_NBSTARTMACRO	{WindowHide}{_save_app_settings}	Hide the Budgeteer window
	{Application.Macro.Macro_Redraw "Both"}	Suppress screen updates during macro execu
	{Application.Title "The Budgeteer"}	Set application title
	{Application.SpeedBar "BUDGTRAK.BAR"}	Set application SpeedBar
	{Application.Display "None,No,No,Yes,A:A1..B:B2"}	Hide Input Line and SpeedBar
	{Application.Enable_Inspection "No"}	Disable right-clicking
	{SETMENUBAR menu_block}	Set menu bar to application menu
	{SETOBJECTPROPERTY "/Print.Depend_On","No,Yes,No,N	Specify when the Print menu is available
	{BRANCH _init_screen}	Display expenses
_NBEXITMACRO	{Let _nbexit,1}	Flag that _NBEXITMACRO is being executed
_clear_app	{IF file_saved=0}{_confirm_quit}	If the active budget file isn't saved, confirm the
	{Application.Macro.Macro_Redraw +app_macro}	Re-enable screen updates
	{Application.Title "Quattro Pro for Windows"}	Reset application title
	{Application.SpeedBar +app_speedbar}	Restore SpeedBar layout
	{Application.Display +app_display}	Restore Display settings
	{Application.Enable_Inspection "Yes"}	Re-enable Object Inspector menus
	{SETMENUBAR}	Restore standard menu system
	{If _nbexit}{Let _nbexit,0}{Quit}	Resets _nbexit variable and doesn't execute F
	{FileCloseAll 0}	Close all files opened by Budgteer
_nbexit		
app_macro	None	
app_display	Standard,Yes,Yes,Yes,A..B:A1..B2	
app_speedbar	PRODUCTY.BAR	
_save_app_settings	{LET app_macro,@COMMAND("Application.Macro.Macro_Redraw")}	
	{LET app_display,@COMMAND("Application.Display")}	
	{LET app_speedbar,@COMMAND("Application.SpeedBar")}	

ıtion

ɛ quit

ileCloseAll in _NBEXITMACRO

```
MENU Budget_Bar
&Budget
        &New...          MACRO [BUDGTRAK.WB1]_new_budget          Create a ne
        &Open...         MACRO [BUDGTRAK.WB1]_open_budget         Open an ex
        &Save            MACRO [BUDGTRAK.WB1]_save_budget         Save the ac
        Save &As...      MACRO [BUDGTRAK.WB1]_save_budget_as      Save the ac
        Add &Expense...  MACRO [BUDGTRAK.WB1]_add_expense         Add an exp
        &Quit            MACRO [BUDGTRAK.WB1]_clear_app           Exit the Bu
&Print
        &All             ON clicked DOMACRO {_print_budget "All"}         Print all ex
        &Current         ON clicked DOMACRO {_print_budget "Current"}     Print all rec
        &Daily           ON clicked DOMACRO {_print_budget "Daily"}       Print Daily
        &Weekly          ON clicked DOMACRO {_print_budget "Weekly"}      Print Week
        &Monthly         ON clicked DOMACRO {_print_budget "Monthly"}     Print Montl
        &Bi-Monthly      ON clicked DOMACRO {_print_budget "Bi_Monthl      Print Bi-M
        &Semi-Annually   ON clicked DOMACRO {_print_budget "Semi_Annu     Print Semi-
        A&nnually        ON clicked DOMACRO {_print_budget "Annually"}    Print Annu
&Statistics MACRO [BUDG Display statistical information about records in the record windo
```

w, empty b"Yes,Yes,No,No,No,No"
isting budg"Yes,Yes,No,No,No,No"
tive budget"No,Yes,No,No,No,No"
tive budget"No,Yes,No,No,No,No"
ense to a d"No,Yes,No,No,No,No"
dgeteer "Yes,Yes,No,No,No,No"

pense datab"No,Yes,No,No,No,No"
ords in the "No,Yes,No,No,No,No"
expense da"No,Yes,No,No,No,No"
ly expense "No,Yes,No,No,No,No"
hly expense"No,Yes,No,No,No,No"
onthly expe"No,Yes,No,No,No,No"
Annual exp"No,Yes,No,No,No,No"
al expense ("No,Yes,No,No,No,No"
"No,Yes,No,No,No,No"

Developer Notes

Click a SpeedButton for more information on:

Macro layout	The layout of macros in this file.
Database structure	How expense databases are stored.
Searching	Searching expense databases.
Graphing	Graphing records in the record window.
Selecting blocks	Developer tips on selecting blocks and tracking block coordinates.
Checking for things	Developer tips on checking whether a block name, graph, or named page exists.

help_input

Enter the cost of each expense below its name. You can change the entry date as well, but press the Date key (Ctrl+Shift+D) before typing the date. When you're done, press Enter by itself or Esc to save the record.

<< Press a key to continue >>

help_stats

You can use the arrow keys to scroll through stats for each expense. When you're done, press Enter by itself or Esc.

<< Press a key to continue >>

help_files

You can use file controls to specify the name of a budget file to open or to specify the name to save the active budget file under. Enter the name under File Name, and then choose OK. To cancel the operation, choose Cancel. Choose New to create a new, empty budget file.

<< Press a key to continue >>

help_filerr

A problem has occurred. Make sure a valid file name and directory is specified.

<< Press a key to continue >>

help_invalidf

The file you have requested is not a valid budget file.

<< Press a key to continue >>

help_noadd

This budget database currently has no expense columns, so you can't add a record to it. To create expense columns, choose Budget|Add Expense.

<< Press a key to continue >>

help_layout1

The Budgeteer notebook is structured to make finding and studying macros easier. Each page of the notebook contains macros common to one Budgeteer task. For example, Print_macros contains macros that print expense data.

On each macro page, column A lists the named blocks in the Budgeteer notebook; each named block is listed to the left of the cell (or block) it references. This is handy for re-creating the block names quickly using Block|Names|Labels. The named blocks in most macros only refer to one cell. If a block name in this notebook starts with a space, it refers to a block larger than one cell. If a block name begins with an underscore (_), it's the name of a macro the Budgteer runs.

<< Press a key to continue >>

help_layout2

Column B in each macro page contains the macro commands and variables used by the Budgeteer. If a cell is colored gray, it contains a formula used by a macro. Some of these formulas are in the macros; this lets the macro adapt to the data in the active budget file.

Column F contains a brief description of what each line of the macro does. At the start of many macros is a boxed message describing what the macro does. Use this documentation to study the Budgeteer more closely.

<< Press a key to continue >>

_help_layout

{MESSAGE help_layout1,10,5,0}

{MESSAGE help_layout2,10,5,0}
{EditGoto A1}{RIGHT}

help_data1

The expense databases created by the Budgeteer are stored in a separate notebook, which is created by the macro _new_budget. Each page of this notebook contains the data for a specific expense frequency (Daily through Annually). The data is stored on the page in a block starting at cell C1. This block is set as the database block when the Budgeteer searches a database. (For more information on database blocks, see the User's Guide). The actual coordinates of the database block are stored in A2, to make it easier for the Budgeteer to set up a search quickly.

<< Press a key to continue >>

help_data2

The list of expenses for a database is stored in column B, starting at row 2. This list is used by the Add Expense dialog box. The coordinates of this list are stored in A5, so that the dialog box can change quickly.

<< Press a key to continue >>

_help_data

{MESSAGE help_data1,10,5,0}
{MESSAGE help_data2,10,5,0}
{EditGoto A1}{RIGHT}

help_search1

The Budgeteer uses the command equivalents of Data|Query to search for records and copy them into the record window. See the User's Guide for more information on setting up a database and query in Quattro Pro.

The query always uses the criteria table stored in the named block criteria_table (in the Budgeteer notebook). The macros _refresh_view and _new_query put formulas into this criteria table whenever a setting is changed on the Budgeteer SpeedBar.

<< Press a key to continue >>

help_search2

The coordinates of the database block are stored on the notebook pages Daily through Annually; cell A2 on each page has the coordinates for the data stored on that page. These coordinates are passed as an argument to {Query.Database_Block} whenever the View control on the Budgeteer SpeedBar changes to a new setting.

Once the database block is set, {Query.Assign_Names} is used to create block names that the Stats page needs to calculate statistical information. The expense names are also copied into the first row of the record window, and then that row is specified as the output block (the row's coordinates are passed as an argument to the command equivalent {Query.Output_Block}). Then copying records into the record window is done by {Query.Extract}.

<< Press a key to continue >>

_help_search

```
{MESSAGE help_search1,10,5,0}
{MESSAGE help_search2,10,5,0}
{EditGoto A1}{RIGHT}
```

help_graph1

The macro in the Budgeteer that graphs data in the record window actually creates a macro that builds the graph. This constructed macro (named _build_graph) is broken down into three parts:

1. The first part of the macro, starting at the named cell _build_graph, creates the graph and displays it in a graph window for editing. This part of the macro never changes.

2. The second part of the macro, starting at the named cell _build_series, is created "on the fly" by the macro _graph_view (the user runs _graph_view by clicking Graph on the Budgeteer SpeedBar). The macro _graph_view measures the data in the record window, and creates a list of {Series.Data_Range} commands, starting at _build_series. This part of the macro defines the series of the graph.

