USING DDE TO COMMUNICATE WITH MICROSOFT PROJECT FROM OTHER APPLICATIONS

Microsoft Project for Windows has been designed so that you can control it using an external macro language. By writing macros which contain Dynamic Data Exchange (DDE) functions, you can issue commands to Microsoft Project and exchange information with it.

This file contains syntax descriptions and macro samples to help you access Microsoft Project information using DDE and the macro languages of other applications, such as Microsoft Excel, Microsoft Visual Basic or Microsoft Word.

There are several advantages to using DDE commands and an external macro language to control and communicate with Microsoft Project. You can:

- Use commands available in the external macro language to create dialog boxes and controls which provide a custom interface to Microsoft Project.

- Exchange information as needed between another application and Microsoft Project by running a macro, without linking the two documents. This provides a faster exchange of information when you need to move a lot of data between applications.

- Access categories of Microsoft Project information which are not available through conventional cutting and pasting or linking. While conventional DDE links are limited to the task and resource fields available in Microsoft Project tables, with macros you can directly access detailed resource assignment fields and some Project Information and Project Status fields.

- Control Microsoft Project from another application by sending it Microsoft Project macro commands included in a macro written with an external macro language.

In general, using an external macro language will give you more flexibility and control than Microsoft Project's macro language, which is used for simple scripting, looping, and branching of Microsoft Project commands. In addition, when you use external macros you can take advantage of the capabilities of two different applications. For example, you might calculate estimated task durations using Microsoft Excel, then use a Microsoft Excel macro to send these durations to Microsoft Project, instructing Microsoft Project to select and link the tasks, and send back the project finish date.

If all you need to do is establish simple DDE Links between your application and the fields in a Microsoft Project file, you do not need to use macros; you can simply copy the information in your application and use the Paste Link command to insert it into Microsoft Project. Or, you can copy information from a table in Microsoft Project, and link it into another application using that application's equivalent of the Paste Link command. This type of DDE linking is described in the "Transferring Information" topic in the *Microsoft Project User's Reference.*

This file contains basic information about the syntax you use to access Microsoft Project information when writing macros in external macro languages. It includes examples using Microsoft Excel's macro language, Microsoft Word's WordBasic, and Microsoft Visual Basic. More complex sample macro files written with these languages are also included with Microsoft Project. You can open these files if you have the appropriate application or language installed on your system. For more

information about the Microsoft Project macro language, refer to the file called COMMANDS.WRI.

For complete information about your macro language's commands, structure, and syntax, refer to your macro language documentation.

Establishing a DDE Conversation with Microsoft Project

Using macros, you can set up a DDE conversation between your application and Microsoft Project.

You control the conversation by issuing commands to Microsoft Project in the form of macro statements. Microsoft Project responds to the commands by performing requested actions, accepting data sent by the macro, or by sending back requested information.

To initiate a DDE conversation with Microsoft Project, you need to send macro commands that do the following:

1. Initiate a DDE conversation, specifying Microsoft Project as the application you want to communicate with. To identify Microsoft Project as the object of a DDE conversation, include its application name: WINPROJ.

2. Specify the "topic" of the conversation. There are two types of topics: "System," for communicating with Microsoft Project itself, or a reference to a specific Microsoft Project file.

The commands are usually arranged as:

<DDE Initiate Command> "Winproj" "*<topic>*"

For example, the following lines from macros establish communication with Microsoft Project at the System level:

Microsoft Excel:	=INITIATE("Winproj","System")
Microsoft Word WordBasic:	channel1 = DDEInitiate("Winproj","System")
Visual Basic:	Label1.LinkTopic = "Winproj|System" Label1.LinkMode = 2

When you initiate a DDE conversation using a System topic, Microsoft Project can respond to requests or execute commands with general system information such as application status or the content or position of the current selection in the currently active file.

When you initiate a DDE conversation using a specific file name as the topic, your macro gives you direct access to the information within that file; you can insert information into the fields of the project file, and request information from it without referring to the current selection in the file. The topic should include the full path and filename of the file, which must currently be open in Microsoft Project.

In other words, using the System topic communicates with Microsoft Project, while

using a filename as a topic communicates with a specific Microsoft Project file.

For example, the following lines from macros establish communication with the Microsoft Project file specified in the topic statement:

Microsoft Excel:	=INITIATE("Winproj","c:\win3\move.mpp")
Microsoft Word WordBasic:	channel1 = DDEInitiate("Winproj","c:\win3\move.mpp")
Visual Basic:	Label1.LinkTopic = "Winproj|c:\win3\move.mpp" Label1.LinkMode = 2

Note that you may want to open more than one channel in any given macro: for example, you may need access to both the general information supplied at the System level and specific information available on a channel opened to a particular file.

