General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)
ROBOT DISPLAY AND CONTROL PROGRAM (ROBDAC)

By: William A. Stewart

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Johnson Space Center
Houston, Texas 77058

July 1976
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBROUTINES DEFINITIONS</td>
<td>1</td>
</tr>
<tr>
<td>Main Display Routines</td>
<td></td>
</tr>
<tr>
<td>Utility Display Routines</td>
<td></td>
</tr>
<tr>
<td>SUMMARY AND INTRODUCTION</td>
<td>2</td>
</tr>
<tr>
<td>PROGRAM DESIGN</td>
<td>4</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>20</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>21</td>
</tr>
</tbody>
</table>

iii
SUBROUTINES DEFINITIONS

Main Display Routines

CONTRL Generates ROBOT Control Menu
PARIN ROBOT Parameter Input Selected in Control
ARRIN ROBOT Array Input Menu
ARRAY STD Array Edit Routine Called from ARRIN (used
34 times in ROBOT arguments - NAME, VALUE, NDP,
DESC, NWD)
TABIN ROBOT Table Input Menu
PLOTIN Controls Input Data Plots
SARDAT Store Plus Retrieve Data
TRAJDAT Setup of Trajectory Plot Data
TRAJPLT Display Plot of Trajectory Output Parameters
NUMTRJ Display Trajectory Output Parameters and Values
NUMCON Display Convergence Output Parameters and Values
CONVPLT Display Plot of Convergence Output Parameters
PERFSUM Display Performance Summary Parameters and Values
DATRAN Data Transformation and Format

Utility Display Routines

DISTAB Display Table of Values
DISPLT Display Plot of ROBOT Input Parameters
DISSCL Display Scale Factors
DISPOP Display Plot Options
SUMMARY AND INTRODUCTION

This report is a preliminary design of the ROBOT Display and Control Program (ROBDAC) to be written for the Adage 340 computer. This program is designed to communicate with the Univac 1110 computer with the aid of the Graphic Support Communications Programs (GS COMM Package). The GS COMM Package allows data to be shipped and received between the Univac and Adage computers when the data meets input/output requirements necessary for a transfer. ROBDAC and the Adage computer act as an input/output device for the program ROBOT, which is run on the Univac 1110.

At first, consideration was given to running the ROBOT Program on the Adage. However, due to the size of ROBOT, and the time and accuracy involved in its calculations, this idea was dropped. ROBDAC controls the input to ROBOT by displaying images of the inputs, providing an editing technique for these inputs, shipping the input data to the Univac, and then commanding ROBOT to start execution. The output is done in a similar manner after receiving the output data from the Univac. This data will be displayed numerically as well as plotted graphs. All of the input/output will be done interactively with practically "finger tip" control. A general program flow diagram of ROBDAC is shown in figure 1.
Figure 1: Robot Display and Control Program
The intent of the ROBDAC Program is to interactively communicate from the Adage 340 with the ROBOT Program on the Univac 1110. The data transfer between the two computers is handled by the GS COMM package. This package is a series of programs on both computers that allows data to be sent, tested and received so that the main programs running on each computer are not interfered with or received erroneous data. A description of the GS COMM programs are in Appendix A. The ROBDAC Program consists of several images which aid in the analysis of input/output ROBOT data.

The first display image is the ROBOT CONTROL MENU. This image consists of input/output options and commands to start ROBOT on the Univac (refer to figure 2). Any option or command is selected by depressing its corresponding function switch. If an option is selected that needs additional input, the ANK is the input device. As noted in figure 2, there are eight input options, of which only the first four will be discussed at this time. The last four options include the same inputs as the first four, but these inputs are grouped in different array images to be designed later.

The first option is LOAD DATA which loads the desired data base from disk into Adage core. Additional input will be typed in from the ANK telling from what pack/volume and file name the data base is stored.