<< Press a key to continue >>

help_graph2

3. The third part of the macro, starting just after the last series command, creates the series defined in part 2, displays the graph, asks the user if they want to print it, and returns to the record window. This part of the macro is stored in the named block finishing_mac; _graph_view copies the macro below the series commands it created for part two of the macro.

<< Press a key to continue >>

_help_graph

{MESSAGE help_graph1,10,5,0}
{MESSAGE help_graph2,10,5,0}
{EditGoto A1}{RIGHT}

help_select1
+

Keeping track of a block whose coordinates change regularly requires a little planning. It's important to make the upper left corner of the block a fixed location. This provides a starting point that you can rely on. The database blocks in the Budgeteer all start at the cell address C1.

You can use the formula @PROPERTY("Active_Block.Selection") to get the coordinates of the selected block. Since this information is returned as a label, you'll also use the function @@ to convert it into an address that Quattro Pro formulas can understand. The function @@ isn't needed in macros; any macro command argument that expects a block will understand the coordinates if they're entered as a string. See Building Spreadsheet Applications for more information on these functions and the argument type location.

<< Press a key to continue >>

help_select2
+

If you'll frequently need to know the width of a block (in columns), set up a row in that block that doesn't contain any blank cells. In the Budgeteer, the first row of the database blocks never contain a blank cell. This way you can move to the first cell of the row and use {SHIFT+END}{SHIFT+RIGHT} to select the entire row, regardless of its actual width. Once this row is selected, you can use the formula
@COLS(@@(@PROPERTY("Active_Block.Selection"))
to return the width of the block.

If you'll frequently need to know the height of a block (in rows), set up a column in that block that doesn't contain any blank cells. In the Budgeteer, the first column of the database blocks (the Date column) never contain a blank cell. This way you can move to the first cell of the column and use {SHIFT+END}{SHIFT+DOWN} to select the entire column, regardless of it's actual height. Once this column is selected, you can use the formula

@ROWS(@@(@PROPERTY("Active_Block.Selection"))
to return the height of the block.

<< Press a key to continue >>

help_select3
+

To select the whole block consistently, combine these two techniques: set up a column and a row in the block that contain no blank cells, preferably the first row and column. Selecting the block breaks down into the following steps:

1. Goto the upper-left corner of the block.

2. Use {END}{DOWN}. This moves the selector to the last row of the block.

3. Use {SHIFT+END}{SHIFT+UP}. This selects the first column of the block and moves the selector back to the upper-left corner of the block.

4. Use {SHIFT+END}{SHIFT+RIGHT}. This selects the first row of the block and all the columns in it.

Now you can manipulate the block using the formulas described earlier. For an example of this, see the macro _update_data_block in this notebook.

You can use the following macro to select all the data on the active page: {HOME}{SHIFT+END}{SHIFT+HOME}.

<< Press a key to continue >>

_help_select

```
{MESSAGE help_select1,10,5,0}
{MESSAGE help_select2,10,5,0}
{MESSAGE help_select3,10,5,0}
{EditGoto A1}{RIGHT}
```

help_check1

The function @ISERR is useful for checking whether a certain object exists. @ISERR takes one argument, a formula, and returns 1 if the formula returns ERR, 0 otherwise. You can use this function in conjunction with @PROPERTY to check the existence of an object: pick a property that's unique to the object you're trying to find (this ensures that objects of the same name but a different type aren't found), and then use @PROPERTY to get the setting of that property. If @PROPERTY returns ERR, the object doesn't exist. To prevent ERR from stopping the macro, pass it as an argument to @ISERR. Then @ISERR returns 0 when the

object exists, 1 when it doesn't.

<< Press a key to continue >>

help_check2
+

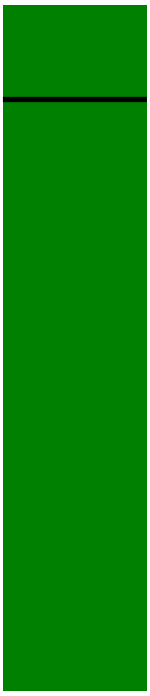
For example, suppose you want to know if the graph named Expense exists in the active notebook. First, pick a unique graph property: the Aspect_Ratio property is only available in graphs. Second, create the @PROPERTY formula that checks for that property setting: @PROPERTY("Expense.Aspect_Ratio"). Third, plug the formula into @ISERR: @ISERR(@PROPERTY("Expense.Aspect_Ratio")). This formula can be used in an {IF} command to see if the object exists.

You can use a similiar technique to find block names and page names: use @@ in an @ISERR function. For blocks, pass the block name as an argument to @@. For example, @ISERR(@@("database_block")) returns 1 if the named block database_block doesn't exist. For named pages, pass the address of a cell on the page. For example, @ISERR(@@("Daily:A1")) returns 1 if the page Daily doesn't exist.

<< Press a key to continue >>

_help_check

```
{MESSAGE help_check1,10,5,0}
{MESSAGE help_check2,10,5,0}
{EditGoto A1}{RIGHT}
```



Error_Check	0	Used by _add_expense to dete
New_Expense		Dialog box setting: the name o
Frequency	Daily	Dialog box setting: the expens
SpdBar_Freq	#NAME?	Setting of View on the Budgete
Current_Items	#REF!	Block derived from SpeedBar s
Current_Freq	#NAME?	Formula to change dashes to u

_add_expense	{LET New_Expense, ""}	Make initial expense setting bla
	{_get_bar_values}	Fetch latest SpeedBar settings
	{LET Frequency, @PROPERTY("[BUDGTRAK.BAR]View.Va	Set default frequency
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]AddExpense:t	Set initial list
	{DODIALOG "[BUDGTRAK.WB1]AddExpense",Error_Check,	Display Add Expense dialog bo
	{IF New_Expense=""}{BRANCH _exit_add}	If expense is blank, end this m
	{IF Error_Check = 0}{BRANCH _exit_add}	Trap if the user cancels the dia
	{BRANCH _add_field}	Run macro that adds expense
_exit_add	{RETURN}	End this macro

_add_field

The macro _add_field is called by _add_expense after the user specifies the name of an expense category to add. The name of the category is stored in new_expense. The name of the expense database to add the field to is stored in temp_freq. Here's the flow of the macro:

1. Goto to page containing the expense database.
2. Select the first non blank cell to the right of the label Date (in C1).
3. Enter the new expense name in that cell.
4. Call the subroutine _update_data_block to update the database block and item list.
5. If the field is in the database being viewed, run a new query.

temp_freq	Daily	Used by _add_field to determir

_add_field	{_get_bar_values}	Fetch latest SpeedBar settings
	{EditGoto +[BUDGTRAK.WB1]temp_freq&":C1"}	Activate the page in the active
	{IF @CELL("type",[]C(1)R(0))="l"}{END}{RIGHT}	Check the column to the right o
	{RIGHT}	Move to the cell where the new
	{LET []C(0)R(0),+New_Expense}	Store the new expense catego
	{LET update_freq,+temp_freq}{_update_data_block}	Update the database block and
	{IF (+"[]"&Frequency)=SpdBar_Freq}{_new_query}	If the new expense category w
	{_restore_view}	Return to the record window

_add_record

The macro _add_record is called by clicking the Add button on the Budgeteer SpeedBar. It adds a record to the expense database being viewed. Here's the flow of the macro:

1. Goto to page containing the expense database.
2. Copy the field names from that page to the notebook page Input.
3. Activate the page Input and protect all the cells on it.
4. Format row 2 to show the data correctly (after the user enters it).
5. Unprotect row 2 so RestrictInput can be used with it.
6. Enter INPUT mode, and ask the user to enter a record.
7. Goto to page containing the expense database.
8. Select the first cell in column D that's blank.
9. Copy data from Input to the current row
10. Call the subroutine _update_data_block to update the database block and item list.
11. Run a new query to reflect the new database entry.