Once your macro has established a DDE communications channel, it can send additional commands to specify what you want Microsoft Project to do, and what "items" you want it to send or receive. "Items" are categories of information that the System can provide, or references to specific fields of information within the file identified in the topic statement. By including DDE commands paired with item references, your macros can:

- Use Request commands that ask Microsoft Project to send back specific information. You can request either an item that the System can provide, or an item from a specific file. For example, your macro might request the data currently selected in a project (a System item), or the finish date of a task (a file-specific item).

- Use Poke commands, which transfer information into Microsoft Project. When you poke information, it is as if the macro is typing information directly into Microsoft Project fields. Usually pokes are directed into specific files, but some System items can also accept poked information.

- Send DDE Execute commands: these are macro commands which allow you to control Microsoft Project as if you were choosing commands from its menus. Most of the Microsoft Project commands are available as equivalent macro commands. In addition, "action-equivalent" macro commands are available, enabling you to instruct Microsoft Project to execute common actions, such as promote or demote tasks. Each macro command consists of a version of the command name or name of the action, followed in some cases by required and optional arguments. For more information, see "Sending DDE Execute Commands" later in this file. The complete syntax of each macro command is described in the file COMMANDS.WRI, and is included in online Help in "Commands Used in Macros."

As a final step, every macro should include DDE Terminate statements, to stop the DDE conversation and close any DDE channels used during execution of the macro, thus freeing up memory and system resources.

So the general structure of a DDE Macro that communicates with Microsoft Project is as follows:

<DDE Initiate Command> "Winproj","*<System, or path and filename>*"
<DDE Request, poke, or execute commands> "*<System or file-specific items, or macro commands>*"
<DDE Terminate Command>

You can send pokes, requests, and macro commands to Microsoft Project on channels established both to the System or to a specific file.

Syntax examples for different macro languages are provided in the following sections of this file.

Using the System Topic and System Items

Here are two examples of simple DDE macros that communicate with Microsoft Project with the System topic.

The following Microsoft Excel macro initiates a DDE conversation with Microsoft Project at the System level, requests the field of data selected in the active project and uses the Formula statement to place it as a formula in the cell B3 in Sheet1. Begin by pasting this macro into a macro sheet starting at cell A1 so the cell references will be correct.

```
=INITIATE("WINPROJ","SYSTEM")
=FORMULA(REQUEST(A1,"ActiveData"),Sheet1!$B$3)
=TERMINATE(A1)
=RETURN()
```

An equivalent Microsoft Word for Windows macro gets the currently selected data in Microsoft Project, and displays it in the Microsoft Word Status Bar.

```
channel1 = DDEInitiate("winproj", "system")
data$ = DDERequest$(channel1, "ActiveData")
Print data$
DDETerminate channel1
```

In Microsoft Visual Basic, the structure would be:

```
Sub Command1_Click ()
  Label1.linktopic = "winproj|system"
  Label1.linkmode = 2
  Label1.linkitem = "ActiveData"
  Label1.linkrequest
End Sub
```

The following list describes the valid System items you can request from Microsoft Project and the information each returns. Returned information is the response sent back to the requesting application by Microsoft Project; how you display it depends on what commands you include in your macro. See the "Examples Section" later in this document for some examples of how to handle Microsoft Project data returned to a macro.

SysItems
Returns a list of all the available System items (as described in this list).

Topics
Returns a list of all the topics currently available for DDE: this is a list of the Microsoft

Project files that are currently open, and "System."

Status
Returns "Ready" or "Busy" depending on the current state of Microsoft Project. Microsoft Project may be busy if it is currently calculating or printing a project, or waiting for input while displaying a dialog box.

Formats
Returns a list all the data formats Microsoft Project supports: Biff3, Biff, RTF, and Text.

Selection
Returns a reference that defines the current selection in Microsoft Project: for example, T((1,2),name) indicates that the name field for two tasks is currently selected. If the selection does not contain any tasks or resources, Microsoft Project refuses the request. If there is a multiple selection in Microsoft Project, only the first block of the selection is returned. For complete explanation of the syntax, see "Syntax Examples for Each Data Type" later in this document.

SelectedData
Returns the data within the current selection as an array (rows and columns). The records are tab delimited, with a carriage return/line feed (CR/LF) included at the end of each record, if more than one record is returned. You can poke information into the SelectedData item, as well as request it.

SelectionPos
Returns the coordinates of the current selection as a series of numbers in the following order: x-coordinate of the upper-left corner of the selection, y-coordinate of the upper-left corner of the selection, width of selection in columns, height of selection in rows.

For example, if you had six cells in the upper left corner of a task sheet selected, SelectionPos would return

1,1,2,3

to indicate that the selection begins at coordinates 1,1 (row 1, column 1), and consists of 2 columns by 3 rows.