The PARAMETER INPUT option is a display of one element array inputs to ROBOT. Figure 3 shows an example format of the parameters. The parameters are listed alphabetically with their values, units and descriptions. All of the input parameters cannot be displayed at one time, therefore, function switch 31 will page the data for display of additional input parameters. Any parameter value may be changed by depressing the function switch associated with that parameter and typing in the change from the ANK. When all parameters are set properly, function switch 32 will return to the control menu.

The ARRAY INPUT option is a display shown in figure 4. This option displays all of the multiple element arrays, the number of elements in each array, and the array description. The commands at the bottom of the display are the same as described earlier. Upon depressing a function switch associated with an array name, a new display appears as shown in figure 5. This display is an example of the array PRINT with each element and its value listed. The description is the same for all elements. Any array element can be selected with its function switch and the array value changed via the ANK.
<table>
<thead>
<tr>
<th>FNS</th>
<th>INPUT OPTIONS</th>
<th>FNS</th>
<th>OUTPUT OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LOAD DATA</td>
<td>16</td>
<td>STORE AND RETRIEVE DATA</td>
</tr>
<tr>
<td>2</td>
<td>PARAMETER INPUT</td>
<td>17</td>
<td>TRAJECTORY PLOT SETUP</td>
</tr>
<tr>
<td>3</td>
<td>ARRAY INPUT</td>
<td>18</td>
<td>PLOT TRAJECTORY</td>
</tr>
<tr>
<td>4</td>
<td>TABLE INPUT</td>
<td>19</td>
<td>NUMERIC TRAJECTORY</td>
</tr>
<tr>
<td>5</td>
<td>THRUST EVENT DEP. INPUT</td>
<td>20</td>
<td>PLOT CONVERGENCE</td>
</tr>
<tr>
<td>6</td>
<td>OPTIMIZATION INPUT</td>
<td>21</td>
<td>NUMERIC CONVERGENCE</td>
</tr>
<tr>
<td>7</td>
<td>GRAVITATIONAL INPUT</td>
<td>22</td>
<td>PERFORMANCE SUMMARY</td>
</tr>
<tr>
<td>8</td>
<td>INITIAL CONDITIONS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>START TRAJECTORY RUN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>CONTINUE TRAJECTORY RUN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>STOPS PROGRAM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 2**  CONTROL DISPLAY.
<table>
<thead>
<tr>
<th>ROBOT PARAMETER</th>
<th>INPUT VALUE</th>
<th>UNITS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNS NAME</td>
<td>ALONGO</td>
<td>DEG</td>
<td>Longitude of Launch Site</td>
</tr>
<tr>
<td>1</td>
<td>80.5649</td>
<td>0.002</td>
<td>Forward Integration Error Check</td>
</tr>
<tr>
<td>2</td>
<td>0.003</td>
<td>90.0</td>
<td>Launch AZ IMUTH</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Robot Array Input

<table>
<thead>
<tr>
<th>ME NAME</th>
<th>NUM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>15</td>
<td>Engine Exit Area for Vehicle 1 (Orbiter)</td>
</tr>
<tr>
<td>AEZ</td>
<td>15</td>
<td>Engine Exit Area for Vehicle 2 (Booster)</td>
</tr>
<tr>
<td>BSTEP</td>
<td>15</td>
<td>Backward Integration Step Size</td>
</tr>
</tbody>
</table>

**Figure 4** Input Array Display.
## ARRAY PRINT

<table>
<thead>
<tr>
<th>FNS</th>
<th>ELEMENT</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2.</td>
<td>PRINT TIME INCREMENT FOR EACH THRUST EVENT</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>8.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>0.</td>
<td></td>
</tr>
</tbody>
</table>