_add_record	{_get_bar_values}	Fetch latest SpeedBar settings
	{_disable_ui "Yes"}	Disable the SpeedBar and mer
	{BLANK []Input:A1..IV2}	Erase any category names left
	{EditGoto +[BUDGTRAK.WB1]Current_Freq&":C1"}	Activate the page in the active
	{IF @CELL("type",[]D1)="b"}{_restore_view}{MESSAGE help	If cell D1 of the active page is l
	{BlockCopy C(0)R(0)..C(253)R(1),Input:A1}	Copy the expense category na
	{BLANK []Input:A2..IV2}	Erase any numerical data left c
	{SETOBJECTPROPERTY "Input:B2..IV2.Numeric_Format","(	Format the second row of the i
	{EditGoto Input:A2}	Activate the notebook page Inp
	{SETOBJECTPROPERTY "Input:A1..IV8192.Protection","Pro	Protect all cells on the page. T
	{COLUMNWIDTH []Input:A1..IV8192,1,2,2}	Auto-size the columns so that t
	{SHIFT+UP}{SHIFT+END}{SHIFT+RIGHT}{SHIFT+DOWN}	Select the cells the user will inp
	{SETPROPERTY "Protection","Unprotect"}	Unprotect the cells the user ca
	{LET []Input:A2,@TODAY}	Store the current date on the fc
	{SETOBJECTPROPERTY "Input:A2.Numeric_Format","Long	Format the default date as Lon
	{COLUMNWIDTH []Input:A2,1,2,2}	Make sure column A is wide er
	{MESSAGE help_input,10,7,0}	Displays instructions on how to
	{Application.Display.Show_InputLine "Yes"}	Displays the input line (hidden
	{RestrictInput.Enter Input:A1..IV2}	Enter INPUT mode. While in IN
	{PAUSEMACRO}	Wait for the user to press Ente
	{Application.Display.Show_InputLine "No"}	Hide the input line again
	{IF @CELL("type",[]Input:A2)<>"v"}{LET []Input:A2,+@TODA	If the user inadvertantly erased
	{EditGoto +[BUDGTRAK.WB1]Current_Freq&":C1"}	Activate the page in the active
	{IF @CELL("type",[]C(0)R(1))="v"}{END}{DOWN}	If the cell below Date (in C1) is
	{DOWN}	Move below the last record in t
	{BlockCopy Input:A2..IS2,C(0)R(0)}	Copy the user data from the inp
	{SETPROPERTY "Numeric_Format",@PROPERTY("[]C(0)R(	Set the numeric format of the c
	{LET update_freq,+current_freq}{_update_data_block}	Update the database block to r

	{PUTBLOCK +view_page&":Database_Block",[BUDGTRAK.V	Copy the coordinates of the ne
	{Query.Database_Block @@(+[BUDGTRAK.WB1]data_block	Set Quattro Pro's database blo
	{_refresh_view}	Return to the record window ar

_update_data_block

The macro _update_data_block is called by _add_field and _add_record after a new expense or record has been added to the database. The name of the database is stored in update_freq. Two pieces of information exist on each database page that help the Budgeteer do queries and add expenses. Database_Block (A2) is the actual database block used with {Query.Extract} to perform searches and copy records into the record window. Item_Block is the address of a list of expense names. This list is used in _add_expense while the user views the Add Expense dialog box to show the expense names that already exist in the database. _update_data_block makes sure these items stay current. Here's the flow of the macro:

1. Goto to page containing the expense database.
2. Select the expense names on the page.
3. Copy the expense names into column B starting at B2 (using BlockTranspose).
4. Select the names just copied in column B.
5. Calculate the coordinates of this block and store it in Item_Block.
6. Select the expense names and records in the database.
7. Calculate the coordinates of this block and store it in Database_Block.
8. Set the variable file_saved to 0 to indicate that unsaved changes exist.

update_freq	[]Weekly	Used by _update_data_block t
copy_cell	Weekly:B2..Weekly:B2	Used by _update_data_block t
_update_data_block	{_get_bar_values}	Fetch current SpeedBar setting
	{EditGoto +[BUDGTRAK.WB1]update_freq&":C1"}	Activate the page in the active
	{IF @CELL("type",[]C(0)R(1))="v"}{END}{DOWN}	If the database contains any re
	{SHIFT+END}{SHIFT+UP}	Select the first column of the d
	{IF @CELL("type",[]D1)="l"}{SHIFT+END}{SHIFT+RIGHT}	If any categories exist in the da
	{LET copy_cell,@PROPERTY("Active_Block.Selection")}	Store the coordinates of the se
	{BlockCopy [BUDGTRAK.WB1]copy_cell,+[BUDGTRAK.WB1	Copy the new coordinates to th
	{EditGoto C1}	Select the first cell of the datab
	{BlockTranspose C(1)R(0)..C(253)R(0),C(-1)R(1)}	Copy the expense category na
	{EditGoto B2}	Select the first cell of the list cr
	{IF @CELL("type",[]C(0)R(1))="l"}{SHIFT+END}{SHIFT+DOW	If more than one record exists,
	{LET copy_cell,@PROPERTY("Active_Block.Selection")}	Store the coordinates of the se
	{BlockCopy [BUDGTRAK.WB1]copy_cell,+[BUDGTRAK.WB1	Copy the new coordinates to th
	{LET file_saved,0}	Specify that until this file is sav

_disable_ui

The macro _disable_ui is called by various macros to disable or enable the Budgteer SpeedBar and menus. Which operation is performed is determined by ui_setting, which is set to Yes to disable the UI, or No to enable the UI. Here's the flow of the macro:

1. Disable/Enable the SpeedBar by setting its Disabled property to the setting stored in ui_setting.
2. Change the Depend On property of the menu items. This prevents them from unprotecting themselves as the user edits cells, activates new windows, and so on.
3. Disable/Enable the menus by setting their Grayed property to the setting stored in ui_setting.
4. If the UI is to be enabled, change the Depend On property settings back to their defaults; then Quattro Pro automatically ungreys the menus.

ui_setting	No	Used by _disable_ui to specify
_disable_ui	{DEFINE ui_setting:string}	Sets up the arguments the ma
	{SETOBJECTPROPERTY "[BUDGTRAK.BAR].Disabled",+[B	Disables/Enables the SpeedBa
	{SETOBJECTPROPERTY "/Budget.Depend_On","No,No,Yes	Changes the areas where Bud
	{SETOBJECTPROPERTY "/Print.Depend_On","No,No,Yes,Y	Changes the areas where Prin
	{SETOBJECTPROPERTY "/Statistics.Depend_On","No,No,Y	Changes the areas where Stat
	{SETOBJECTPROPERTY "/Budget.Grayed",+[BUDGTRAK.W	Disables/Enables the Budget n
	{SETOBJECTPROPERTY "/Print.Grayed",+[BUDGTRAK.WB	Disables/Enables the Print me
	{SETOBJECTPROPERTY "/Statistics.Grayed",+[BUDGTRAK	Disables/Enables the Statistics
	{IF [BUDGTRAK.WB1]ui_setting="Yes"}{RETURN}	If the macro is supposed to dis
	{SETOBJECTPROPERTY "/Budget.Depend_On","Yes,Yes,N	If the macro is supposed to ena
	{SETOBJECTPROPERTY "/Print.Depend_On","No,Yes,No,N	If the macro is supposed to ena
	{SETOBJECTPROPERTY "/Statistics.Depend_On","No,Yes,I	If the macro is supposed to ena

ermine how the Add Expense dialog box was closed
f the new expense to add
e database to add a new expense to

eer SpeedBar
setting
underscores. For example, Bi-Monthly becomes Bi_Monthly

ank
:

ox
acro
log box
to budget

ne the name of the expense database to add a new expense column to

:
notebook that contains the expense database specified by temp_freq
of the date field to see if any expense categories have already been added. If so, select all the expense category names
v expense category name will be stored
ry name in the active cell
l list of expense categories to reflect the new category
as added to the database being viewed, set up a new query, since the information has changed

nus while adding a record
over from the last Add operation
notebook that contains the expense database specified by current_freq
blank (which indicates that no expense categories have been added to the database) display an error message and return
mes to the notebook page Input, which is the form the user will input the new record into
over from the last Add operation
nput form as currency, so that when the user enters a value, it displays in currency format. Notice B1 isn't included here, si
ut, which contains the input form the user will fill in
his ensures that only the cells unprotected later in the macro are accessible to the user
the expense names aren't cropped
ut data into, so that they can be unprotected
n input data into. Used by RestrictInput later in the macro
rm under the date column. This way the user doesn't have to enter a date, and the form defaults to the current date
g International (MM/DD/YY)
nough to show the date
use the form, and how to complete the add operation
previously by the macro \0) so that the user can see and edit entries as they're typed
NPUT mode, only cells in the block that are unprotected are accessible to the user
r or Esc. These user actions "approve" the new record

the date listed in the date column, place the current date there again, since the date is critical to querying macros in the a
notebook that contains the expense database specified by current_freq
n't a value, then records already exist in the database and the cell selector should move to the last record in the database
he database
put form (on page Input) to the active row
lata to the same numeric format as the previous record
eflect the new record

w database block from the page containing the expense database to the variable data_block, which is used to perform que
ck to new coordinates. The variable data_block contains a cell address that indicates where the new coordinates are (not t
nd refresh the record display

o determine which expense database needs updating
o store the new database block and the new list of expense category names. This information is copied onto the notebook

gs
notebook that contains the expense database specified by update_freq
cords, move to the last record in the database
atabase. Note the active cell is now at the top of the database (C1)
atabase, select them. The new database block is now selected
lected block in the variable copy_cell
ne cell A2 on the active page. This information is used by query macros
ase (the top of the Date column)
mes to column B starting at B2. This information is used by _add_expense
eated by the last command
select all the records
lected block in the variable copy_cell
ne cell A5 on the active page. This information is used by query macros
ed, the user should be warned that unsaved changes exist

whether the UI should be disabled or enabled

cro accepts
ar
get is available. This ensures that Budget isn't accidentally enabled when it shouldn't be
t is available. This ensures that Print isn't accidentally enabled when it shouldn't be
istics is available. This ensures that Statistics isn't accidentally enabled when it shouldn't be
menu
nu
command
able the UI, the macro is finished, and the macro is stopped
able the UI, this command resets the areas in which the Budget menu is available
able the UI, this command resets the areas in which the Print menu is available
able the UI, this command resets the areas in which the Statistics command is available

to the record window, stopping the Add operation

nce that's where the date is entered

plication

ries

he coordinates themselves)

page containing the expense database

_new_query

The macro _new_query is called when the user selects a new choice from the View control on the Budgeteer SpeedBar, or an operation is performed that affects the expense names that will be shown in the record window. It searches the database and copies records that meet the SpeedBar criteria into the record window. It also sets up the statistics displayed on the Stats page. Here's the flow of the macro:

1. Activate the record window and clear it of all data.
2. Get the latest SpeedBar settings.
3. Activate the page Input and protect all the cells on it.
4. Calculate the coordinates of the database block to search.
5. Set the database block to the property coordinates.
6. Create field names based on the new coordinates.
7. Set and specify the criteria table to search with, using SpeedBar settings.
8. Copy the expense category names into the first row of the record window.
9. Set the output block to the first row of the record window.
10. Dress up the record window.
11. Set up the Stats page to display the current statistical information.
12. Call _refresh_view, which copies records into the record window.

criteria_table	Date	Date	Date	This is the criteria table used to determine wh
criteria				
	crit_1	crit_2	crit_3	The named blocks crit_1 through crit_3 are u
view_page	[]Monthly			Used by _new_query to determine the expen
current_day	#NAME?			The current setting of Day on the SpeedBar
current_month	#NAME?			The current setting of Month on the SpeedBa
current_year	#NAME?			The current setting of Year on the SpeedBar
data_block	#REF!			The coordinates of the database block to que
_new_query	{EditGoto Expense_View:A1}			Activate the record window
	{SHIFT+END}{SHIFT+HOME}			Select all data in the record window
	{ClearContents 1}			Clear the contents of the record window
	{EditGoto Expense_View:A1}			De-select the record window
	{_get_bar_values}			Fetch current SpeedBar settings
	{LET view_page,current_freq}			Copy the name of the expense database beir
	{PUTBLOCK "@@(+[BUDGTRAK.WB1]view_page&"":A2"")",			Copy the coordinates of the new database bl
	{Query.Database_Block +[BUDGTRAK.WB1]data_block}			Set Quattro Pro's database block to new coo
	{Query.Assign_Names}			Create field names (used in the criteria table)
	{Query.Criteria_Table [BUDGTRAK.WB1]criteria_table}			Set the criteria table to the block criteria_table

	{BLANK @CELL("ThreeDAddress",crit_1)&".."&@CELL("Thre	Erase the contents of the criteria table
	{IF Current_Day<>"All"}{PUTBLOCK "@CHOOSE(@MOD(+[	If the Day control on the SpeedBar isn't set to
	{IF Current_Month<>"All"}{PUTBLOCK "@CHOOSE(@MOD(	If the Month control on the SpeedBar isn't set
	{IF Current_Year<>"All"}{PUTBLOCK "(@YEAR([]DATE)+19(	If the Year control on the SpeedBar isn't set t
	{EditGoto +[BUDGTRAK.WB1]Current_Freq&":C1"}	Activate the page in the active notebook that
	{IF @CELL("type",[]C(1)R(0))="l"}{SHIFT+END}{SHIFT+RIGH	If any categories exist in the database, select
	{BlockCopy @PROPERTY("Active_Block.Selection"),Expens	Copy the expense category names to the first
	{EditGoto Expense_View:A1}	Activate the record window
	{IF @CELL("type",[]C(1)R(0))="l"}{SHIFT+END}{SHIFT+RIGH	If any expense category names exist, select t
	{SETPROPERTY "Font.Bold","Yes"}	Set the names to a bolder typeface
	{SETPROPERTY "Alignment","Center"}	Center the names
	{COLUMNWIDTH @PROPERTY("Active_Block.Selection"),1	Auto-size the columns so that the expense na
	{BLANK []Stats:B1..IV6}	Clear out any old data on the notebook page
	{BlockCopy Expense_View:B1..IV1,Stats:B1}	Copy the expense names to the first row of S
	{Query.Output_Block @PROPERTY("Active_Block.Selection'	Set the output block to the first row of the rec
	{EditGoto Stats:B2}	Activate the notebook page that contains stat
	{_add_stat "SUM","Currency,2"}	Add a formula to the Stats page that totals th
	{_add_stat "AVG","Currency,2"}	Add a formula to the Stats page that returns t
	{_add_stat "MAX","Currency,2"}	Add a formula to the Stats page that returns t
	{_add_stat "MIN","Currency,2"}	Add a formula to the Stats page that returns t
	{COLUMNWIDTH []Stats:B1..IV5,1,2,2}	Auto-size the columns so that the expense na
	{EditGoto Expense_View:A1}	Activate the record window
	{SETOBJECTPROPERTY "Expense_View:A1..A8192.Numer	Set the numeric format of the first column to c
	{COLUMNWIDTH []Expense_View:A1..IV8192,1,2,2}	Auto-size the columns so that the expense na
	{_refresh_view}	Return to the record window and refresh the r

_refresh_view

The macro _refresh_view is called when the user selects a new choice from the Day, Month, or Year control on the Budgeteer SpeedBar. It copies the records that meet the criteria specified in the SpeedBar into the record window. Here's the flow of the macro:

1. Get the latest SpeedBar settings.
2. Erase the entries in the current criteria table (the named block criteria_table).
3. Set up the criteria table based on the latest settings.
4. Copy the records into the record window.

_refresh_view	{_get_bar_values}	Fetch the current SpeedBar settings
	{EditGoto Expense_View:A1}	Activate the record window
	{BLANK @CELL("ThreeDAddress",crit_1)&".."&@CELL("Thre	Erase the contents of the criteria table
	{IF Current_Day<>"All"}{PUTBLOCK "@CHOOSE(@MOD(+[	If the Day control on the SpeedBar isn't set to
	{IF Current_Month<>"All"}{PUTBLOCK "@CHOOSE(@MOD(	If the Month control on the SpeedBar isn't set
	{IF Current_Year<>"All"}{PUTBLOCK "(@YEAR([]DATE)+19(	If the Year control on the SpeedBar isn't set t
	{ONERROR _do_nothing}	This ensures that if no records match the crite
	{Query.Extract}	Copies records that meet the criteria specifie
	{RECALCCOL []Stats:A1..B8192}	Refreshes the formulas on the notebook page

	{COLUMNWIDTH []Expense_View:A1..IV8192,1,2,2}	Auto-size the columns so that the expense na
	{_restore_view}	Return to the record window

_add_stat

The macro _add_stat is called by _new_query whenever a new database @function needs to be added to the Stats page (which displays statistical information about the records displayed in the record window). The named block stat_formula contains a text formula that results in the statistical formula to enter.

stat_formula	0	Used by _add_stat to specify a formula to typ
operation	MIN	Used by _add_stat to specify the database @
num_format	Currency,2	Used by _add_stat to specify the numeric for
_add_stat	{DEFINE operation:string, num_format:string}	Sets up the arguments the macro accepts
	{_get_bar_values}	Fetch the current SpeedBar settings
	#REF!	When a budget file is active, this text formula
	{PUTBLOCK +[BUDGTRAK.WB1]stat_formula}	Copies the formula returned by the formula st
	{SETPROPERTY "Numeric_Format",+[BUDGTRAK.WB1]nur	Sets the format of the active block to the num
	{DOWN}	Moves to the next row. Used when adding mu

_get_bar_values

The macro _get_bar_values is called whenever the Budgeteer needs the latest SpeedBar settings. These SpeedBar settings are returned by formulas in the notebook. It recalculates these formulas.

_get_bar_values	{RECALCCOL ChangeDB_Macros..Query_Macros:B1..B150	Recalculates the formulas in ChangeDB_mac

_do_nothing

The macro _do_nothing is called whenever no records are found by a database search (the search is started by _refresh_view). It makes sure the record window is tidy by calling _restore_view.

_do_nothing	{_restore_view}	Used when a {Query.Extract} operation finds

_view_stats

The macro _view_stats is called when the user chooses Statistics from the Budgeteer menu bar. It displays the statistical information stored on the notebook page Stats and then returns to the record window.

_view_stats	{_disable_ui "Yes"}	Disable the SpeedBar and menus while addir
	{SETOBJECTPROPERTY "Stats:A1..IV8192.Protection","Prc	Protect all cells on the page. This ensures tha

	{EditGoto Stats:B1}	Activate the notebook page Stats, which disp
	{WindowTitles Vertical}	Lock column A of Stats so that no matter how
	{IF @CELLPOINTER("type")="b"}{BRANCH _restore_view}	If no statistics are available, re-activate the re
	{IF @CELL("type",[]C(1)R(0))<>"b"}{SHIFT+END}{SHIFT+RI	If any expense category names exist, select t
	{SETPROPERTY "Protection","Unprotect"}	Unprotect the cells the user can select
	{RestrictInput.Enter @PROPERTY("Active_Block.Selection")}	Confine selector movement to the first row of
	{MESSAGE help_stats,10,5,0}	Display a help message that describes how t
	{PAUSEMACRO}	Wait for the user to exit the viewing by pressi
	{_restore_view}	Activate the record window

_restore_view

The macro _restore_view is called whenever the record window needs to be activated (page Expense_View) and freshened up.