ActiveCell
Returns a reference that defines the active task or resource and field in the active window.

ActiveData
Returns the data in the active cell. This item is similar to SelectedData but includes only one field of information.

Note that you can poke information into the ActiveData item, as well as request it.

NumLines
Returns the total number of lines in the active view, up to and including the last non-blank task or resource. If a filter has been applied to a view, only the visible lines are counted.

LastError
Returns the last error state active when a macro sent data to Microsoft Project. Returned as an array consisting of:

Topic	Item	Error Code

where Error Code is 0 = no error
1 = syntax error
2 = value error
3 = clipboard format error
4 = memory error

For example, if you tried to poke information into a request-only item, such as NumLines, requesting LastError would return: "System","NumLines", 1.

You can request the following System items to find out the current state of Microsoft Project. By requesting these items in your macro and storing them as variables, you can use them later in the macro to restore Microsoft Project to its original state.

ViewFile
Returns the full path and filename of the current Microsoft Project view file.

CalendarFile
Returns the full path and filename of the current Microsoft Project calendar file.

ActiveView
Returns the name of the active view. If a dual-pane view is being displayed, only the name of the active pane is returned.

ActiveTable
Returns the name of the table applied to the view.

ActiveFilter
Returns the name of the filter applied to the view.

ActiveProject
Returns the full path and filename of the active project.

Using a Specific Microsoft Project File as a Topic

This section explains how to set up a macro to exchange information with a particular Microsoft Project file, including how to transfer information into specific fields in a Microsoft Project file, and how to specify the location of data that you want to extract from Microsoft Project files.

To access information in a specific Microsoft Project file, you need to first establish a DDE communication channel with a DDE Initiate command that specifies the topic as the path and name of a project file.

In Microsoft Excel you would include the line: =INITIATE("WINPROJ","*path and filename*")

For example: =INITIATE("WINPROJ","c:\win3\project\aero.mpp")

In Microsoft WordBasic, you would include the line:
  channum = DDEInitiate ("winproj","*path and filename*")

For example: channel1 = DDEInitiate("winproj","c:\win3\move.mpp")

In Microsoft Visual Basic, you would include the lines:
  object.linktopic = "winproj|*path and filename*"
  object.linkmode = 1 or 2

For example:
  label1.linktopic = "winproj|c:\win3\project\aero.mpp"
  label1.linkmode = 2

Then subsequent lines in the macro can refer back to this channel to request information from the file, or poke information into it.

When you refer to information in a Microsoft Project file, you list the type of information you are requesting or sending, followed by lists of unique task or resource ID numbers and lists of fields. For example, you might request task information, specifying names and durations; or you might send resource information, specifying rates for wages. The general format is:

  Information Type((*list of Unique ID numbers*),(*list of fields*))

The Unique ID number is a number Microsoft Project assigns sequentially to each task or resource as it is created in a project. This Unique ID number is distinct from the Task or Resource ID number, which may change for a given task or resource when tasks or resources are inserted or deleted, or if you renumber during sorting.

It is important that your macros include the Unique ID number when referring to fields related to a specific task or resource. To find out the Unique ID number for a task or resource, create a table in Microsoft Project that includes a column for Unique ID. Or, you can set up your macro to request the Unique ID for a task or resource for subsequent use within the macro. For example, you could set up a request using the wildcard symbol (*) to request this information for all tasks, such as the following, which requests the name and Unique ID number for each task in the current project file:

  T(*,(Name,Unique ID))

The following example shows how to refer to task information for specific tasks within a Microsoft Project file:

  T((1,2,5,7),(Name,Cost))

T = Task Information (to indicate fields that contain information about tasks).

1,2,5,7 = The Unique ID numbers of tasks you want to refer to in the project.

Name, Cost = The information contained in the name and cost fields for the tasks specified by the Unique ID numbers.

In other words, you need to tell Microsoft Project what type of data you want sent or received, identify the tasks or resource it applies to, and identify precisely what fields of information are being exchanged.

Syntax Guidelines

The following general guidelines and rules will help you in structuring macros that access the information in Microsoft Project files:

- All the letters and names are case-insensitive.

- You can arrange the Unique ID numbers and fields in whatever order you wish; information will be sent or returned in the order in which it appears within the macro statements.

- List separator characters: the default list separator character, usually a comma or semicolon, can be changed using the Preferences command on the Options menu in Microsoft Project. If you are using a different list separator character in Microsoft Project, substitute that character in the portions of the macro that refer to Microsoft Project fields. For example, if the list separator character had been changed to a semicolon, you would use syntax like this:

  =DDEPoke(*channum*,"T(1;name)")

  If you want to assure that you are using the correct list separator character, include a DDE execute statement in your macro that explicitly sets the list separator character. For example:

  OptionsPreferences .ListSeparator=[,]

- Wildcards: in most cases you can substitute the wildcard character (*) for lists of task or resource Unique ID numbers, in order to request information about all tasks or resources in the active project. Wildcard examples are included where appropriate in the syntax examples in the next section.