31 PAGE THE DATA  (THIS SPACE AVAILABLE)
32 RETURN TO ARRAY LIST  (FOR ANY INPUT)

**FIGURE 5**  ARRAY EXAMPLE DISPLAY.
TABLE INPUT is the last input method discussed and its display is shown in figure 6. The display lists the seven tables of ROBOT input with a brief description of each. The X-name is the independent variable plotted along the X-axis and there are as many as three Y-names which are the dependent variables plotted along the Y-axis. An example of selecting the second table, which has three dependent variables, is shown in figure 7. This display writes the name of the table at the top of the CRT and calls four subimages. The subimages are listed as utility display routines in the subroutines definitions. DISTAB displays a table of values in the upper right hand corner of the CRT. The example in figure 7 shows PNM and CAN values listed. The plot of PNM VS. CAN in the upper left hand corner is displayed by the image DISPLT. The scale for the plot defaults to equally segmented grids between the maximum and minimum values that are plotted. A user has the capability to change the scale with the image DISSCL displayed in the lower right hand corner. The maximum and minimum values and the number segments can be changed via the function switches. In the lower left hand corner DISPOP is displayed which is the options of this input technique. Either one of the other two dependent variables of this table can be selected and plotted via the function switches. The numeric values of the selected plot will change accordingly. The option GLOMMER allows a user to move a point on the plot to another location. The SCROLL option will move additional numeric data points into view if a table has more than fifteen (15) points. When all inputs are satisfied with the selected table, function switch 32 will return to the table input for another table selection.

The first option in the output is STORE AND RETRIEVE DATA shown in figure 8. The options here allow a user to store or retrieve specified data via the function switches. For example, a user might want to retrieve a data base EDIN0620. This would be accomplished by depressing function switch 2, then 3. A message at the lower right hand corner will appear asking for the name of the data base and the pack/volume where it is located. After the computer has time to read the data into core function switch 32 will return to the control menu.

The TRAJECTORY PLOT DATA lists all of the output parameters of ROBOT as well as their values, units and descriptions (refer to figure 9). Two parameters need to be selected via the function switches, then by depressing function switch 30, they will be plotted. The first parameter selected will be plotted along the X-axis and the second along the Y-axis. Function switch 29 is a reset option if a wrong parameter is selected and a user wishes to start over.
<table>
<thead>
<tr>
<th>FNS</th>
<th>X-NAME</th>
<th>Y-NAME</th>
<th>Y-NAME DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PFM</td>
<td>CNF</td>
<td>POWER OFF AERODYNAMICS = F(MACH)</td>
</tr>
<tr>
<td>2</td>
<td>PNM</td>
<td>CJOTAB</td>
<td>POWER ON AERODYNAMICS = F(MACH)</td>
</tr>
<tr>
<td>3</td>
<td>XMACCH</td>
<td>CMOTAB</td>
<td>PITCH AERODYNAMICS = F(MACH)</td>
</tr>
<tr>
<td>4</td>
<td>FABAL</td>
<td></td>
<td>BASE DRAG = F(ALT)</td>
</tr>
<tr>
<td>5</td>
<td>TT1</td>
<td>WPTN</td>
<td>THRUST AND FUEL FLOW = F(TIME)</td>
</tr>
<tr>
<td>6</td>
<td>WITAB</td>
<td>XCGTAB</td>
<td>CG = F(WEIGHT)</td>
</tr>
<tr>
<td>7</td>
<td>TTBL</td>
<td>CPTBL</td>
<td>CONTROL TABLES = F(TIME)</td>
</tr>
</tbody>
</table>

**Figure 6**

Input Table Display.
POWER-ON AERODYNAMICS = F(MACH)

Figure 7 Table Example Display.

<table>
<thead>
<tr>
<th>FNS</th>
<th>OPTIONS</th>
<th>FNS</th>
<th>OPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>PNM VS CAN</td>
<td>25</td>
<td>GLOMMER</td>
</tr>
<tr>
<td>23</td>
<td>PNM VS CN OTAB</td>
<td>26</td>
<td>SCROLL</td>
</tr>
<tr>
<td>24</td>
<td>PNM VS CNN</td>
<td>27</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FNS</th>
<th>X-Axis</th>
<th>Y-Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIN</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>MAX</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>SEG</td>
<td>20</td>
<td>21</td>
</tr>
</tbody>
</table>

