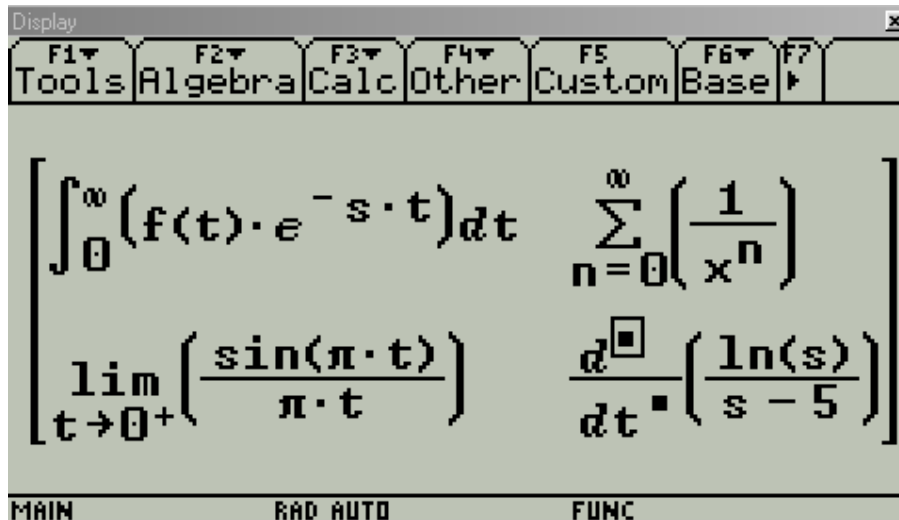




User Manual for Equation Writer:
A Graphical Equation Writer
for the TI-89 and TI-92Plus

EQW version 1.01 Flash Application



EQW program © 2002 Creative SW Design
User Manual © 2002 Creative SW Design



1. DISCLAIMER

In no event shall Creative SW Design be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the use of these materials. Moreover, Creative SW Design shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.



2. Table of Contents

1. DISCLAIMER	2
2. TABLE OF CONTENTS	3
3. INTRODUCTION	6
3.1 CONVENTIONS USED IN THIS DOCUMENT.....	7
3.2 EQW FLASH APPLICATION VS. EQW PROGRAM.....	7
3.2.1 <i>EQW Hotkeys</i>	7
3.2.2 <i>New editing functions</i>	8
3.2.3 <i>Improved editing functions</i>	8
3.2.4 <i>Evaluating expressions</i>	8
3.2.5 <i>Changing text size</i>	8
3.2.6 <i>View Screen</i>	8
3.2.7 <i>Moving cursor</i>	9
3.2.8 <i>Typing functions</i>	9
3.2.9 <i>New menus</i>	9
4. INSTALLING AND UNINSTALLING	11
4.1 REQUIREMENTS.....	11
4.2 FILES.....	11
4.3 TRANSFERRING FILES.....	11
4.4 UNINSTALLING THE EQW.....	12
5. STARTING AND EXITING	12
5.1 STARTING THE EQW.....	13
5.1.1 <i>Via the Flash Application menu</i>	13
5.1.2 <i>Via the API</i>	13
5.2 THE START-UP SCREEN.....	13
5.2.1 <i>Via keyboard shortcuts</i>	13
5.3 EXITING THE EQW.....	14
5.3.1 <i>Exiting and passing the content of the editor to Home</i>	14
6. ENTERING EXPRESSIONS	14
6.1 PLACING PARENTHESIS USING THE CURSOR.....	15
6.2 ENTERING FUNCTIONS/COMMANDS/PROGRAMS.....	17
6.2.1 <i>Typing functions/commands/program names</i>	19
6.3 ENTERING A MATRIX.....	20
6.4 ENTERING VECTORS.....	20
6.5 ENTERING A LIST.....	21
6.6 ENTERING A STRING.....	22
6.7 INSERTING SUBSCRIPTION.....	23
6.8 ENTERING DEGREE MINUTES SECONDS (DMS).....	23
7. MOVING CURSOR	23
7.1 CONTROLLING THE MOVING MODE:.....	24
8. EDITING AN EXPRESSION	25
8.1 MARKING SUB-EXPRESSIONS.....	25
8.1.1 <i>Marking adjacent expressions</i>	27
8.2 DELETING PARTS OF AN EXPRESSION.....	27
8.2.1 <i>Deleting functions leaving the argument: [DEL]</i>	28
8.2.2 <i>Deleting optional arguments: [BACKSPACE]</i>	28