Information Types and Syntax Examples

This section describes the syntax you use to refer to the fields of information contained in Microsoft Project files. When writing macros to exchange information with Microsoft Project, you follow the DDE Request or DDE Poke commands with references to the information type, Unique ID numbers, and fields you want to request information from or poke information into.

There are five different Microsoft Project information types:

- Project Information
- Task Information
- Resource Information
- Resource Assignment Information
- Unique Resource Assignment Information

Each information type is designated by a different identifying letter. You can also use equivalent identifying numbers, listed below, instead of the identifying letters. Use

the numbers instead of the letters if your macro must work with versions of Microsoft Project that have been translated into different languages.

You can refer to fields by field name, as shown in the examples in this section, or you can use the equivalent field numbers, which match the field numbers used in the Microsoft Project Exchange (MPX) file format. For example, 1 = Name, 40 = Duration, etc. The field number equivalents for Microsoft Project fields are included in the files MPXFILE.WRI and FIELDS.WRI. These numbers are also used when working with versions of Microsoft Project that have been translated into different languages.

Type P or 0 - Project Information (Global)

P = Project information, or "global" information which applies to an entire project, not particular task or resource fields. Can also be specified by the equivalent information type number, 0.

The fields correspond to the fields found in the Project Information and Project Status dialog boxes in Microsoft Project. Only number entries are allowed for these fields, so use the field numbers listed below:

1 = Project
2 = Company
3 = Manager
4 = Project Calendar
5 = Project Start Date
6 = Project Finish Date
7 = Schedule From
8 = Current Date
9 = Notes
10 = Cost
11 = Planned Cost
12 = Actual Cost
13 = Work
14 = Planned Work
15 = Actual Work
16 = % Work Complete
17 = Duration
18 = Planned Duration
19 = Actual Duration
20 = % Complete
21 = Planned Start
22 = Planned Finish
23 = Actual Start
24 = Actual Finish
25 = Start Variance
26 = Finish Variance
27 = Number of Tasks
28 = Number of Resources

For example, P(1,3,6) or 0(1,3,6) refers to the Project Name, the Manager, and the Project Finish Date.

Pokes into fields 10 through 17 will be ignored, because the fields are always calculated by Microsoft Project.

Type T or 1 - Task Information

T = Task information, used for references to task fields. Task fields are displayed in task tables in Microsoft Project. Can also be specified by the number 1.

List of IDs = Unique ID number for the tasks being referenced
List of Fields = Names (or numbers) of Task fields being referenced, as in:

T((*list of IDs*),(*list of fields*))

This example requests the name and duration for the tasks with Unique ID numbers 33 and 35.

T((33,35),(Name,Duration)) or
1((33,35),(Name,Duration)) or
1((33,35),(1,40))

The value is returned as an array. For example, if the two tasks are called "site survey" and "excavation" and their durations are 10d and 20d respectively, Microsoft Project would return:

{"site survey",10;"excavation",20}

How this array would appear depends on how your macro is set up to display returned values.

Using wildcards, you could request the name and duration of all tasks in the project:

T(*,(Name, Duration))

Type R or 2 - Resource Information

R = Resource information for references to resource fields. Resource fields are displayed in resource tables in Microsoft Project. Can also be specified by the number 2.

List of IDs = Unique ID numbers for the resources being referenced
List of Fields = Names (or numbers) of Resource fields being referenced, as in:

R((*list of IDs*),(*list of fields*))

This example request the standard resource rate for the resources with Unique IDs 21 through 25:

R((21,22,23,24,25),(Standard Rate)) or
2((21,22,23,24,25),(Standard Rate)) or
2((21,22,23,24,25),(42))

The value returned would be a an array, pairing resource names with standard rate values.

Using wildcards, you could, for example, request the name and cost for each resource in the project:

R(*,(Name, Cost))

Type A or 3 - Resource Assignment Information

A = Resource assignment information and refers to the resource assignment fields displayed in the fields at the bottom of the Task Form or Resource Form in Microsoft Project. These fields contain detailed information about each resource assigned to a task. Can also be specified by the number 3.

List of IDs = pairs of numbers that define what resource is assigned to what task (Task Unique ID number followed by Resource Unique ID number).

List of Fields = the field numbers that represent the resource assignment fields. Only number entries are allowed for these fields, so use the field numbers listed below:

1 = Unique Assignment ID (defined in "Type U or 4" section below)
2 = Task Unique ID
3 = Resource Unique ID
4 = Units
5 = Work
6 = Planned Work
7 = Actual Work
8 = Overtime Work
9 = Cost
10 = Planned Cost
11 = Actual Cost
12 = Scheduled Start
13 = Scheduled Finish
14 = Delay

Pokes into fields 1, 2, 3, 12, and 13 will be ignored because they are always calculated by Microsoft Project.