RETURN TO TABLE INPUT
## Store and Retrieve Data

<table>
<thead>
<tr>
<th>FNS</th>
<th>Options</th>
<th>PUV</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STORE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RETRIEVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DATA BASE</td>
<td>222</td>
<td>EDIN0620</td>
</tr>
<tr>
<td>4</td>
<td>TRAJECTORY DATA</td>
<td>110</td>
<td>TRAJ0620</td>
</tr>
<tr>
<td>5</td>
<td>CONVERGENCE DATA</td>
<td>110</td>
<td>CONV0620</td>
</tr>
</tbody>
</table>

32. Return to Control Menu  
(THE SPACE AVAILABLE FOR ANY INPUT)

**Figure 8** Store and Retrieve Display.
TRAJECTORY PLOT DATA

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>UNITS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0</td>
<td>SEC</td>
<td>TRAJECTORY TIME</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>FT/SEC</td>
<td>VELOCITY</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>FT</td>
<td>VELOCITY</td>
</tr>
</tbody>
</table>

29 RET - CHANGE PARAMETERS TO BE PLOTTED
30 PLOT - TWO PARAMETERS NEED TO BE SELECTED
31 PAGE TRAJECTORY DATA
32 RETURN TO CONTROL MENU

FIGURE 9  TRAJECTORY PLOT SETUP DISPLAY.
The TRAJECTORY PLOT display plots the two parameters selected in TRAJECTORY PLOT DATA, as shown in figure 10. The plotting format and method are the same as in the input table plot image.

The NUMERIC TRAJECTORY is a display of the ROBOT output parameters, values and units for specific observation points. Options at the bottom of the display steps or scrolls the values for other iteration points forward or backward. Figure 11 shows an example of this display.

The CONVERGENCE PLOT display plots the optimization parameter and its terminal constraints. Figure 12 shows an example with GLOW being the optimization parameter and the other four parameters are constraints. A series of plots of the parameters can be selected via the function switches in the lower right hand corner.

The NUMERIC CONVERGENCE displays the numeric values of the convergence parameters as shown in figure 13. Additional parameters are displayed to aid a user when a particular trajectory might not be converging. The commands at the bottom of the display have the same function as in the NUMERIC TRAJECTORY.

The last display is the PERFORMANCE SUMMARY in figure 14. This is a list of critical trajectory parameters and their values at the end of each stage that is available.
FIGURE 10
TRAJECTORY PLOT DISPLAY.

RETURN TO TRAJECTORY DATA
FOR ANK INPUT

TRAJECTORY PLOT

<table>
<thead>
<tr>
<th>T</th>
<th>VEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>8000.0</td>
</tr>
</tbody>
</table>

X-Axis: FNS Y-Axis: FNS

SEGS: 16 17 18 19 20

TEXT AVAIL: FOR ANK INPUT
FIGURE 11  NUMERIC TRAJECTORY OUTPUT DISPLAY.
Figure 12

Convergence Plot Display.

Convergence Plot

FUS PLOT
1 NEW PLOT
2 OLD PLOT
3 ASK PLOT
4 GOT PLOT
5 PCT PLOT

Trajectory Plot

Return to Control Menu
### Numeric Convergence

<table>
<thead>
<tr>
<th></th>
<th>GLOW</th>
<th>VEL</th>
<th>GAM</th>
<th>R</th>
<th>MASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>-87809+06</td>
<td>-59843+03</td>
<td>26724+01</td>
<td>38026+06</td>
<td>-11031+04</td>
</tr>
<tr>
<td>NEW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>DPAR</th>
<th>OLD PCON</th>
<th>NEW PCON</th>
<th>P SCALE</th>
</tr>
</thead>
<tbody>
<tr>
<td>.3626</td>
<td>3504-01</td>
<td>.1349</td>
<td>3455-01</td>
<td>.1766</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSUBI</td>
<td>.47853699-02</td>
<td>.29480213+04</td>
<td>.82644805-11</td>
<td></td>
</tr>
<tr>
<td>WIBT</td>
<td>.10000000+01</td>
<td>.10000000+01</td>
<td>.10000000+01</td>
<td></td>
</tr>
<tr>
<td>X KAY</td>
<td>.10000000+01</td>
<td>.10000000+01</td>
<td>.10000000+01</td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>.97337738+00</td>
<td>.10000000+01</td>
<td>.10000000+01</td>
<td></td>
</tr>
<tr>
<td>DEL CHIP</td>
<td>.4504-02</td>
<td>.10000000+01</td>
<td>.10000000+01</td>
<td></td>
</tr>
</tbody>
</table>

