICKLOOK\_FILES function. DELETE\_FILES is used to determine whether a previously existing file should be deleted. If the output PDF file name matches a previously existing file name, then one of two things will happen based on the user’s parameters. If the function is turned to off, then (#) will be added to the end of the output file name. The number in the parenthesis will change based on how many times the file has been recreated. If the user has elected to delete files, then any file that matches the directory and name of the output file will be deleted and the program will not stop. Again, it is recommended that the user has the DELETE\_FILE variable turned to ‘off’.

After those variables are defined the program initializes a set of variables. These variables are initialized to allow something to be passed between the main program and its functions or to start a counting mechanism that is used further in the program. Nearly all the variables in this section will get changed in different functions or in the main program.

After initializing the variables, the program starts the main while loop. The purpose of this while loop is to read multiple rows on the CSV file. It will stop the loop when the last row of the CSV has been read. At the start of the while loop there is a counter that tells us how many rows from the CSV file the program has read. The program will only go through on rendition of the while loop if custom data entry is used.

The program then starts the OAP\_QUICKLOOK\_GETSETUP function. The purpose of this function is to read in the different conditions that the user wants to use. To read more about the function, please see the OAP\_QUICKLOOK\_GETSETUP section. The main program and the OAP\_QUICKLOOK\_GETSETUP function pass 6 variables between each other to communicate. The function returns a structured variable named STARTING\_VARIABLES that consist of 11 data fields if INPUT\_TYPE is equal to ‘custom’ and 15 data fields if it is equal to ‘file’. This structured variable consists of conditions that the user would like to use. This includes PROC and DIMG file names, data, start/end times, restrictions, output file information, and a skip line option.

After completing the OAP\_QUICKLOOK\_GETSETUP function the program unpacks some of the variables from STARTING\_VARIABLES if the user is using a file as the input method. These variables keep track of progress in the .CSV file.

If the user has elected to use a file as the input method, they have the option to skip rows in the CSV file. To learn more about how to complete this step visit the OAP\_QUICKLOOK\_GETSETUP function section. If the user has elected to skip a row, then the while loop will begin again and read the next row in the CSV file.

The program then unpacks the start and end time from STARTING\_VARIABLES. It checks the start and end time to see if they are compatible. If they are not, the program will prompt the user and skip tat line in the CSV.

After testing the times, the program then starts the OAP\_QUICKLOOK\_FILES function. The purpose of this function is to test files for compatibility, define the output file path and name, delete files, and reformat start and end time. To read more about this function please see the OAP\_QUICKLOOK\_FILES section. The main program and OAP\_QUICKLOOK\_FILES function communicate using 4 variables. The function returns a structured variable named FILE\_VARIABLES that consist of 5 data fields. The variables are start/end times in HHMMSS format, output file full path and name, and selected probe type.

The function then returns to the main program. If there is a preexisting file that matches the name of the output file and the delete function is off, then the program will add a (#) to the files name and alert the user of this change.

The program then continues onto the next function titled OAP\_QUICKLOOK\_READPROC. The purpose of this function is to read the proc file and the information of each particle in the file. It will return all the information back to the main program. It also defines the buffer length and width that should be used based on the probe type. To read more about this function visit the OAP\_QUICKLOOK\_READPROC function section. The main program and OAP\_QUICKLOOK\_READPROC communicate by passing 6 variables. The function returns a structured variable called RUNNING\_VARIABLES consisting of 17 data fields. These data fields include several aspects of each particle including times, position, identifier, diameter, etc. The structure variable also has information about the buffer size.

Once OAP\_QUICKLOOK\_READPROC is complete the program unpacks several variables. The needed variables are part of two structured variables: STARTING\_VARIABLES and FILE\_VARIABLES. These variables are all descriptors of the conditions that the user has identified. These conditions will be printed at the top of the first page in the PDF document.

To print out the conditions, the program opens a window and then enters the information from those variables defined above into a designated format. The information at the top of the page is as follows; probe type, date, start-end times in seconds, start-end times in HHMMSS, minimum and maximum diameter, particles touching edge inclusion information, and rejected particles inclusion information, and general information.