The syntax is:

A((*list of ID number pairs*),(*list of field numbers*))

Examples:

A(((3,3),(3,4),(3,5)),(5,6))

Refers to the Work and Planned Work fields (fields 5 and 6), for resources with Resource Unique IDs 3, 4, and 5, which are assigned to a task with Task Unique ID = 3.

A((1,*),(5,6))

Refers to the Work and Planned Work fields (fields 5 and 6), for all the resources

assigned to the task with Task Unique ID=1.

A(((*,2),(*,3)),5)

Refers to the Work field (field 5), for all tasks that have the resources with Resource Unique IDs = 2 and 3 assigned to them.

Type U or 4 - Resource Assignment Information, Referencing Unique ID

U = Unique Resource Assignment information. In addition to task and resource Unique ID numbers, Microsoft Project assigns a Resource Assignment Unique ID number each time a resource is assigned to a task. You may want to access this number to identify a specific instance in which a resource has been assigned to a task, for example if the resource has been assigned more than once to the same task.

These numbers are not displayed in Microsoft Project, so you must request them in your macro if you need to refer to them.

To get the unique resource assignment ID numbers for a task, you need to send a request to Microsoft Project, using information type A (resource assignment information), and asking for field 1 (Assignment Unique ID). For example:

A((1,3),1)

means "show me all the Unique Resource Assignment ID numbers that have been assigned to the pairing of the task with Unique ID = 1 and the resource with Unique ID = 3." If you get more than one number back, it indicates that the resource is assigned to the same task more than once.

Unique Resource Assignment ID numbers can also be specified by the number 4.

List of IDs = Task Unique ID, paired with Unique Resource Assignment ID.

List of Fields = Only number entries are allowed for these fields, so use the field numbers listed below:

1 = Unique Assignment ID
2 = Task Unique ID
3 = Resource Unique ID
4 = Units
5 = Work
6 = Planned Work
7 = Actual Work
8 = Overtime Work
9 = Cost
10 = Planned Cost
11 = Actual Cost
12 = Scheduled Start
13 = Scheduled Finish
14 = Delay

Pokes into fields 1, 2, 3, 12, and 13 will be ignored, because the fields are always

calculated by Microsoft Project.

The syntax is:

U((*list of ID numbers paired with Unique Resource Assignment ID*),(*list of field numbers*))

Examples:

U((3,10),5)

Refers to the Work field (field 5), for the task with Task Unique ID = 3, which has Unique Resource Assignment 10.

U(((3,5),(4,7)),(5,6))

Refers to Task 3, Assignment 5, Task 4, Assignment 7, fields 5 and 6 (Work and Planned Work) (3,5,4,7 are all Unique references).

Combining System and File Topics in the Same Macro

You may want your macros to open more than one DDE channel to communicate with Microsoft Project: one channel to open a specific file you want to access information in, and a System channel so that you can request System items. You'll need to set up the macro so that System-related requests refer back to the System channel, while the file specific lines of the macro refer back to the channel opened to a specific file. For more information, see the "Examples Section" later in this file.

Sending DDE Execute Commands

In addition to Pokes and Requests, you can send commands directly to Microsoft Project. Most of the commands on the Microsoft Project menus are available in a macro command format, so that you can operate Microsoft Project from within a macro. In addition, "action-equivalent" commands are available to allow macros to perform common operations that are not specifically related to commands, such as selecting tasks, collapsing and expanding outlines, and so on. For more information, see the file COMMANDS.WRI which describes the Microsoft Project macro language.

Execute commands can be sent on DDE communication channels established either with the System or with a specific Microsoft Project file, but we recommend establishing a DDE communication channel with the System whenever execute commands are sent to Microsoft Project. The commands always act on the active project.

A given command may be followed by required and/or optional arguments which correspond with the settings or options usually available when you are running Microsoft Project directly. Other commands do not require any arguments.

When you send macro commands to Microsoft Project, no dialog boxes are displayed for most commands. If you want Microsoft Project to display dialog boxes during the execution of a macro, you should include SendKeys commands in the macro.

Arguments in command functions are separated by the list separator character,

usually a comma or semicolon, as specified in the Preferences dialog box, or by spaces. If the value within a field contains the list separator character or a space, it must be surrounded by quotation marks. Microsoft Project supports the use of either brackets or quotation marks, because you may find that some applications work better with one or the other.

For commands with arguments, the general syntax is:

command name **.***argument name=argument value*

Note that a space follows the command name and precedes the period before the argument name.