28 STEP FORWARD
29 STEP BACKWARD
30 SCROLL FORWARD
31 SCROLL BACKWARD
32 RETURN TO CONTROL MENU

**Figure 13** Numeric Convergence Output Display.
<table>
<thead>
<tr>
<th>STAGE</th>
<th>GROSS ENG. WT.</th>
<th>CUT-OFF W. T.</th>
<th>THRUST</th>
<th>BURN TIME</th>
<th>FUEL BURNT</th>
<th>DEL V RESERVES</th>
<th>STAGE WT.</th>
<th>MASS FRACTION</th>
<th>DELTA V</th>
<th>PAYLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>25.0723.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>II</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>III</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 14

Performance Summary Display.
CONCLUSION

This preliminary design document serves as a foundation as to how the ROBDAC Program will be written. In no way should this document be taken as an official design criteria that can not be deviated from. As changes to this design are made or supplemental options added, they will be written in the form of a NASA memorandum so that a complete and final document can be written at the end of the project.
APPENDIX A - GRAPHIC SUPPORT COMMUNICATIONS PROGRAMS.

COMMUNICATION CONTROL SUBPROGRAMS

OUTLINE

The GS (Graphic Support) subprograms used for data transfer between application programs written on the host processor (1100 Series) and the display processor (ADS 100/300 Series) individually and used for program synchronization are described in this section.

The statuses of communication are classified as follows:

1. The initiation of the communication is declared from the host and display processor sides respectively.

2. The report data are sent from the host processor side to the display one or vice versa.

3. The existence of communication information sent from the opposite side is checked.

4. The report data sent from the opposite processor are received.

5. The end of communication is declared in the host and display processor sides respectively.

GS subprograms used for communication are as follows:

1. GSGRH ---- This declares the initiation of communication.

2. GSCLS ---- This declares the end of communication.

3. GSRPT ---- This sends the report data.

4. GSENG ---- This checks the existence of the communication information sent from the opposite side.

5. GSGRB ---- This receives the report data.

The way (specification) to use the GS subprogram on communication is common to both host and display processor sides.

The outline of communication is as follows:

<table>
<thead>
<tr>
<th>Host Processor Side</th>
<th>Display Processor Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1100 Series)</td>
<td>(ADS 100/300 Series)</td>
</tr>
<tr>
<td>User Program</td>
<td></td>
</tr>
<tr>
<td>GSGRH</td>
<td>Initial Synchronization</td>
</tr>
<tr>
<td>GSGRH</td>
<td></td>
</tr>
<tr>
<td>GSCLS</td>
<td></td>
</tr>
<tr>
<td>GSRPT</td>
<td></td>
</tr>
<tr>
<td>GSGRB</td>
<td></td>
</tr>
<tr>
<td>GSGRH</td>
<td></td>
</tr>
<tr>
<td>GSGRH</td>
<td></td>
</tr>
<tr>
<td>GSCLS</td>
<td></td>
</tr>
</tbody>
</table>

Each GS subprogram is described in the text as follows:

1. Purpose

The content of the job performed by each subprogram is described briefly.
Format
The general format to call each subprogram including arguments is described below:
(a) The underlined arguments indicate that values are returned to them by GS. Therefore, these arguments have to be variables or arrays.
(b) The arguments enclosed by brackets [ ] indicate that they can be omitted.
(c) The argument "$n$" can be used only for the GS subprogram in the host processor side. The argument "$n$" cannot be specified for the GS subprogram in the display processor side.