After opening the first window and printing the information the program starts a second while loop. The purpose of this while loop is to print multiple buffers onto a page in the PDF and to save multiple pages to the PDF. Each rendition of the while loop is one buffer on the page. The while loop is constrained by the number of particles in the given times and conditions. Once the program is out of particles for those times and conditions then the program ends the loop.

The while loop utilizes the last function of the program, OAP\_QUICKLOOK\_GET\_IMAGE. The purpose of this function is to read through both the PROC and DIMG files to select and describe particles that are being used based on the given times and conditions. It then takes those particles and puts them into a buffer, which is returned to the main program. To read more about this function visit the OAP\_QUICKLOOK\_GET\_IMAGE function section. The main program and this function communicate using 8 variables. The function returns a structured variable titled FINAL\_VARIABLES which consists of 3 data fields. These data fields are the buffer’s data, and the start and stop time for that buffer.

Once OAP\_QUICKLOOK\_GET\_IMAGE is finished it returns to the main program. If the program has ran out of particles to display based on the given conditions, then a variable is turned on. This variable is called STOP\_VARIABLE. If this variable equals ‘on’ then the program breaks out of the while loop.

If the STOP\_VARIABLE is not equal to ‘on’ then the program resumes. Next, the file is saved and appended with a function called w. save, /append. The w. save, /append function is not in a logical place. There are two issues with having this save function in a more logical place, such as beneath where the image is created. We need to use the w. save, /append function to allow us to create multipage PDFs. To use the w. save, /append function we also need to use the w. save,/close function. There are no other closing methods that counter the /append function. Issue 1: because we must use both the /append and /close function this means that we must save twice, this leads to duplicate last pages in certain scenarios. To fix this we tried to not save the image twice if those scenarios we present, this lead to issue 2. Issue 2: we didn’t want to save twice but we must use the /append function, but there will be no /close function used. This led to the PDF never being closed and therefore multiple probe types would be on the same PDF document. The heart of these issues occurs when we have a full page of buffers on the last page of the PDF. Moving the save function to this location was the best solution that we could find. Moving the save, /append to this spot allows the program to save the full window as we normally would and before variables get changed. It also allows for the program to check to make sure there are more particles before it uses the /append function. Seeing that the w. save, /append function is beneath the break for the stop variable, the program will exit the while loop if there are no more particles before the save, /append is executed. When the program breaks out of this while loop, the only save function that can be used is the save, /close.

Next, the program unpacks the variables from the OAP\_QUICKLOOK\_GET\_IMAGE function. These include the buffer display information and the start and stop times for the buffer.

After unpacking the variables, the program opens a new window if it needs to start a new page. Once the window has been opened then it can create an image from the FILE\_VARIABLES. The image is placed into the window and put into its position based on the amount of times the program has gone through the while loop.

Once the image has been put into the window then the program resets some variables and goes through another rendition of the while loop. If there are no more particles, then it exits the loop. Once the program has exited the loop, it saves the final PDF with the w. save, /close function. This saves the PDF regardless of how many buffers are on the page. It also closes the PDF and doesn’t allow from more pages to be attached.

After saving and closing the PDF, the program resets variables to allow for another set of conditions to go through the loop. After that is done, the program restarts the first while loop to read in another row the CSV file if INPUT\_TYPE equal’s ‘file’. If there are no more rows to read from the CSV file or if INPUT\_TYPE equal’s ‘custom, it then ends the program.

**OAP\_QUICKLOOK\_GETSETUP**

The purpose of OAP\_QUICKLOOK\_GETSETUP function is to read in the different conditions that the user wants to look at. The main program and this function communicate using 6 different variables; INPUT\_TYPE, FILE\_RUN, RECORD\_COUNT, INFILE, LINE\_READ\_VARIABLE. INPUT\_TYPE is the only variable used if custom data is what the user has elected to use. If INPUT\_TYPE is equal to file, then the function utilizes 5 other variables. FILE\_RUN is the line that should be read in the CSV. RECORD\_COUNT is the total number of rows that are in the CSV file. INFILE is the path and name of the input file. LINE\_READ\_VARIABLE is whether the program should read the line in the CSV or whether it should skip it. File\_directory is the file path that is used to find the input CSV file. The OAP\_QUICKLOOK\_GETSETUP function starts by checking to see what input type is being used.