_restore_view	{EditGoto Expense_View:A1}	Activate the notebook page Expense_View, v
	{WindowTitles Clear}	Clear any locked titles, if they exist
	{DOWN}	Move to cell A2
	{COLUMNWIDTH []Expense_View:A1..IV8192,1,2,2}	Auto-size the columns so that the expense na
	{_disable_ui "No"}	Make sure the UI is enabled

ıich records display in the record window.

sed by _new_query and _refresh_query to specify the search criteria

se database being queried

.r

ʻry

ıg viewed into the variable view_page
ock from the page containing the expense database to the variable data_block
ʻdinates. The variable data_block contains a cell address that indicates where the new coordinates are (not the coordinates

e, which is used to specify search criteria

› All, place a formula in the criteria table that searches for records with the day specified
: to All, place a formula in the criteria table that searches for records with the month specified
o All, place a formula in the criteria table that searches for records with the year specified
contains the expense database specified by current_freq
: their names
t row of the record window

hem

ames aren't cropped
Stats
tats
ord window
istical information about records in the record window
e cost for each expense
he average cost of each expense
he maximum cost of each expense
he minimum cost of each expense
ames and data aren't cropped

date, so that dates display in MM/DD/YY format
ames and data aren't cropped
record display

› All, place a formula in the criteria table that searches for records with the day specified
: to All, place a formula in the criteria table that searches for records with the month specified
o All, place a formula in the criteria table that searches for records with the year specified
eria specified in the SpeedBar, an error message isn't displayed
d by the SpeedBar into the record window
e Stats that return statistical information

ames and data aren't cropped

e into the notebook page Stats
)function to enter on the notebook page Stats
mat of the database @function entered on the notebook page Stats

returns a macro command that selectes a row of the active page
tored in stat_formula into the active block
neric format specified by num_format
ultiple formulas

cros and Query_macros. This ensures that current results are used

no records, this command re-activates the record window

ng a record
at only the cells unprotected later in the macro are accessible to the user

lays a list of statistics
/ far right the user scrolls, the titles are shown
ecord window
hem

the page. This lets the users scroll through the statistics, but prevents them from editing the formulas that provide the statis
o finish viewing the statistics
ng Enter or Esc

which is the record window the user sees

ames and data aren't cropped

; themselves)

stics

_new_budget

The macro _new_budget is called when the user chooses Budget|New or clicks the New button in the initial Budgeteer dialog box. It creates a new budget file on the desktop, which the user can add records to and save. Here's the flow of the macro:

1. Check if unsaved changes exist.
2. If unsaved changes exist, warn the user the changes will be lost.
3. Check if any windows are open on the desktop, and close them.
4. Create a new notebook.
5. Set up the notebook to make it a valid budget file.

file_saved	1	Used by _open_budget, _new_budget, and _cc
file_name	NEWBUDG.BDG	Used by _save_budget to determine the name
_new_budget	{IF file_saved=1}{BRANCH no_warning}	If the file contains no unsaved changes, branch
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Yes.Bitr	Make the Yes button (in the dialog box Warn) d
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Messag	Set the warning message appearing in the dial
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Yes.Lat	Make the Yes button (in the dialog box Warn) d
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:No.Labe	Make the No button (in the dialog box Warn) dis
	{DODIALOG "[BUDGTRAK.WB1]Warn",Error_Check}	Display the dialog box Warn to caution the user
	{IF Error_Check=0}{RETURN}	If the user cancels the dialog box, stop the oper
no_warning	{_check_windows "{FileClose 0}","{FileNew}"}	Reset the SpeedBar to default settings (Monthl
	{EditGoto A:A1}	Check if any windows are open on the desktop
	{_reset_speedbar}	Select the first page of the new notebook. Note
	{Page.Name "Expense_View"}	Name the page Expense_View. This will be the
	{SETOBJECTPROPERTY "A1..A8192.Numeric_Format","Lon	Format the first column in Short Date format (M
	{SETOBJECTPROPERTY "B1..IV8192.Numeric_Format","Cu	Format the remaining columns in Currency form
	{CTRL+PGDN}{Page.Name "Stats"}	Move to the next page, and name it Stats. This
	{EditGoto A1}	Move to cell A1 of the active page
	{PUTCELL Cost}{SETPROPERTY "Font.Bold","Yes"}{DOWN	Enter the label Cost and set it to bold; move do
	{PUTCELL Total}{DOWN}	Enter the label Total; move down a cell
	{PUTCELL Average}{DOWN}	Enter the label Average; move down a cell
	{PUTCELL Maximum}{DOWN}	Enter the label Maximum; move down a cell
	{PUTCELL Minimum}{DOWN}	Enter the label Minimum; move down a cell
	{CTRL+PGDN}{Page.Name "Input"}	Move to the next page, and name it Input. This
	{CTRL+PGDN}{Page.Name "Daily"}	Move to the next page, and name it Daily. This
	{CTRL+PGDN}{Page.Name "Weekly"}	Move to the next page, and name it Weekly. Th
	{CTRL+PGDN}{Page.Name "Monthly"}	Move to the next page, and name it Monthly. Th
	{CTRL+PGDN}{Page.Name "Bi_Monthly"}	Move to the next page, and name it Bi_monthly
	{CTRL+PGDN}{Page.Name "Semi_Annually"}	Move to the next page, and name it Semi_Annu
	{CTRL+PGDN}{Page.Name "Annually"}	Move to the next page, and name it Annually. T
	{PUTBLOCK "Date",Daily..Annually:C1}	Enter the label Date in cell C1 of the pages Dai
	{SETOBJECTPROPERTY "Daily..Annually:C1.Numeric_Forn	Format the cell on each database page that wil
	{PUTBLOCK "Item List",Daily..Annually:B1}	Enter the label Item List in cell B1 of the pages
	{PUTBLOCK "Database_Block",Daily..Annually:A1}	Enter the label Database_Block in cell A1 of the
	{PUTBLOCK "Item_Block",Daily..Annually:A4}	Enter the label Item_Block in cell A4 of the pag

	{PUTBLOCK "@CELL(""ThreeDAddress"",Daily:C1)",Daily..A	Put the address of the first cell of the database
	{BlockName.Create "Database_Block","Daily:A2"}	Create the block name Database_Block, and m
	{BlockName.Create "Item_Block","Daily:A5"}	Create the block name Item_Block, and make i
	{EditGoto Expense_View:A1}	Return to the page that's used as the record wi
	{Page.Borders "No,No"}	Hide the row and column borders on the active
	{Page.Grid_Lines "Yes,No"}	Hide the vertical grid lines on the active page
	{Notebook.Display "Yes,Yes,No"}	Hide the page tabs on the active page. This hid
	{LET File_Saved,1}	Set file_saved to indicate that no unsaved char
	{_new_query}	Set up a new database query

_check_validity

The macro _check_validity is called whenever a budget file is opened to make sure the notebook loaded is a valid budget file. It checks for the existence of pages and named blocks the Budgeteer must use.

_check_validity	{IF @ISERR(@@("[]Expense_View:A1"))}{BRANCH _invalid	Check if the active notebook contains a page n
	{IF @ISERR(@@("[]Stats:A1"))}{BRANCH _invalid_file}	Check if the active notebook contains a page n
	{IF @ISERR(@@("[]Daily:A1"))}{BRANCH _invalid_file}	Check if the active notebook contains a page n
	{IF @ISERR(@@("[]Weekly:A1"))}{BRANCH _invalid_file}	Check if the active notebook contains a page n
	{IF @ISERR(@@("[]Monthly:A1"))}{BRANCH _invalid_file}	Check if the active notebook contains a page n
	{IF @ISERR(@@("[]Bi_Monthly:A1"))}{BRANCH _invalid_file	Check if the active notebook contains a page n
	{IF @ISERR(@@("[]Semi_Annually:A1"))}{BRANCH _invalid	Check if the active notebook contains a page n
	{IF @ISERR(@@("[]Annually:A1"))}{BRANCH _invalid_file}	Check if the active notebook contains a page n
	{IF @ISERR(@@("[]Database_Block"))}{BRANCH _invalid_fi	Check if the active notebook contains a block n
	{IF @ISERR(@@("[]Item_Block"))}{BRANCH _invalid_file}	Check if the active notebook contains a block n
	{_new_query}	Set up a new database query

_invalid_file

The macro _invalid_file is called when the active window can't be used as a budget file. It displays a message explaining that and closes the active window.

_invalid_file	{MESSAGE help_invalidf,10,15,0}	Display a message explaining that the active n
	{FileClose 0}	Close the active notebook
	(_new_budget}	Create a new budget file

_check_windows

The macro _check_windows is called by various macros to check whether windows are open on the desktop. If a window is open, the macro commands stored in the variable open_routine are run. If no windows are open, the macro commands stored in the variable close_routine are run. This macro is self-modifying; it changes depending on the arguments passed to it.

open_routine	{FileClose 0}	Used by _check_windows to determine what o

close_routine	{FileNew}	Used by _check_windows to determine what o
_check_windows	{DEFINE open_routine:string,close_routine:string}	Sets up the arguments the macro accepts
_check_win_loop	{BLANK open_windows}	Erase the cell named open_windows (the cell c
	{GETWINDOWLIST open_windows}	Get a list of open, unhidden windows and store
	{IF @CELL("type",[BUDGTRAK.WB1]open_windows)<>"b"}{	If the cell open_windows is blank, then no wind
	{close_routine}	If the cell open_windows isn't blank, then no wi

_save_budget

The macro _save_budget is called when the user chooses File|Save or File|Save As. It saves the active notebook under the name stored in the variable file_name, and then sets file_saved to 1 to indicate that no unsaved changes exist.