General Description
The use of this job is described below.

INITIATION AND TERMINATION
GSOPH
(a) Purpose
This subprogram declares the initiation of the communication control GS subprograms.
(b) Format
CALL GSOPH (["$n"] istat)
where,
$\$n$: When an error has occurred, the control is returned to the statement number specified here. When omitted, the control is returned to the next line.

istat: The error code caused in calling all GS subprograms is stored in this area. (Integer Variable)
(c) General Description
(i) This subprogram initializes the communication line and synchronizes the communication program with the opposite processor.
(ii) The user program must call GSOPH without fail prior to every use of the communication control GS subprograms (except for GSOPH itself.)
GSCLS

(1) Purpose
This subprogram declares the termination of the communication control GS subprograms.

(2) Format
CALL GSCLS (I, n )

(3) General Description
(a) This subprogram sends a termination message to the opposite processor and releases the communication line.

(b) After the completion of (a), the calling of other communication control GS subprograms than GSOPN cannot be accepted.

(c) The sent termination message is accepted when the opposite user program called the "GSEND" or "GSCLS" subprogram.

COMMUNICATION CONTROL

GSREP

(1) Purpose
This subprogram sends the report data.

(2) Format
CALL GSREP (I, n, last, data, count[], type[])

where,

last : Specify whether the data block sent now is the last one or not.
\( \emptyset \) : Data to send still remain.
1 : This data is the last.

data : Data to send. Constant, variable, or array.

count[] : The amount of data to send.

(type[] : Specify the data type. (If omitted, the value \( \emptyset \) is assumed.)
\( \emptyset \) : Integer
1 : Real number
2 : Character

In case of integer, the absolute value must not be more than \( 2^{32} - 1 \).

In case of real number, the number of decimal places is 11.

In case of character, the length of the string is up to 80.

In case of data to send, the total number of data blocks including this one must not exceed 256.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.

The total number of data blocks sent must not exceed 256.

The total amount of data sent must not exceed 4096 bytes.
(3) General Description

(a) The report data is transferred to the user program of the opposite processor in the sent order of the report data.

(b) Whether the report data is sent from the user program of the opposite processor or not is checked by the GSEND subprogram.

(c) GSGB is called to pick up the sent report data.

(d) One report data block is completed when the last data constructing the block ("last = 1") is transferred. Only completed report data are transferred to the user program.

(4) GSEND

(1) Purpose

This subprogram checks the existence of the communication information.

(2) Format

CALL GSEND ([5n,] check, lIVPD)

Where,

- check : # : When this value is specified, this subprogram waits until any information is sent from the opposite user program.
- 1 : When this specified, this subprogram examines whether the information is sent from the opposite user program.

- lIVPD : Specify the integer variable indicating the sort of information sent from the opposite user program.
- # : No information is being sent.
  (Only for "check" = 1)
- 1 : Report data
- 2 : The opposite user program terminates (The GSCLS subprogram was called.)
- 3 : The opposite user program is waiting (only on AGS side.)
General Description

(a) This subprogram examines the sort of communication data sent by the opposite user program.

(b) When no communication data is sent:
    If "check" = φ, G3 waits until the communication data is sent and it does not return control to the user program. If "check" = 1, G3 returns the control to the user program immediately specifying "itype" = φ.

(c) When the report data is sent:
    "itype" is set to be 1. Then, the corresponding report data can be picked up by the GSRRB subprogram.

(d) When the call "check" = φ was performed on the display processor side and the call was also performed on the host processor side, "itype" is set to 3 and immediately control is returned on the display processor side. The host processor continues to wait.

(e) After the report data (not the last, "last" = φ) was sent, this subprogram cannot be called without sending the last report data ("last" = 1.)

(f) When both incomplete report data and "GSCLS" information sent from the opposite user program exist in calling this subprogram, only the latter is transferred, the former being missed.

(g) When "GSENQ" subprogram was called before picking out the report data sent from the opposite user program, the report data is lost.