If the input type is custom, then the user can put in the conditions they would like to run. The user can change 11 options in this section. These include the following: DIMG file, PROC file, date, start time in seconds, end time in seconds, output file path, output file type, display particles touching edge, display rejected particles, minimum diameter, and maximum diameter. The program then creates a structured variable with all the variables above. This structured variable is then used throughout the rest of the main program and other functions.

If the input type is a file, then the user will be prompted to pick a file. Once the file is selected the program reads that file. It starts by reading the file to see how many rows the file contains. It then reads it again. This time the row being read is controlled by a counter that keeps track of how many times the file has been read. When the row is read the READ\_CSV function automatically creates a structured variable. The row counter, total rows, and infile are then added to the structured variable to allow them to be passed back to the main program.

The formatting for the CSV file should be as follows. The first row should contain the title information for each column. The order of columns should be as follows. A= PROC\_FILE, B= DIMG\_FILE, C=DATE, D=START\_TIME\_SEC, E=END\_TIME\_SEC, F=DISPLAY\_PARTICLES\_TOUCHING\_EDGE, G= DISPLAY\_REJECTED\_PARTICLES, H=minD, I=maxD, J= OUTPUT\_FILE\_PATH, K= OUTPUT\_FILE\_TYPE, L= READ ROW.

DISPLAY\_PARTICLES\_TOUCHING\_EDGE and DISPLAY\_REJECTED\_PARTICLES should be set to either ‘on’ or ‘off’. minD and maxD are the minimum and maximum diameters their possible range is 0 to 15000 micrometers. Do not include units in the CSV file. OUTPUT\_FILE\_PATH should be the full path for the output file. The name of the file will be added later in the OAP\_QUICKLOOK\_FILES function. OUTPUT\_FILE\_TYPE should be a .PDF. Capitalization does matter for OUTPUT\_FILE\_PATH and OUTPUT\_FILE\_TYPE. If the user is doing multiple runs of the same date, times, and probe type but changing diameters, particles touching an edge, or rejected particles then the file path and name will be the exact same. To avoid files being overwritten or deleted an identifier should be noted at OUTPUT\_FILE\_TYPE. For example, 1.PDF or rej.PDF. READ\_ROW is a variable that is meant to be set to either ‘run’ or ‘skip’. If READ\_ROW is set to ‘skip’ then the conditions described by that row will not be ran through the program, instead the program will read the next row.

**OAP\_QUICKLOOK\_FILES**

The purpose of this function is to test files for compatibility, define the output file path and name, delete files, and reformat start and end times. The main program and this function communicate using 4 different variables; STARTING\_VARIABLES, DELETE\_FILE, FILE\_DELETE\_STOP\_VARIABLE, and File\_run. STARTING\_VARIABLES is the structured variable that was created in OAP\_QUICKLOOK\_GETSETUP. DELETE\_FILES is the variable that tells the program whether it can delete a file or not. FILE\_DELETE\_STOP\_VARIABLE is a variable that tells the program whether to stop if a file matching the output file name already exist and DELETE\_FILE is turned to ‘off’. If FILE\_DELETE\_STOP\_VARIABLE is turned to ‘on’ then the matching file name will be deleted. File\_run is a counter that tells the user which line in the CSV they are on if an error occurs.

This function starts by unpacking variables that were created in the OAP\_QUICKLOOK\_GETSETUP. After the variables have been unpacked, the function checks to see if the PROC and DIMG file have matching probe types. If the files match, then it proceeds to set the probe type. The options are H.2DS, V.2DS, HVSP, CIP, and CIPG. The program then tests to make sure that there is a supported probe type.