Following are some examples of commands with and without arguments. The Examples section at the end of this file also includes the use of DDE Execute commands within macros.

The following examples show macros that set the start date for a project, select and link the tasks within it, and then request the project finish date.

Note that in the following examples, Project1 (an untitled Project file) is used to establish a DDE communications channel with a file. Once a project has been saved, you must use the full path and filename in order to establish a DDE communications channel with the file.

Microsoft Excel (paste into a macro sheet starting at cell A1):

```
=INITIATE("Winproj", "System")
=INITIATE("Winproj","Project1")
=EXECUTE(A2,"OptionsProjectInfo .Start=9/22/92")
=EXECUTE(A2, "View [Task Sheet]")
=EXECUTE(A2, "SelectAll")
=EXECUTE(A2,"EditLinkTasks")
=FORMULA(REQUEST(A2,"P(6)"),(Sheet1!$A$1))
=TERMINATE(A2)
=TERMINATE(A1)
=RETURN()
```

Microsoft Word WordBasic:

```
channel1 = DDEInitiate("Winproj","System")
channel2 = DDEInitiate("Winproj","Project1")
DDEExecute channel2, "OptionsProjectInfo .Start=9/22/92"
DDEExecute channel2, "View [Task Sheet]"
DDEExecute channel2, "SelectAll"
DDEExecute channel2, "EditLinkTasks"
Finish$ = DDERequest$ (channel2,"P(6)")
Print Finish$
DDETerminate channel2
DDETerminate channel1
```

Microsoft Visual Basic:

```
label1.linktopic = "winproj|system"
label1.linktopic = "winproj|project1"
label1.linkmode = 2
label1.linkexecute "OptionsProjectInfo .Start=9/22/92"
label1.linkexecute "View [Task Sheet]"
```

```
label1.linkexecute "SelectAll"
label1.linkexecute "EditLinkTasks"
label1.linkitem = "P(6)"
label1.linkrequest
label1.linkmode = 0
```

Examples Section

This section contains short examples of macros written in the macro languages of Microsoft Excel, Microsoft Word, and Microsoft Visual Basic. You can run these by copying and pasting them into their respective application or language.

Longer examples of macros written in each of these languages have been included with Microsoft Project, and can be found in the directory where you installed Microsoft Project. You can launch these macros using the AppExecute function in the Microsoft Project macro language, or run them from their respective applications. You can also add them to the Microsoft Project Macro menu or assign them to buttons on the tool bar.

Microsoft Excel Macro Examples

This section contains examples of short Microsoft Excel macros which place information into or extract information from Microsoft Project. You can copy and adapt them for use in you own macros, using the syntax described in this document. For the cell references in these macros to be accurate, you should begin by pasting them into a macro sheet starting at cell A1. Also, you should turn off R1C1 references (using the Workspace function).

Note that in these examples, Project1 (an untitled Project file) is used to establish a DDE communications channel with a file. Once a project has been saved, you must use the full path and filename in order to establish a DDE communications channel with the file.

This macro requests all tasks in Project1, and puts the data into worksheet Sheet1:

```
=INITIATE("WINPROJ","System")
=INITIATE("WINPROJ","Project1")
=DEFINE.NAME("AllTasks",REQUEST(A2,"T(*,Name)"))
=FORMULA.ARRAY(AllTasks,"Sheet1!R1C1:R"&ROWS(AllTasks)&"c")    Must enter with Ctrl+Shift+Enter
=TERMINATE(A2)
=TERMINATE(A1)
=RETURN()
```

This macro also requests all tasks in Project1, but uses a named range instead of a specific references. For example, if the link name MyName is defined as T(*,Name) in the File Links dialog for Project1, the above macro can be written as follows:

```
=INITIATE("WINPROJ","System")
=INITIATE("WINPROJ","Project1")
=DEFINE.NAME("AllTasks",REQUEST(A2,"MyName"))   Define MyName as T(*,Name) in File Links dialog of
WinProj
=FORMULA.ARRAY(AllTasks,"Sheet1!R1C1:R"&ROWS(AllTasks)&"c")       Must enter with Ctrl+Shift+Enter
=TERMINATE(A2)
=TERMINATE(A1)
=RETURN()
```

This macro pokes a single duration into a project file called TASKDUR.MPP. Note that

TASKDUR.MPP must contain a task with Unique ID = 1.

```
30d
=INITIATE("WINPROJ","System")
=INITIATE("WINPROJ","c:\win3\taskdur.mpp")
=POKE(A3,"T(1,Duration)",A1)                A1 contains the data (i.e., 30d)
=TERMINATE(A3)
=TERMINATE(A2)
=RETURN()
```