(h) When the opposite user program called "GSCLS" subprogram, "itype" is set to be 2. When "itype" = 2 on the host processor side, the communication is automatically terminated, so that the user program does not have to call the "GSCLS" subprogram.
GSORB

1. Purpose

This subprogram receives report data.

2. Format

CALL GSORB ( [SN ] array, size, icount, itype)

Where,

- array: The array which receives data.
- size: The size of the array.
- icount: The integer variable of the amount of data received (when this data is numeric, the number of words is returned and when characters, the number of characters is returned.)

When the data overflows the array, the exact amount of data filling the array is stored, and a negative value is returned into "icount."

The rest can be picked up by calling this subprogram again.

- itype: The integer variable of the data type.
  - 0 : Integer
  - 1 : Real number
  - 2 : Characters
  - 3 : All report data has already been picked out.

3. General Description

(a) Before picking out the report data by this sub-

program, the user program has to make certain by the
"GSPRT" subprogram that the report data has been sent.

(b) When the "GSPRT" subprogram is called again without
picking out data by "GSORB" subprogram after making
certain by the "GSPRT" subprogram that the report data
has been sent, the report data will be missed.

(c) If the size of the array "array" presented is insuffi-
cient to store sent data (corresponding to GSPRT called
once), the value of data amount with a minus sign actu-
ally stored in "array" will be returned to "icount."

Then, the consecutive part of the previously picked up
data can be picked up by calling this subprogram again.
**Supplement**

**Data Transfer**

The relation of GSRPT, GSENQ, and GSORB is illustrated and the transfer of communication data is indicated as follows:

<table>
<thead>
<tr>
<th>UNIVAC 1100 Side (or AGS Side)</th>
<th>AGS Side (or UNIVAC 1100 Side)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: GSRPT (L)</td>
<td>A: GSRPT (L)</td>
</tr>
<tr>
<td>B: GSRPT (C)</td>
<td>B: GSRPT (C)</td>
</tr>
<tr>
<td>C: GSRPT (C)</td>
<td>C: GSRPT (C)</td>
</tr>
<tr>
<td>D: GSRPT (L)</td>
<td>D: GSRPT (L)</td>
</tr>
<tr>
<td>E: GSRPT (L)</td>
<td>E: GSRPT (L)</td>
</tr>
</tbody>
</table>

**NOTE:**
- L: Last Data
- C: Continued Data
- G: Get

**The Kinds of Errors and Error Processing**

(1) **AGS Side**

The following two kinds of errors exist on the AGS Side.

(a) **Status Error**

When the user's calling condition is bad, this shall be called status error and a positive error code is given. (See the error code table on the next page.

(b) **System Error**

The error caused by trouble in the system or that generated depending on the communication processing shall be a system error.

* **Status Error Processing**

The error code being returned to the user, the GS CALL statement causing the error is skipped. This appears from the user's viewpoint as if this GS subprogram was not called.

* **System Error Processing**

Following messages are outputted on the system console and the communication is terminated.
**III GSA SYSTEM ERROR ***
* ERROR ADDRESS = XXXXX

* Status Error Code Table (common to AGS and UNIVAC 1100 sides)

<table>
<thead>
<tr>
<th>ERROR</th>
<th>DESCRIPTION</th>
<th>ROUTINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>GS has been already initiated.</td>
<td>GSOIP</td>
</tr>
<tr>
<td>10</td>
<td>There are too many arguments.</td>
<td>All Routines</td>
</tr>
<tr>
<td>11</td>
<td>Arguments are insufficient.</td>
<td>All Routines</td>
</tr>
<tr>
<td>36</td>
<td>The specification of the amount of data is not appropriate. (The amount of data ≠ ϕ)</td>
<td>GSRPT</td>
</tr>
<tr>
<td>63</td>
<td>The specification of the condition code is not appropriate. (The condition code ≠ ϕ or 1)</td>
<td>GSRPT, GOENQ</td>
</tr>
<tr>
<td>71</td>
<td>The specification of the size of the array to pick out data is not appropriate. (The size = 0)</td>
<td>GSGBB</td>
</tr>
<tr>
<td>72</td>
<td>There is no data to pick out. (all data have been already picked out.)</td>
<td>GSGBB</td>
</tr>
<tr>
<td>74</td>
<td>The specification of the data type is not appropriate. (Data type ≠ ϕ, 1, or 2)</td>
<td>GSRPT</td>
</tr>
<tr>
<td>77</td>
<td>The report data sent previously was completed. (The last report data not sent.)</td>
<td>GSBNQ</td>
</tr>
</tbody>
</table>