If the files doe does not match, then an error will be prompted in IDL and that line of the CSV will be skipped.

After setting and checking the probe type, the function changes the start and end times. It converts the times from seconds to HHMMSS format.

Once the times have been changed then the function sets the output file name and path. The path and type of the file was defined in the OAP\_QUICKLOOK\_GETSETUP. The full name of the file contains information such as date, start and end times in HHMMSS, and probe type.

After setting the name of the output file the function checks to see if a file matching its path and name already exist. If the file exists with a matching name to the output file, then the function has two options delete or rename. If the user has elected to allow for files to be deleted, then the function gives the line number for the CSV, deletes the file, and resumes. If the user has not elected to delete files, then the program will add (#) to the end of the file name. this number is based on how many times the fille name has already been duplicated. The function will prompt the user of the change and tell them which line in the CSV it occurred on, then it will resume.

Once the file has been tested and possibly deleted or changed then the program creates a structured variable. This structured variable has the new start and end times, probe types, and output file path and name. After the structured variable has been created it returns to the main program.

**OAP\_QUICKLOOK\_READPROC**

The purpose of the OAP\_QUICKLOOK\_READPROCis to read the proc file and the information of each particle in the file. It will return all the information back to the main program. It also defines the buffer length and width that should be used based on the probe type. This function and the main program communicate using 6 variables. These include STARTING\_VARIABLES, FILE\_VARIABLES, FIRST, LAST, ERROR\_VARIABLE, and File\_run. STARTING\_VARIABLES and FILE\_VARIABLES are both structured variables that were made in previous functions. FIRST and LAST variables are the first and last possible particles that are within the given times. ERROR\_VARIABLE tells the main program if there was an issue with OAP\_QUICKLOOK\_READPROC. File\_run tells the function what line of the CSV is on.

The OAP\_QUICKLOOK\_READPROC function starts by unpacking several variables from FILE\_VARIABLES and STARTING\_VARIABLES. After unpacking the variables, the program opens the PROC file.

Once the PROC file is open then the function reads the PROC file to find the times of all the particles in that file. After the times of each particle has been described then it defines the Start\_time\_index and End\_time\_index., which are later defined as the first and last possible particles the user is interested in.

For the date of 08/10/2020, our HVPS probe was running 140 seconds behind. There is a section of code that compensates for that.

After defining the Start\_time\_index and End\_time\_index, the function tests whether the start and end times make sense in comparison with the file. There are 4 errors that can occur. 1: The desired start and end times are outside of the proc files times. 2: The start time is less than or equal to the first particle in the file, but the end time is ok. 3: The end time is greater than or equal to the last particle in the proc file, but the start time is ok. If one of these three errors occur, then that line of the CSV will be skipped. A file will still be made and will contain the error information. The user will be notified in the IDL prompt window that an error has occurred and what row it occurred on. 4: The start time is less than the start time of the PROC file and the end time is greater than the end time of the PROC file. If this error occurs than is assumed not to be a mistake. The whole PROC file will be read, although it will take some time. The user will be alerted of this and told what row in the CSV the error occurred on.

The function reads then proc file to find the position of each of the particles on the buffer. It then finds the slice count, parent record number, image touching edge, image auto reject, and image diameter of each of the particles in the PROC file.

Once each particles characteristics’ have been described, it then sets up the buffer specifications. The buffer length and width are based on the probe type.

After the buffer length and width have been described, the function defines the structured variable that it is going to return to the main program. This structured variable called RUNNING\_VARIABLES contains 16 data fields. The data fields are as follows; HHMMSS, start time index, end time index, position, slice count, record number, particle touching edge, auto reject, diameter, time display, position display, particle count, display parts, data length, data width, temporary length, and temporary width.

After the structured variable has been defined then the function returns it to the main program.

**OAP\_QUICKLOOK\_GET\_IMAGE**

The purpose of the OAP\_QUICKLOOK\_GET\_IMAGE function is to read through both the PROC and DIMG file to select and describe particles that are being used based on the given times and conditions. It then takes those particles and puts them into a buffer, which is returned to the main program.

