GRiDTaskBuilder & GRiDTask II User's Guide

November 1985

2535 Garcia Avenue
P.O. Box 7535
Mountain View, CA 94039-7535
(415) 961-4800

Manual Name: GRiDTaskBuilder & GRiDTask II User's Guide
Order Number: 021250-43
Issue date: November 1985

Contents

GRiDTaskBuilder & GRiDTask II User's Guide

GRiDTaskBuilder and GRiDTask II are two programs that let you construct Task programs that are significantly more efficient than those developed for use with the original GRiDTask. Only minor changes are required to make most existing Task programs work with GRiDTask II. Refer to the GRiDTask User's Guide for information on the GRiDTask language. This document describes only the minor differences and special considerations that must be observed when writing Task programs for use with GRiDTaskBuilder and GRiDTask II. Operating instructions for the GRiDTaskBuilder program are also provided in this document.

An Overview of GRiDTaskBuilder This section provides an overview of GRiDTaskBuilder and highlights the differences between Task programs that are developed using GRiDTaskBuilder and those developed for use with the original GRiDTask.

Figure 1 illustrates the relationship between the original GRiDTask and Task program files.

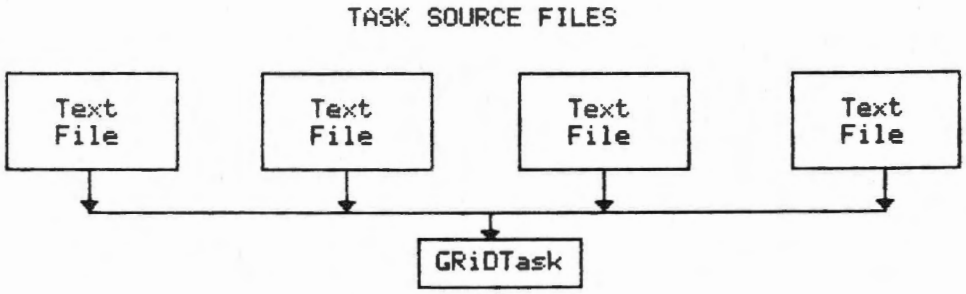


Figure 1. GRiDTask Program Files

Each Task program file consists of ASCII text comprising statements that are to be executed as part of the GRiDTask program. A Task program file has a Kind of ^Task^ and can be executed by selecting it from the File form. GRiDTask is then loaded into memory and begins reading the contents of a source file, one sector at a time, and executing the statements in the file. Procedures that are defined in a source file are kept in memory to speed execution. The program can consist of multiple source files: GRiDTask will access the additional files as directed by statements in the program.

Figure 2 illustrates the relationship between GRiDTask II, GRiDTaskBuilder, and Task II program files.

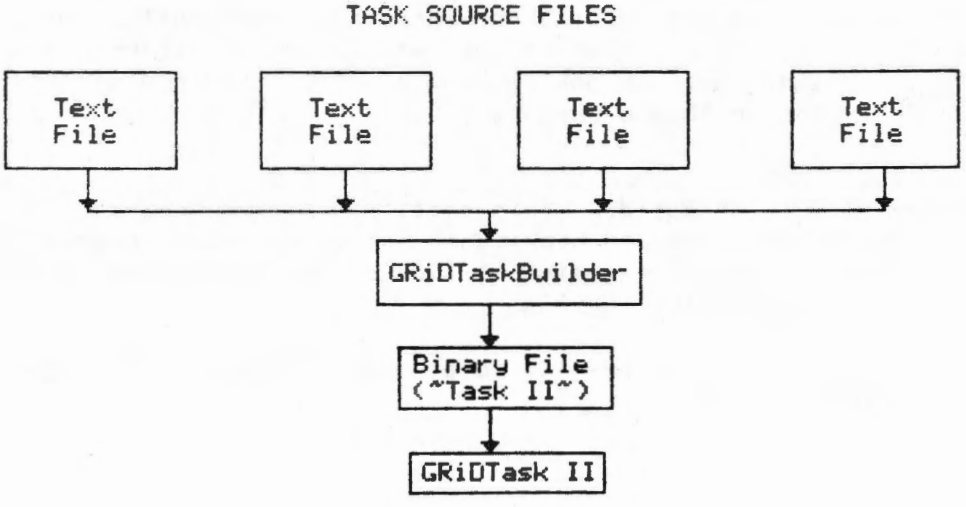


Figure 2. An Overview of the GRiDTaskBuilder Program Environment

The text files shown in this figure are the same as the Task source files shown in Figure 1: they contain the GRiDTask verbs and statements. There are minor differences (described later) between source files written for the original version of GRiDTask and those to be used in the Builder/Task II environment, but conceptually, they are the same. However, these text files are not executed directly by GRiDTask II. Instead, the text source files are processed by GRiDTaskBuilder, which produces a single binary file containing all of the pertinent data from the source files. This single binary file has a Kind of "Task II" and can be executed by selecting it from the File form. GRiDTask II is then loaded into memory, reads the binary file into memory, and executes the statements in the file.