This macro pokes an array into Microsoft Project. Note that Project1 must contain tasks with Unique IDs 4 and 5.

```
=INITIATE("WINPROJ","System")
=INITIATE("WINPROJ","Project1")
=POKE(A2,"T((4,5),(Name,Duration))",A7:B8)          The array is in A7:B8
=TERMINATE(A2)
=TERMINATE(A1)
=RETURN()
task4  2d
task5  5d
```

This macro sends DDE Execute commands to Microsoft Project:

```
=INITIATE("WINPROJ","SYSTEM")
=EXECUTE(A1,"AppMaximize")
=EXECUTE(A1,"View [PERT Chart]")
=TERMINATE(A1)
=RETURN()
```

This macro requests all critical tasks and places their names, along with the name of the current project, onto Sheet1 (which must be active when this macro is run):

```
=INITIATE("WINPROJ","system")   Initiate SYSTEM chan
=REQUEST(A1,"ActiveProject")
=INITIATE("WINPROJ",A2)            Initiate FILENAME chan
=REQUEST(A3,"P(1)")     Get and store the project name
=IF(LEN(A4)=0,A2,A4)    or if none, the filename
=EXECUTE(A1,"View [Task Sheet]")
=EXECUTE(A1,"OutlineExpandAll")
=EXECUTE(A1,"Filter [Critical]")
=EXECUTE(A1,"SelectAll")
=REQUEST(A1,"Selection")          Select the critical tasks
=DEFINE.NAME("CritTasks",REQUEST(A3,LEFT(A10,SEARCH(")",(",A10)+2)&"Name))"))    Get the
(selected) critical tasks
=FORMULA("The following tasks are critical in "&A5&":",Sheet1!A1)
=FORMULA.ARRAY(!CritTasks,"Sheet1!R2C1:R"&ROWS(!CritTasks)+1&"c1")      Must enter with
Ctrl+Shift+Enter
=RETURN()
```

Microsoft Word for Windows Macro Examples

This section contains examples of short Microsoft Word WordBasic macros which place information into or extract information from Microsoft Project. You can copy and adapt them for use in your own macros, using the syntax described in this document and the guidelines outlined above.

This macro requests the Name and Duration of all tasks in the active project. First, it gets the active project, using the System topic, then opens a conversation on that topic. Note that the project must contain tasks with Unique IDs 1, 3, and 5.

```
Sub MAIN
        channel1 = DDEInitiate("winproj", "system")
        channel2 = DDEInitiate("winproj", DDERequest$(channel1, "ActiveProject"))
        AllTasks$ = DDERequest$(channel2, "T(*,(Name,Duration))")
        Insert AllTasks$
        DDETerminate channel1
        DDETerminate channel2
End Sub
```

This macro pokes a previously defined Microsoft Word glossary entry called "ProjectNotes" into the Notes field for the project, using the same type of DDE connection as the one above. Note that the macro will only paste one line of information into the Notes field. If there are tab characters or a return character in the Microsoft Word text, only the characters preceding the first tab or carriage return will be pasted in:

```
Sub MAIN
        channel1 = DDEInitiate("winproj", "system")
        channel2 = DDEInitiate("winproj", DDERequest$(channel1, "ActiveProject"))
        DDEPoke channel2, "P(9)", GetGlossary$("ProjectNotes")
        DDETerminate channel1
End Sub
```

This macro pokes the current selection in a Microsoft Word for Windows document into the Name and Duration fields for tasks 1, 3, and 5, using the same type of DDE connection as the one above:

```
Sub MAIN
        channel1 = DDEInitiate("winproj", "system")
        channel2 = DDEInitiate("winproj", DDERequest$(channel1, "ActiveProject"))
        DDEPoke channel2, "T((1,3,5),(Name,Duration))", Selection$()
        DDETerminate channel1
        DDETerminate channel2
End Sub
```

This macro sends DDE Execute commands to Microsoft Project. First it displays a Task Sheet and filters for critical tasks, and then it displays the list of critical tasks in Microsoft Word:

```
Sub MAIN
  channel1 = DDEInitiate("winproj", "system")
  channel2 = DDEInitiate("winproj", DDERequest$(channel1, "ActiveProject"))
  Project$ = DDERequest$(channel2, "P(1)")
  If Len(Project$) <= 2 Then
        Project$ = DDERequest$(channel1, "ActiveProject")
  End If
 CurrentView$ = DDERequest$(channel1, "ActiveView")
  DDEExecute channel1, "View [Task Sheet]"
  DDEExecute channel1, "OutlineExpandAll"
  DDEExecute channel1, "Filter [Critical]"
  DDEExecute channel1, "SelectAll"
  Crit$ = DDERequest$(channel2, "T(" + Mid$(DDERequest$(channel1, "Selection"), 3, InStr(1,
DDERequest$(channel1, "Selection"), "),(") - 2) + ",Name)")
  Insert "The following tasks are critical in " + Project$ + Chr$(13) + Chr$(11)
  Insert Crit$
  DDEExecute channel1, "View [" + CurrentView$ + "]"
  DDETerminateAll
End Sub
```