The OAP\_QUICKLOOK\_GET\_IMAGE function and the main program communicate using 8 variables. The first 3 are structured variables that were produced by the three pervious functions. These include STARTING\_VARIABLES, FILE\_VARIALBES, and RUNNING VARIABLES. The other variables that are used to communicate and are FIRST, LAST, STP, TOT\_SLICE, and STOP\_VARAIBLE. FIRST and LAST are the first and last possible particles for the given times. STP is the last particle that was displayed in the previous buffer. TOT\_SLICE, is the total amount if slices the previous buffer held. STOP\_VARIABLE is used if there are no more particles left in the function.

The function starts by unpacking the variables from the structured variables STARTING\_VARIABLES, FILE\_VARIALBES, and RUNNING VARIABLES. It then opens the PROC and DIMG file to be read.

For this program the accepted auto\_rejects are 48, 104, 72, 117, 82.

After opening the files, the function begins its first for loop. This for loop determines whether the particles in the proc file meet the user’s conditions such as times, diameter, particles touching an edge, and rejected particle. The for loop continues till the function has found enough particles to fill one buffer. The program then determines the start and stop time of the buffer.

If the function has reached this point and STP is equal to -1 then that means that there are no particles that fit the user’s conditions that can start to fill a buffer. If this is the case, then the STOP\_VARIABLE will be turned to ‘ON’. Then the program prints a statement and returns to the main program.

If the STOP\_VARIABLE is not turned to ‘ON’ then it continues with the function. The next objective of the function is to read the DIMG file. The function reads in all the particles from the DIMG file. It then starts a new for loop. This loop sorts through all the particles in a given time and accepts them or rejects them based on the user’s conditions. This loop is a near exact replica of the for loop used at the beginning of this function. The particles that fit the user’s conditions are put into a buffer.

After the buffer has been filled, the function transposes the data into a displayable image. This is done through a case statement based on the probe type. HVPS and 2DS probes record data in the same way and there for are grouped together. CIP and CIPG probe types record their data differently than each other and from HVPS and 2DS, and therefore are broken up into different statements. The case statement returns a variable called data\_record which formatted displayable buffer information.

After the function has finished the case statement it builds a structured variable consisting of three variables. The three variables are a DATA\_RECORD, TIME\_DIS\_STT, and TIME\_DIS\_STP. The TIME\_DIS variables are the times in seconds of the first and last particles in that buffer. These times will be displayed above the buffer on the printed page.

Once the function has created the structured variable it returns to the main program with the information from the structured variable.

**The PDF File**

This section covers the information that is included in the PDF file. At the top of the first page of each PDF there is section dedicated to the parameters that decided the particles to be displayed. Viewing the pdf, the information should be displayed similar to this.

PROBE TYPE-

DATE- (in yyyymmdd format)

Times in seconds (start-end) ##### - ##### UTC

Times in hhmmss (start-end) ###### - ###### UTC

Diameter in microns (min-max) = ###### - #####

Only displaying particles NOT touching an edge |or| Displaying particles touching an edge

Only displaying accepted particles |or| Displaying both accepted and rejected particles

*All particles are displayed when diameters are set to 0-15000, particles touching an edge, accepted, and rejected particles are displayed.*

If there errors or no particles within the given parameters, then a message should be stated below this the title information. The different errors and messages can be found in the Error Types section of this article.

If there are no issues with the program and there are particle within the given parameters, then at least one buffer should be displayed. The buffer should have a time stamp on it. This time stamp is in hhmmss UTC format. The first time is the timing of the first particle in the buffer, while the last time corresponds to the timing of the last particle in the buffer. If the first page is full then there should be 9 buffers displayed. If there are enough buffers to fill consecutive pages, then those pages will have 10 buffers displayed.

**Errors and Messages**

This section discusses all the possible errors and messages that are known to the program, their respective outputs, and how to correct them.