The one disadvantage of the Builder/Task II approach is that it requires one additional step - processing the source files with GRiDTaskBuilder. The benefits provided by this approach, however, are significant and include the following:

- Smaller size. The binary file produced by GRiDTaskBuilder is significantly smaller than the Task source files.

- Faster execution. The smaller binary file produced by GRiDTaskBuilder is loaded entirely into memory thus reducing the number of file accesses required. Additionally, much of the source file processing that was performed "on the fly" by the original GRiDTask is performed ahead of time by GRiDTaskBuilder. This reduces the processing that must be performed by GRiDTask II and significantly speeds up execution.

- Greater program integrity. Since GRiDTask II uses only the binary file produced by GRiDTaskBuilder, the program source files need not be available at execution time. Since the binary file cannot be edited or changed, it provides protection against inadvertent or malicious modification of a GRiDTask program.

GRiDTaskBuilder Operation The paragraphs that follow describe how to invoke GRiDTaskBuilder to operate on text files and create executable Task II files.

Command Line Syntax

GRiDTaskBuilder is invoked from a command line using the following syntax:

```
GRiDTaskBuilder mainFile,[file2,file3, ...][TO outputFile],
[LIST listFile],[NOLIST],[LIBRARIES a,b,c, ...]
```

Each item in the command line entry is separated from the preceding item by a comma or by one or more spaces.

Items enclosed by square brackets are optional. Each of the items

and the default values (if any) are described below. NOTE: GRiDTaskBuilder assumes that all of the input files have a Kind of ^Text^. Therefore, when you specify the files, you need not append ^Text^ to the file titles.

- mainFile is the title of the text file that is the main GRiDTask module. Note that the first file (mainFile) in the list of input files must be the MAIN module of the program and that module must contain at least one statement that is outside of a procedure.

- file2,file3,... are the titles of additional, non-main modules that are to be part of the GRiDTask program. These modules cannot contain any statements that are outside of procedures.

- outputFile is the title that is to be assigned to the binary file produced by GRiDTaskBuilder. If you do not specify TO and an output file title, the output file is given the same title as the mainFile but with a Kind of ^Task II^.

- listFile is the title that is to be assigned to the list file produced by GRiDTaskBuilder. If you do not specify a list file title, the list file is given the same title as the mainFile but with a Kind of ^Lst^. If you do not want a list file, you must use the NOLIST control.

- LIBRARIES a,b,c,... specifies the names of any libraries that are to be used by the GRiDTask program. Note that the names must match those used with the INSTALL verb in your program and that you still must have an INSTALL statement in the program for each library.

Invoking GRiDTaskBuilder

To issue the command line that invokes GRiDTaskBuilder you create a command file that contains the command line and then use the Do program to run that file. The DO program lets you execute a prearranged sequence of commands contained in a command file of Kind ^Com^. The Do program reads the commands from the file and presents them one at a time to the command line interpreter for execution.

To create a command file, follow these steps:

1. Using GRiDWrite, create a file that contains the GRiDTaskBuilder command line.

2. Be sure to end the command line with a carriage return.

3. Save the file, specifying a Kind of ^Com^.

The file Sample^Com^ on your GRiDTaskBuilder diskette is an example of a command file used to invoke GRiDTaskBuilder.

The command file is executed by simply selecting it from the File form.

When you want to edit a command file using GRiDWrite, you cannot directly retrieve the file with the File form since this would automatically retrieve the Do program with the file instead of GRiDWrite. Instead, you should select GRiDWrite and then choose the "Exchange for another file" option on the Transfer menu. Fill out the File form with the name of your command file, and be sure to choose "Get new file only" for the "Next action" item. This will retrieve your command file without the Do program.

NOTE: GRiDTaskBuilder users who also have the GRiD Development Tools can use GRiDDevelop to invoke GRiDTaskBuilder. To do this you would define a LINK, TEST, or unique token such as BUILD that would contain the desired command line.

GRiDTaskBuilder Status Display and List File

GRiDTaskBuilder displays informational messages while it is processing files and a number of statistics after it has completed parsing the file(s). The same information that is displayed on the screen is also written to the list file (unless you have specified NOLIST) so that it can be examined later if necessary. An example of the information that is dislayed and written to the list file is shown below:

```
SCANNING FILE:  sample
PARSING FILE:  sample

Statistics
-----------------------------------------
Procedures.............................43
String variables.......................37
Real variables.........................21
Installed libraries....................0
Total lines read.......................228
Total lines parsed.....................168
Size of input file(s)..............2810
Size of output file.................1760
Errors.................................0
-----------------------------------------

Parse time:  3 minutes  8 seconds
```

If errors are encountered, then the error messages are written to the screen and to the list file. No output file will be created by GRiDTaskBuilder if any errors are encountered.

Converting Old GRiDTask Programs This section describes the changes that must be made to programs written for the original version of GRiDTask before they can be used with GRiDTaskBuilder and GRiDTask II. The changes required can be summarized as follows:

- Converting DO statements to procedures.
- Converting "loose" statements to procedures in non-main modules.
- Ensuring that INSTALL and WINDOWMOTION verbs use constants instead of variables or expressions.