Microsoft Visual Basic Macro Examples

This macro is an example of how Microsoft Visual Basic can request information from Microsoft Project. The macro gets the Project, Company, and Manager name from Options Project Info for the active project, and displays it in a label. To try this code, create a form with a label control and a command button control, and paste the code into the form.

```
Sub Command1_Click ()
  ' Get general project information from Microsoft Project
   Label1.LinkTopic = "WINPROJ|System"         ' DDE topic for ActiveProject
   Label1.LinkMode = 2                        ' Use a cold (one-time) link
   Label1.LinkItem = "ActiveProject"           ' Request the path of the
   Label1.LinkRequest                        '   currently active project
   Label1.LinkMode = 0            ' Changing topic

   Label1.LinkTopic = "WINPROJ|" + Label1.Caption
   Label1.LinkMode = 2                        ' Use a cold (one-time) link
   Label1.LinkItem = "0(1,2,3)"                ' Request the Project, Company, and
   Label1.LinkRequest                        ' Manager for this project
   Label1.LinkMode = 0                        ' Done linking

   ' Replace tabs with carriage returns
   Capt$ = Label1.Caption                     ' Convert for string routines
   While (InStr(1, Capt$, Chr$(9)) <> 0)
      Mid$(Capt$, InStr(1, Capt$, Chr$(9)), 1) = Chr$(13)
   Wend
   Label1.Caption = Capt$                     ' Convert back

End Sub
```

As an example of a macro which sends a DDE Execute command, this example automatically adjusts the width of all columns in the current table to fit the widest entry in the columns, using Microsoft Project's Best Fit option. To try this code, create a form with a label control and a command button control and paste the code into the form.

```
Sub Command1_Click ()

  Label1.LinkTopic = "WINPROJ|System"     ' DDE topic for executes
  Label1.LinkMode = 2                     ' Use a cold (one-time) link

  On Error Resume Next                    ' Handle errors manually

  Label1.LinkExecute "SelectBeginning"    ' Select top left cell
  Counter = 0                           ' Initialize counter
  While Counter < 10                     ' Do ten iterations
     Counter = Counter + 1                ' Increment counter
     Label1.LinkExecute "ColumnBestFit"  ' Best fit this column
     Label1.LinkExecute "SelectCellRight"  ' Go to next column
  Wend
  Label1.LinkMode = 0                     ' Done linking

End Sub
```

This example demonstrates how to poke information into Microsoft Project using DDE from Visual Basic. It allows you to enter the same note for each selected task or resource. To try this code, create a form with a label control, a text control, and a command button control. Paste the code into the form.

```
   Sub Command1_Click ()
```

```
   ' Request information from active project
   Label1.LinkTopic = "WINPROJ|System"         ' DDE topic for ActiveProject
   Label1.LinkMode = 2                         ' Use a cold (one-time) link
   Label1.LinkItem = "ActiveProject"           ' Request the path of the
   Label1.LinkRequest                          '   currently active project
   Text1.LinkTopic = "WINPROJ|" + Label1.Caption ' DDE topic for the project-
   Text1.LinkMode = 2                          '   specific info
   Label1.LinkItem = "Selection"               ' Request the selection
   Label1.LinkRequest

   ' Parse out the list of IDs from the selection, and prepare to DDE poke
   ' into their notes fields
   Text1.LinkItem = Left$(Label1.Caption, 1) + "(" + IDListOf$(Label1.Caption) + ",Notes)"

   ' Loop to multiply number of times text occurs, so note shows up in each
   ' task or resource in Microsoft Project
   StartPos = InStr(Text1.LinkItem, ","): Temp$ = Text1.Text + Chr$(13)
   Do While (StartPos <> 1)
      Temp$ = Temp$ + Text1.Text + Chr$(13)
      StartPos = InStr(StartPos, Text1.LinkItem, ",") + 1
   Loop
   Text1.Text = Temp$
   Text1.LinkPoke                              ' Poke the information
   Label1.LinkMode = 0
   Text1.LinkMode = 0

End Sub

Function IDListOf$ (ByVal A$)    ' ByVal to accept VB control captions
   ' Takes a DDE selection description string from Microsoft Project, like
   ' "T((1,2,4),(Name,Duration))" and returns the ID or list of IDs in the
   ' description as a string, like "(1,2,4)"
   CommaPos = InStr(A$, "),") + 1
   If CommaPos = 1 Then CommaPos = InStr(A$, ",")
   IDListOf = Mid$(A$, 3, CommaPos - 3)
End Function
```