Main Program

1. Output message on IDL: ‘Start\_time is greater than End\_time, input new values’

‘Occurred on row # of the CSV file’

Error: The given start time is greater than the end time, and thus the program cannot be run. The program will skip these times and move on to the next line of the CSV or end the program.

Solution: Make sure that the start time is less than the end time.

1. Output message on PDF: 'There are no particles within the given condition’

Error: There is no error here. If there are no particles within the given parameters, then the message above will be displayed on the PDF. This just alerts the user that there was no error in the program or parameters.

Solution: There is no solution needed.

OAP\_QUICKLOOK\_GETSETUP

1. Output message on IDL: ‘A file must be selected’

‘Ending program’

Error: If the user has elected to use a file as the input method then they will be asked to select a file with a dialog box. If the user exits or cancels before a file is selected, then there will be no file selected. The program will end.

Solution: The user can either change the input type to ‘custom’ or they can restart the program. When prompted select the desired CSV file.

OAP\_QUICKLOOK\_FILES

1. Output message on IDL: ‘DIMG and PROC unmatched, check probe type and spelling’

‘Occurred on row # of the CSV file’

Error: The probe type associated with the given PROC and DIMG files do not match. This will cause the program to skip the current line in the CSV and move on.

Solution: check the endings of the PROC and DIMG files to make sure that they match. Capitalization, spelling, and probe type all need to match.

1. Output message on IDL: ‘Unsupported probe type, line skipped’

‘Occurred on row # of the CSV file’

Error: The given probe type on the PROC file does not match a usable probe type for this program. The only useable probe types are 2DS, HVPS, CIP, CIPG (the program sees CIPG as ‘cip’ when in the proc file).

Solution: check the spelling, capitalization and probe type in the given proc file name. Make sure they match the possible probe types.

1. Output message on IDL: 'This file already exist and File\_delete indicates that you do not want to delete file.’

‘(#) will be added to the end of the output file

‘Occurred on row # of the CSV file’

Error: There is no error here, this message is just to alert the user that the end of the file has changed because a file already exists.

Solution: No solution is needed

1. Output message on IDL: ‘the file: output file name’

‘has been deleted because it matches the output files name’

‘Occurred on row # of the CSV file’

Error: there is no error here. This message us to alert the user that a file has been deleted to make sure that the file that the program is making can exist.

Solution: No solution is needed

OAP\_QUICKLOOK\_READPROC

1. Output message on IDL: ‘The start time is less than the start time of the proc file and the end time is greater than the end time of the proc file.'

'The whole proc file will be read. This may take a while'

Error: It is unlikely that this is an error here. The user likely plugged in times that were before and after the start and end times of the file. This will result in whole file being read, which can take a very long time.

Solution: If the user doesn’t want the whole proc file to be read then they should completely stop the program. They should then enter new times and skip any lines they have already read and resume the program. Otherwise the user can just let the program run.

1. Output message on IDL: ‘An issue occurred on row # of the CSV file’

‘Please see the output file for more information’

Error: this error could be caused by 3 different scenarios which are broken down below

1. Output message on PDF: ‘Issue: The desired start and end times are outside of the proc files times.'

Error: Both the start time and the end time are either before or after the file. This means that no particles will be found, and the program will not run correctly. The line in the CSV that this error occurred on will be skipped.

Solution: Make sure that the desired times are within the proc file times.

1. Output message on PDF: 'The start time is less than or equal to the first particle in the file, but the end time is ok.'

Error: The given start time is less than or equal to the first time in the PROC file. This will cause there to be missing desired particles. This is considered an error. The line in the CSV that this error occurred on will be skipped.

Solution: Make sure that the desired start time occurs after the start of the file.

1. Output message on PDF: ‘'The end time is greater than or equal to the last particle in the proc file, but the start time is ok.'

Error: the given end time is greater than or equal to the last particle in the proc file. This will cause there to be missing desired particles. This is considering an error. The line in the CSV that this occurred on will be skipped.

Solution: Make sure that the desired end time is before the end of the file.