_save_budget	{ONERROR _filing_error}	If a problem occurs while saving the file, run the
	{EditGoto Expense_View:A2}	Move back to the record window
	{FileSaveAs +[BUDGTRAK.WB1]file_name,Replace}	Save the file, replacing it if it already exists on c
	{LET file_saved,1}	Set file_saved to indicate that no unsaved char

_open_budget

The macro _open_budget is called when the user chooses Budget|Open. It loads a budget file. Here's the flow of the macro:

1. Check if unsaved changes exist.
2. If unsaved changes exist, warn the user the changes will be lost.
3. Check if any windows are open on the desktop, if there aren't any, create a new notebook.
4. Request the name of a budget file to load.
5. Load the budget file into the active notebook.
6. Check if the file is a valid budget file (using _check_validity).
7. If the file is valid, start a new query using current SpeedBar settings (this step is done in the macro _check_validity). If not, close the file.

temp_file	*.BDG	Used by _open_budget and _save_budget_as
filing_result	0	Used by _open_budget and _save_budget_as

_open_budget	{IF file_saved=1}{BRANCH no_warning2}	If the file contains no unsaved changes, branch
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Yes.Bitr	Make the Yes button (in the dialog box Warn) d
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Messag	Set the warning message appearing in the dial
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Yes.Lat	Make the Yes button (in the dialog box Warn) d
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:No.Labe	Make the No button (in the dialog box Warn) di

	{DODIALOG "Warn",Error_Check}	Display the dialog box Warn to caution the use
	{IF Error_Check=0}{RETURN}	If the user cancels the dialog box, stop the ope
no_warning2	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]InitOpen:New	Hide the new button in the dialog box InitOpen
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]InitOpen:.Title	Set the title appearing at the top of the dialog b
	{LET temp_file,"*.BDG"}	Set the temp_file to *.BDG. The DODIALOG co
	{DODIALOG "[BUDGTRAK.WB1]InitOpen",filing_result,temp_	Display a dialog box requesting the name of the
	{IF filing_result=0}{QUIT}	If the user cancels the dialog box, stop the ope
	{FileClose 0}{FileNew}	Check if any windows are open on the desktop
	{ONERROR _filing_error}	If a problem occurs while opening the file, run t
	{FileRetrieve +[BUDGTRAK.WB1]temp_file}	Open the new budget file
	{LET file_name,temp_file}	Copy the new file name to file_name, which alv
	{LET file_saved,1}	Set file_saved to indicate that no unsaved char
	{_check_validity}	Check whether the new budget file is a valid bu

_save_budget_as

The macro _save_budget_as is called when the user chooses File|Save As. It requests a new name for the active budget file, and saves the active budget file under the new name.

_save_budget_as	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]InitOpen:New	Hide the New button in the dialog box InitOpen
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]InitOpen:.Title	Set the title appearing at the top of the dialog b
	{LET temp_file,@UPPER(file_name)}	Set the variable temp_file to the current name o
	{DODIALOG "[BUDGTRAK.WB1]InitOpen",filing_result,temp_	Display a dialog box requesting the new name
	{IF filing_result=0}{BRANCH _restore_view}	If the user cancels the dialog box, stop the ope
	{LET file_name,temp_file}	Copy the new file name to file_name, which alv
	{_save_budget}	Save the budget file under the new name
	{LET file_saved,1}	Set file_saved to indicate that no unsaved char
	{_restore_view}	Activate the record window

_init_screen

The macro _init_screen is called when Budgeteer first opens. It closes any windows the Budgeteer can't use, and then displays a dialog box requesting the name of a budget file to open. Here's the flow of the macro:

1. Reset the SpeedBar to default settings.
2. Close any open windows, prompting the user if any of the windows has unsaved changes.
3. Request the name of a budget file to load. If the user cancels the dialog box, or chooses New from it, create a new budget file.
4. Load the budget file into the active notebook.
5. Make sure it's a valid budget file.
6. Start a new query using _new_query

_init_screen	{_check_windows "{FileClose 1}{BRANCH _check_win_loop}	Check if any windows are open on the desktop
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]InitOpen:New	Make sure the New button in the dialog box Init

	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]InitOpen:.Title	Set the title appearing at the top of the dialog b
	{LET temp_file,"*.BDG"}	Set the temp_file to *.BDG. The DODIALOG co
	{DODIALOG "[BUDGTRAK.WB1]InitOpen",filing_result,temp	Display a dialog box requesting the name of the
	{IF filing_result=0}{BRANCH _new_budget}	If the user cancels the dialog box, create a new
	{LET file_name,temp_file}	Copy the new file name to file_name, which alv
	{ONERROR _filing_error}	If a problem occurs while opening the file, run t
	{FileNew}{_reset_speedbar}	Create a new notebook and reset the SpeedBa
	{FileRetrieve +[BUDGTRAK.WB1]temp_file}	Open the budget file
	{LET file_saved,1}	Set file_saved to indicate that no unsaved char
	{_check_validity}	Check whether the new budget file is a valid bu
	{_restore_view}	Activate the record window
	{_new_query}	Set up a new database query

_reset_speedbar

The macro _reset_speedbar is called whenever a new budget file is created. It sets the SpeedBar to the following defaults: View to Monthly, Day to All, Month to All, and Year to the current year.

_reset_speedbar	{SETOBJECTPROPERTY "[BUDGTRAK.BAR]View.Value","M	Set the View control on the SpeedBar to Month
	{SETOBJECTPROPERTY "[BUDGTRAK.BAR]Month.Value",	Set the Month control on the SpeedBar to All
	{SETOBJECTPROPERTY "[BUDGTRAK.BAR]Day.Value","A	Set the Day control on the SpeedBar to All
	{SETOBJECTPROPERTY "[BUDGTRAK.BAR]Year.Value",+	Set the Year control on the SpeedBar to the cu

_filing_error

The macro _filing_error is called whenever a problem occurs while loading or saving a budget file. It sounds the computer's bell and displays an error message.

_filing_error	{BEEP 2}{BEEP 4}{MESSAGE help_filerr,10,15,0}	Sound the computer's bell and display a messa

_confirm_quit

The macro _confirm_quit is called by the macro _clear_app when unsaved changes exist. It asks the user to verify that they want to quit the Budgeteer and lose the changes.

_confirm_quit	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Yes.Bitr	Make the Yes button (in the dialog box Warn) d
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Messag	Set the warning message appearing in the dial
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Yes.Lat	Make the Yes button (in the dialog box Warn) d
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:No.Labe	Make the No button (in the dialog box Warn) di
	{DODIALOG "Warn",Error_Check}	Display the dialog box Warn to caution the user
	{IF Error_Check=0}{QUIT}	If the user cancels the dialog box, stop the ope
open_windows		Used by _check_windows to store a list of oper

onfirm_quit to determine whether unsaved changes exist in the active budget file
of the active budget file

ı to the part of the macro that creates the new budget file; no warning is needed
lisplay the OK button bitmap
og box Warn. Message is the name of the label in the dialog box that contains this warning
lisplay Yes on its face
splay Cancel on its face
r that creating a new budget file will destroy unsaved changes to the active budget file
ration
y database, All days and months, the current year)
. If there aren't, create one
that even if the first page of the notebook is named Expense_View, A:A1 will still take you there; named pages can still be
record window
lM/DD/YY)
nat
page contains the statistical information the user views when they choose Statistics

wn a cell

page contains the form that the user can enter records with
page contains the Daily expense database
ıis page contains the Weekly expense database
his page contains the Monthly expense database
. This page contains the Bi-Monthly expense database
ually. This page contains the Semi-Annual expense database
This page contains the Annual expense database
ly through Annually
l contain the first entry date to Short Date format (MM/DD/YY)
Daily through Annually
e pages Daily through Annually
es Daily through Annually

block in cell C1 on pages Daily through Annually. This cell is used on each page to determine the coordinates of the datab
ıake it refer to the cell A2. This block name is used by many Budgeteer macros to determine where the data is located on e
t refer to the cell A5. This block name is used by many Budgeteer macros to determine where a list of expense category na
ndow
page

les the existence of other notebook pages from the user
ıges exist

amed Expense_View. If it doesn't, branch to the macro that closes the invalid budget file
amed Stats. If it doesn't, branch to the macro that closes the invalid budget file
amed Daily. If it doesn't, branch to the macro that closes the invalid budget file
amed Weekly. If it doesn't, branch to the macro that closes the invalid budget file
amed Monthly. If it doesn't, branch to the macro that closes the invalid budget file
amed Bi_Monthly. If it doesn't, branch to the macro that closes the invalid budget file
amed Semi_Annually. If it doesn't, branch to the macro that closes the invalid budget file
amed Annually. If it doesn't, branch to the macro that closes the invalid budget file
ıamed Database_Block. If it doesn't, branch to the macro that closes the invalid budget file
ıamed Item_Block. If it doesn't, branch to the macro that closes the invalid budget file