* Other Error Messages
  (a) "GSOIP" is not called or resulted in an error, and any other GS subprogram was called. PLEASE CALL GSOIP

(b) An error depending on the communication processing has occurred during the execution of the "GSOIP" subprogram.

* GSOPH ERROR ... GS CLOSE *

(2) UNIVAC 1100 Side
The following three kinds of errors exist on UNIVAC 1100 side.

(a) **Status Error**
When the user call condition is bad, this shall be the status error and the positive error code is given. (See the error code list.)

(b) **System Error**
This is the error caused by some trouble in the system or the error generated depending on the communication processing, and a negative error code is given.

(c) **Contingency Error**
An illegal hardware instruction or guard mode, etc.
shall cause a contingency error and a negative error code, (-100) is given.

- **Status Error Processing**
The error code is returned to the user and the GS subprogram causing the error is skipped. This seems to the user as if this GS subprogram was not called.

- **System Error Processing**
The GS subprogram which caused the error is skipped and the "GSCLS" subprogram is called internally. (The user cannot call other subprograms than "GSOPN" hereafter.) Then, the error information in the following figure A is printed and the processing is continued.

- **Contingency Error Processing**
The GS program which caused the error is skipped, "GSCLS" instruction is called internally. (The user cannot call other subprograms than "GSOPN" hereafter.) Then, the error information in the following figure B is printed and the processing is continued.

![Diagram](image)

Figure A System Error Information

<table>
<thead>
<tr>
<th>BB</th>
<th>System Error Code (Decimal Number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSXXX</td>
<td>GS Subprogram Name</td>
</tr>
<tr>
<td>WWW</td>
<td>Walk Back Point (Octal Number)</td>
</tr>
</tbody>
</table>
**CONTINGENCY ERROR IN GSX**

Contingency Type:
- AX and AR Registers
- Contingency Words (2 words)
- Dump of User Working Area

Figure B: Contingency Error Information

Contingency Type:
- Illegal Operation
- Guard Mode
- Floating Point Overflow
- Floating Point Underflow
- Divide Overflow
- Error Mode Entry

*System Error Code Table*

<table>
<thead>
<tr>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1~ -63</td>
<td>The status error of IOE function</td>
</tr>
<tr>
<td>-64</td>
<td>The integer number overed 30 bits.</td>
</tr>
<tr>
<td>-67</td>
<td>An invalid message came from AGS side while GSOPN is executed.</td>
</tr>
<tr>
<td>-68</td>
<td>The address of the pool buffer is improper.</td>
</tr>
<tr>
<td>-69</td>
<td>An invalid message came from the AGS side while GSENQ is executed.</td>
</tr>
<tr>
<td>-70</td>
<td>Operation error of call queue stack</td>
</tr>
<tr>
<td>-71</td>
<td>Conflict of internal working parameters</td>
</tr>
<tr>
<td>-72</td>
<td>Initialization/termination error of IOE</td>
</tr>
<tr>
<td>-100</td>
<td>Contingency error</td>
</tr>
</tbody>
</table>
Other Error Messages

1. Any other subprogram was called without calling COOPN.

2. There is an error in COOPN call.

3. COOPN has not been called.

4. Syntax error exists in COOPN call.

5. Printing for System/Contingency Error Information.

6. Read lock in COOP: not graphics.

7. Processing may be required by A...

8. Subsequent call to COOPN...

ORIGINAL PAGE IS OF POOR QUALITY