The third item listed above is self-explanatory and should have little impact -- especially since it is not common practice to use anything but constants with the INSTALL and WINDOWMOTION verbs. The first two items require a bit more consideration, however, and are described in the paragraphs that follow.

Converting DO Statements to Procedures
DO statements cannot be used in programs processed by GRiDTaskBuilder. GRiDTaskBuilder will generate an error if it encounters a DO statement; program processing will continue but no output file will be produced. Therefore, all DO statements must be converted to procedures. For example, consider the following program segment that uses the DO statement:

```
Query$ = " PRINT ""What is your sign?"" "
DO Query$
```

This program segment could be converted to the following:

```
PROCEDURE Query
  PRINT "What is your sign?"
ENDP
```

and the procedure could then be invoked by the statement:

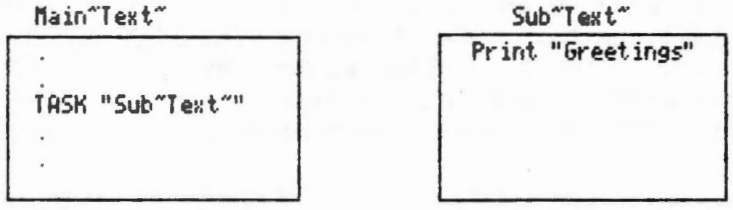
```
Query
```

Converting "Loose" Statements to Procedures
GRiDTaskBuilder requires that all non-main modules contain only procedures. Therefore, you must convert any "loose" statements (that is, statements that are not part of a procedure) to procedures.

Under the original GRiDTask, you could use the TASK verb to jump to the beginning of another file for execution and this second file could consist of any task statements. For example, in the following illustration, program control is passed from the Main module to the

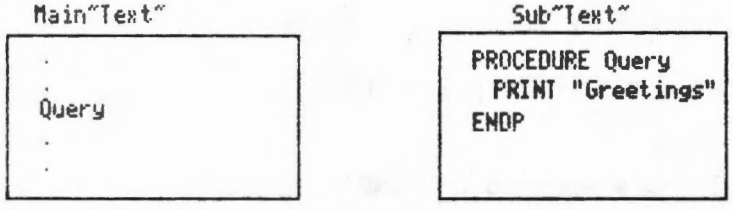
Sub module by the TASK statement, and the PRINT statement in Sub is executed.

OLD STYLE TASK PROGRAM



GRiDTask II, however, requires that all files except the main program file contain only procedures. For example, the PRINT statement in the Sub file illustrated above would have to be converted to a procedure. The Main program would then cause the PRINT statement to be executed by calling that procedure by name. This approach can be illustrated as follows:

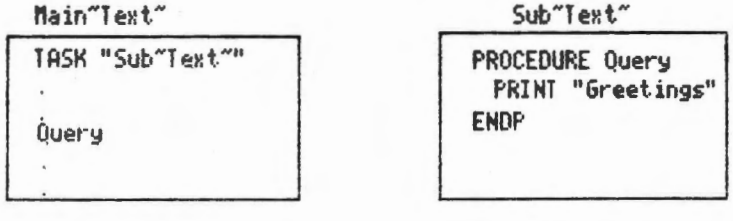
NEW STYLE TASK PROGRAM



No TASK statement is required: the procedure named Query in the Sub file is called by name in the Main file. Of course, the file Sub~Text~ must be one of the input files when you invoke GRiDTaskBuilder as described earlier.

GRiDTaskBuilder ignores any TASK statements it encounters in a file. This fact can be exploited to write programs that will work properly with both the original GRiDTask and GRiDTask II. This technique can be illustrated as follows:

TWO-WAY COMPATIBLE TASK PROGRAM



In this example, the loose statement in the Sub file has been converted to a procedure as required by GRiDTask II. The Query procedure is then called by name in the Main file. Thus far, this example is the same as the preceding one. There is one additional statement in the Main file, however. The TASK statement that we removed earlier has been restored in order to make the program compatible with the original GRiDTask. Since GRiDTaskBuilder ignores the TASK statement, the program remains compatible with GRiDTask II.

GRiDTask II Enhancement One new language enhancement has been added to GRiDTask II. Under GRiDTask II, the format of the IF/THEN/ELSE/ENDIF statement is more flexible. There is no longer a restriction that an IF statement be broken into multiple lines. All parts of the statement can be on a single line. For example, the following statements all have valid syntax.

```
IF TIME$ = "05/00/00 p.m." THEN STOP ELSE PRINT "More work to do"
ENDIF

IF TIME$ = "05/00/00 p.m."
THEN STOP
ELSE PRINT "More work to do" ENDIF

IF TIME$ = "05/00/00 p.m." THEN
  STOP
ELSE
  PRINT "More work to do" ENDIF
```

The "THEN" following the expression is optional, but will make your program more readable if you include it in IF statements that have multiple statements on one line. Programs written under the original GRiDTask will run under GRiDTask II without IF statement modification. However, this change was not implemented in the original version of GRiDTask, and programs written for that version must still have IF statements broken into multiple lines.