ɔtebook doesn't have the items needed to be a budget file

ɔeration should be performed if there's a window open on the desktop

peration should be performed if there are no windows open on the desktop

pen_windows is at the bottom of this page)
them starting at the named cell open_windows
ows are open, so the macro stored in open_routine is run
ndows are open, so the macro stored in close_routine is run

e macro _filing error

lisk
ges exist

to specify the files shown in the InitOpen dialog box
to find out how the user closed the InitOpen dialog box (OK or Cancel)

to the part of the macro that opens the new bugdet file; no warning is needed
lisplay the OK button bitmap
g box Warn. Message is the name of the label in the dialog box that contains this warning
lisplay Yes on its face
splay Cancel on its face

r that creating a new budget file will destroy unsaved changes to the active budget file
ration

ox
ommand in the next cell uses this cell to specify what file or list of files should display
e budget file to open
ration
. If there aren't, create one
he macro _filing_error

ways contains the name of the active budget file
nges exist
udget file

ox
of the budget file. The DODIALOG command in the next cell uses this cell to specify what file or list of files should display
of the budget file
ration
ways contains the name of the active budget file

nges exist

. If there are, close them, prompting the user to save any changes
:Open isn't hidden

ox
ımmand in the next cell uses this cell to specify what file or list of files should display
e budget file to open
budget file
vays contains the name of the active budget file
he macro _filing_error
r to default settings (Monthly database, All days and months, the current year)

ıges exist
ıdget file

ly

rrent year

ıge explaining that an error has occurred

lisplay the OK button bitmap
ɔg box Warn. Message is the name of the label in the dialog box that contains this warning
lisplay Yes on its face
splay Cancel on its face
that quitting the Budgeteer will destroy unsaved changes to the active budget file
ration

ı, unhidden windows on the desktop

referred to by their original letter names

ase block on the active page. As the PUTBLOCK command copies the entry down through the pages, the reference Daily:
ach page
mes is located on each page. _add_expense uses this list

C1 adjusts to refer to the active page (Daily:C1, Weekly:C1, Monthly:C1, and so on)

_print_budget

The macro _print_budget is called when the user chooses a command on the Print menu. It prints the budget database(s) specified by the user. Here's the flow of the macro:

1. If the user chose Print|All, run the macro _print_all instead.
2. If the user chose Print|Current, run the macro _print_current instead.
3. Activate the notebook page that contains the expense database to print.
4. Check if there's any data to print; if not, stop the operation.
5. Clean up the expense data to make it presentable.
6. Set the print block and headings to the data in the expense database.
7. Print the expense database and return to the record window.

print_data	Annually	Used by _print_budget to determine what data
entire_block	Monthly:C1..Monthly:E3	Used by _print_budget to store the coordinate
p_width	2	Used by _print_budget to store the width of the
p_height	2	Used by _print_budget to store the height of th
print_block	[]C(0)R(0)..C(1)R(1)	Used by _print_budget to calculate the block c

_print_budget	{DEFINE print_data:string}	Sets up the arguments the macro accepts
	{IF print_data="All"}{BRANCH _print_all}	If the user requested that all data be printed, r
	{IF print_data="Current"}{BRANCH _print_current}	If the user requested that all data in the record
	{EditGoto +[BUDGTRAK.WB1]print_data&":C2"}	Activate the page in the active notebook that c
	{IF @CELL("type",[]D1)="b"}{_restore_view}{RETURN}	If D1 on this page is blank, no expense categc
	{IF @CELLPOINTER("type")="b"}{_restore_view}{RETURN}	If C2 on this page is blank, no records exist in
	{BlockCopy +[BUDGTRAK.WB1]print_data&":database_block	Copy the database block coordinates to the va
	{COLUMNWIDTH +[BUDGTRAK.WB1]entire_block,1,2,2}	Auto-size the columns so that the expense na
	{EditGoto +[BUDGTRAK.WB1]entire_block}	Select the block of data that will print
	{SETPROPERTY "Line_Drawing","Clear,Clear,Clear,Clear,Tl	Draw thin vertical lines between each column
	{LET p_width,@COLS(@@(+"[]"&entire_block))-1}	Store the number of columns in the selected b
	{LET p_height,@ROWS(@@(+"[]"&entire_block))-1}	Store the number of rows in the selected block
	{RECALCCOL Print_macros:B1..B100}	Update the formulas on this page that calculat
	{SelectBlock +[BUDGTRAK.WB1]print_data&":1"}	Select the first row of the active page (contain
	{SETPROPERTY "Alignment","Center"}	Center the expense category names
	{EditGoto D2}	Move to the first cell of the first record in the e
	{Print.PrintReset}	Clear any old print settings
	{Print.Orientation Landscape}	Specify that the data should print sideways on
	{Print.Top_Heading C1}	Set the top heading to the first row (the expen
	{Print.Left_Heading C1}	Set the left heading to the first column (the ent
	{Print.Block +[BUDGTRAK.WB1]print_block}	Set the print block to the coordinates calculate
	{Print.DoPrint}	Print the data

	{SelectBlock +[BUDGTRAK.WB1]entire_block}	Select the database block again
	{SETPROPERTY "Line_Drawing","Clear,Clear,Clear,Clear,Cl	Erase any line drawings left previously
	{_restore_view}	Activate the record window

_print_all

The macro _print_all is called by _print_budget when the user chooses Print|All from the Budgeteer menu. It prints all the expense databases using _print_budget.

_print_all	{_print_budget "Daily"}	Print Daily expense database
	{_print_budget "Weekly"}	Print Weekly expense database
	{_print_budget "Monthly"}	Print Monthly expense database
	{_print_budget "Bi_Monthly"}	Print Bi-Monthly expense database
	{_print_budget "Semi_Annually"}	Print Semi-Annual expense database
	{_print_budget "Annually"}	Print Annual expense database

_print_current

The macro _print_current is called by _print_budget when the user chooses Print|Current from the Budgeteer menu. It prints the records displaying in the record window. Here's the flow of the macro:

1. Check if there are any records to print; if not, stop the operation.
2. Set the print block and headings to the data in the record window.
3. Activate the notebook page that contains the expense database to print.
4. Check if there's any data to print; if not, stop the operation.
5. Clean up the expense data to make it presentable.
6. Print the expense data and return to the record window.

top_heading	Expense_View:B1..Expense_View:C1	Used by _print_current to specify the top heac
left_heading	Expense_View:A2..Expense_View:A3	Used by _print_current to specify the left head
whole_block	[]C(-1)R(-1)..C(1)R(1)	Used by whole_block to calculate the block of
_print_current	{EditGoto Expense_View:B1}	Activate the record window
	{IF @CELLPOINTER("type")="b"}{BRANCH _restore_view}	If no categories are shown in the record windc
	{IF @CELL("type",[]A2)="b"}{BRANCH _restore_view}	If no records are shown in the record window,
	{IF @CELL("type",[]C(1)R(0))<>"b"}{SHIFT+END}{SHIFT+RI	If any categories exist in the database, select
	{LET top_heading,@PROPERTY("Active_Block.Selection")}	Set the top heading to the first row (the expen
	{LET p_width,@COLS(@@(+"[]"&top_heading))}	Store the number of columns in the selected b
	{EditGoto A2}	Move to the cell A2
	{IF @CELL("type",[]C(0)R(1))<>"b"}{SHIFT+END}{SHIFT+DC	If more than one record exists, select each cel

{LET left_heading,@PROPERTY("Active_Block.Selection")}	Set the left heading to the first column (the ent
{LET p_height,@ROWS(@@(+"[]"&left_heading))}	Store the number of rows in the selected block
{RECALCCOL Print_macros:B1..B100}	Update the formulas on this page that calculat
{EditGoto B2}	Move to the cell B2
{COLUMNWIDTH +[BUDGTRAK.WB1]whole_block,1,2,2}	Auto-size the columns so that the expense na
{SelectBlock +[BUDGTRAK.WB1]whole_block}	Select the whole block of data, including expe
{SETPROPERTY "Line_Drawing","Clear,Clear,Clear,Clear,Th	Draw thin vertical lines between each column
{EditGoto B2}	Move to the first cell of the first record in the e
{Print.PrintReset}	Clear any old print settings
{Print.Orientation Landscape}	Specify that the data should print sideways on
{Print.Top_Heading +[BUDGTRAK.WB1]top_heading}	Set the top heading to the first row (the expen
{Print.Left_Heading +[BUDGTRAK.WB1]left_heading}	Set the left heading to the first column (the ent
{Print.Block +[BUDGTRAK.WB1]print_block}	Set the print block to the coordinates calculate
{Print.DoPrint}	Print the data
{SelectBlock +[BUDGTRAK.WB1]whole_block}	Select the whole block of records and names a
{SETPROPERTY "Line_Drawing","Clear,Clear,Clear,Clear,Cl	Erase any line drawings left previously
{EditGoto B2}	Activate the record window

a should print
s of the block that will print
e expense database, in columns
ne expense database, in rows
of data to set as the print block. This formula is needed because the first column and row of the expense database can't be

un the macro _print_all
l window be printed, run the macro _print_current
contains the expense database specified by print_data
ories exist in the database, and the print operation is stopped
the database, and the print operation is stopped
ariable entire_block
mes and data aren't cropped

of the block
lock in the variable p_width
k in the variable p_height
e the print block and headings
ing the expense names)

xpense database

the page
se category names). If the printout spans multiple pages, this row prints at the top of each page
try dates). If the printout is too wide to fit on one page, this column prints at the left of each page
ed by the formula stored in the variable print_data

ling of the printout. This information prints at the top of each page of the printout

ling of the printout. This information prints at the left side of each page of the printout

data to set as the print block. This formula is needed because the first column and row of the data can't be included in the

w, stop the print operation

stop the print operation

their names

se category names). If the printout spans multiple pages, this row prints at the top of each page

lock in the variable p_width

l in the first column that contains a record

try dates). If the printout is too wide to fit on one page, this column prints at the left of each page
k in the variable p_height
e the print block and headings

mes and data aren't cropped
nse names
of the block
xpense database

the page
se category names). If the printout spans multiple pages, this row prints at the top of each page
try dates). If the printout is too wide to fit on one page, this column prints at the left of each page
d by the formula stored in the variable print_data

again

included in the print block; they're used as the top and left heading

print block; they're used as the top and left heading

_graph_view

The macro _graph_view is called when the user chooses Graph from the Budgeteer SpeedBar. It constructs a macro (named _build_graph) that creates a graph using the data in the record window. Here's the flow of the macro:

1. Check if there are records to graph; if not, stop the operation.
2. Select the first row of the record window and store its coordinates in the block legend_series. This block is used later.
3. Measure the width of the row; this is to determine how many series have to be created in the graph.
4. Select the first column of the record window and store its coordinates in the block xaxis_series. This block is used later.
5. Measure the height of the column; this is to determine how many data points will exist in each series.
6. Using the measurements previously calculated, create a list of macros commands starting at the cell named _build_series that creates series in the graph.
7. Copy the macros commands stored in the block finishing_mac just below the commands created in the previous step.
8. Run the macro that builds the graph (_build_graph).

g_width	2	Used by _graph_view to store the number of
g_height	2	Used by _graph_view to store the number of
legend_series	Expense_View:B1..Expense_View:C1	Used by _graph_view to store the block in the
xaxis_series	Expense_View:A2..Expense_View:A3	Used by _graph_view to store the block in the
_graph_view	{EditGoto Expense_View:B1}	Activate the record window, and select the fir
	{IF @CELLPOINTER("type")="b"}{BRANCH _restore_view}	If there's no first record, stop the operation
	{IF @CELL("type",[]A2)="b"}{BRANCH _restore_view}	If there are no expense categories, stop the c
	{IF @CELL("type",[]C(1)R(0))<>"b"}{SHIFT+END}{SHIFT+RI	If expense categories exist, select their name
	{LET legend_series,@PROPERTY("Active_Block.Selection")}	Set the legend series to the selected block. A
	{LET g_width,@COLS(@@(+"[]"&legend_series))}	Store the number of series to create in the va
	{EditGoto A2}	Move to cell A2
	{IF @CELL("type",[]C(0)R(1))<>"b"}{SHIFT+END}{SHIFT+DC	If more than one record exists, select all cells
	{LET xaxis_series,@PROPERTY("Active_Block.Selection")}	Set the X-Axis series to the selected block. T
	{LET g_height,@ROWS(@@(+"[]"&xaxis_series))}	Store the number of data points in each serie
	{EditGoto A2}	Move to cell A2
	{FOR series_number,1,g_width,1,_add_series}	Dynamically create the macro commands tha
	{RECALCCOL Graph_macros:B1..B250}	Update the formulas in this page that determi
	{BlockCopy [BUDGTRAK.WB1]finishing_mac,+[BUDGTRAK.	Copy the macro stored in finishing_mac to jus
	{_build_graph}	Run the macro that actually builds the graph

finishing_mac

The macro finishing_mac isn't called by itself; it's copied to the end of the list of macro commands created by _graph_view. Here's the flow of the macro:

1. Create the series just defined by _build_series.
2. Make the graph more presentable.
3. Display the graph full screen.
4. Ask the user if they'd like to print the graph; if so, print the graph.
5. Activate the record window.

finishing_mac	{Series.Go}	Create the series
	{SETOBJECTPROPERTY "G$Y1Axis.Numeric_Format","Cur	Set the numeric format of the current graph's
	{WindowClose}	Close the active graph window
	{GraphView Expense}	View the graph just created
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Yes.Bitr	Make the Yes button (in the dialog box Warn)
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Messag	Set the warning message appearing in the di
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:Yes.Lat	Make the Yes button (in the dialog box Warn)
	{SETOBJECTPROPERTY "[BUDGTRAK.WB1]Warn:No.Labe	Make the No button (in the dialog box Warn)
	{DODIALOG "[BUDGTRAK.WB1]Warn",Error_Check}	Display the dialog box Warn to ask the user if
	{IF Error_Check=0}{SELECTBLOCK Expense_View:A2}{RET	If the user cancels the dialog box, don't print t
	{GRAPHPAGEGOTO}	Activate the Graphs page
	{SELECTOBJECT Expense}	Select the icon representing the named graph
	{Print.DoPrintGraph}	Print the named graph Expense
	{SELECTBLOCK Expense_View:A2}	Activate the record window
	{RETURN}	Stop the macro

_add_series

The macro _add_series is called by _graph_view to create a command in the macro _build_graph that creates a graph series. This command is copied into the cell whose address is calculated by the formula stored in command_cell (which recalcutes to a different address each time _add_series is called). Here's the flow of the macro:

1. Select the data that this series plots. The coordinates are calculated by the formula stored in series_block.
2. Copy the coordinates of the selected block to the variable data_range.
3. Recalculate the formulas that calculate the series command to create.
4. Copy the command created by the formula stored in series_command into the cell whose address is calculated by the formula stored in command_cell.

series_block	[]C(3)R(0)..C(3)R(1)	Used by _add_series to return the relative ad
data_range	Expense_View:C2..Expense_View:C3	Used by _add_series to store the coordinates

series_number	3	Used by _add_series to store the series numl
series_command	{Series.Data_Range 3,Expense_View:C2..Expense_View:C3	Used by _add_series to return the command

_add_series	{EditGoto A2}	Move to cell A2
	{SelectBlock +[BUDGTRAK.WB1]series_block}	Select the block whoses coordinates are com
	{LET data_range,@PROPERTY("Active_Block.Selection")}	Store the regular block coordinates of the sel
	{RECALCCOL Graph_macros:B1..B250}	Recalculate the formula that determines the s
	{LET +command_cell,+series_command}	Copy the series command calculated in the n
command_cell	[BUDGTRAK.WB1]Graph_macros:B137	Used by _add_series to return the cell addres

_build_graph

The macro _build_graph is called by _graph_view to create a graph that plots the expense data in the record window. The portion of _build_graph above the named cell _build_series never changes; starting at _build_series, commands are created by _graph_view that create the series plotted in the graph, display the graph full screen, and ask the user if they want to print the graph.

1. Select the data that this series plots. The coordinates are calculated by the formula stored in series_block.
2. Copy the coordinates of the selected block to the variable data_range.
3. Recalculate the formulas that calculate the series command to create.
4. Copy the command created by the formula stored in series_command into the cell whose address is calculated by the formula stored in command_cell.

_build_graph	{IF @ISERR(@PROPERTY("Expense.Aspect_Ratio"))}{Grapl	If the named graph Expense doesn't exist, cr
	{GraphEdit "Expense"}	Display the graph Expense in a graph windov
	{GraphSettings.Type "Line"}	Set the graph type to Line
	{Series.Data_Range "LegendSeries",+[BUDGTRAK.WB1]leg	Set the legend series to the coordinates store
	{Series.Data_Range "XAxisLabelSeries",+[BUDGTRAK.WB1	Set the x-axis series to the coordinates stored
_build_series		The first available cell to copy macro comma

series in the graph
points on each series of the graph
e record window that contains legend text
e record window that contains labels for the X-Axis

st cell of the first record

peration
s
legend appears in the graph showing what each series represents; this legend series specifies the labels that appear in th
riable g_width

in the first column that contain entry dates
he X-Axis series specifies the labels that appear at the bottom of the graph
s in the variable g_height

t will create the series in the graph. These commands are stored starting at the named cell _build_series
ne where the last part of _build_graph should be placed
st after the series commands created by _add_series

Y-Axis to Currency

) display the print button bitmap
alog box Warn. Message is the name of the label in the dialog box that contains this warning
) display Print on its face
display No on its face
f they wish to print the graph
the graph and return to the record window

ı Expense

dress of the series data to select
s of the series to add

ber being created
that will be copied to _build_series

puted by the formula in series_block
ected block in data_range
series command to add to the macro _build_graph
amed cell series_command to the cell calculated by the formula command_cell

ss of the next cell that the macro can copy a command to

eate it
v for editing

ed in the variable legend_series
d in the variable xaxis_series
nds in to. As the macro _graph_view runs, various macro commands are created, starting here

at legend

All	All	Daily
January	Monday	Weekly
February	Tuesday	Monthly
March	Wednesday	Bi-monthly
April	Thursday	Semi-Annually
May	Friday	Annually
June	Saturday	
July	Sunday	
August		
September		
October		
November		
December		