8.2.3	Deleting single characters: [BACKSPACE]	28
8.2.4	Deleting parts of an expression: [BACKSPACE]	28
8.2.5	Deleting rows and columns of a matrix [BACKSPACE]	29
8.3	[COPY], [CUT] AND [PASTE]	29
8.4	UNDOING OPERATIONS [INS]	30
9.	EVALUATING EXPRESSIONS	30
10.	OPENING/RECALLING VARIABLES	32
11.	SAVING EXPRESSIONS	35
11.1	EXAMPLE: DIFFERENCES BETWEEN STO AND "SAVE COPY AS.."	35
11.2	VIEWING THE CONTENT OF A VARIABLE.	37
12.	CREATING FORMS	37
12.1	USING STORED EXPRESSIONS AS FORMS	38
12.2	USING STRINGS AS FORMS	38
13.	THE VIEW SCREEN [F8]	38
14.	LOCAL AND GLOBAL VARIABLES	39
15.	CUSTOMISING THE EQW	39
15.1	THE FORMAT MENU [F1][C]	39
15.1.1	Auto Execute	40
15.1.2	Keyboard Shortcut	40
15.1.3	Text Size/View Text Size	41
15.2	USING THE STANDARD CUSTOM MENUS	42
15.3	EQW KEYBOARD PROGRAMS	42
16.	EQW KEYS AND FUNCTIONS	43
16.1	CURSOR MOVEMENTS	43
16.2	SCROLL EXPRESSION	44
16.3	EXPRESSION EVALUATION	44
16.4	EXPRESSION EDITING	44
16.5	MARKING EXPRESSION	45
16.6	OPERATORS	45
16.7	FUNCTIONS	45
16.8	MATRIX	45
16.9	LISTS	46
16.10	STRINGS	46
16.11	MENUS AND WINDOWS	46
16.12	KEYBOARD PROGRAMS	46
16.13	SWITCHING APPLICATION	47
16.14	OTHER KEYS	47
17.	API	48
17.1	EQW.CURRENT	48
17.1.1	EQW.Current()	48
17.2	EQW.GET	48
17.2.1	EQW.Get()	48
17.3	EQW.INPUT	49
17.3.1	EQW.Input()	49
17.3.2	EQW.Input(VAR)	50
17.4	EQW.INSERT	51
17.4.1	EQW.Insert(expression)	51
17.4.2	EQW.Insert(expression-string)	52
17.4.3	EQW.Insert(command-string)	52



17.4.4	<i>EQW.Insert(key-string)</i>	53
17.5	<i>EQW.NEW</i>	55
17.5.1	<i>EQW.New()</i>	55
17.6	<i>EQW.OPEN</i>	55
17.6.1	<i>EQW.Open(VAR)</i>	55
18.	TROUBLESHOOTING	57
18.1	THE EQW WILL NOT MARK THE NOMINATOR/DENOMINATOR OF A FRACTION.....	57
18.1.1	<i>Unevaluated numerical fractions</i>	57
18.1.2	<i>Evaluated fractions</i>	57
18.2	HOW TO ENTER AN EMPTY STRING	57
18.3	THE CURSOR DISAPPEARS WHEN INSIDE AN EMPTY STRING	57
18.4	THE EQW WILL NOT DELETE A PLACEHOLDER	57
18.5	HOW TO CHANGE THE DIRECTION OF A LIMIT.....	57
18.6	ERROR WHEN TRYING TO USE A VARIABLE CREATED BY THE EQW	57
18.7	UNABLE TO DELETE THE EQW OR APPLICATIONS INSTALLED BEFORE THE EQW	57



3. Introduction

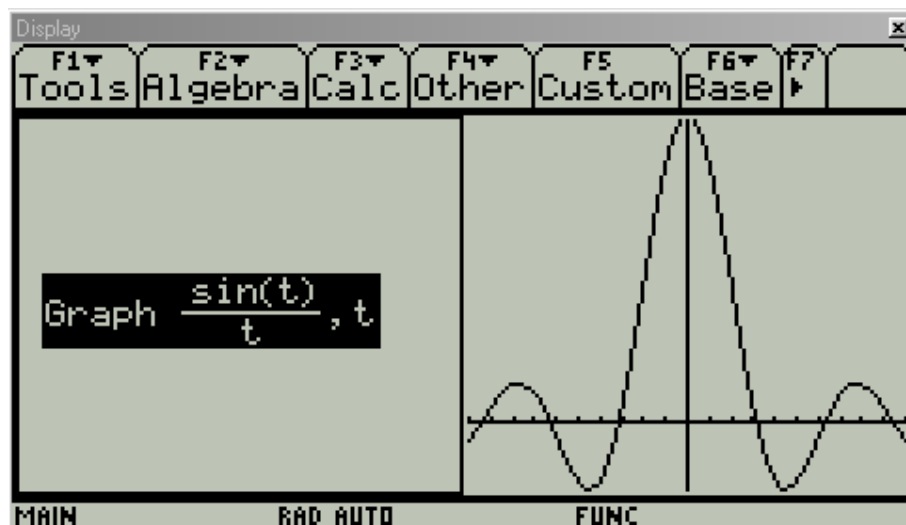
EQW is a Graphical Equation Writer for Texas Instruments TI-89 and TI-92Plus calculators. With the EQW equations, expressions, lists and matrix can be entered in a graphical format instead of the normal entry-line format. This is often a faster and less error prone way to enter and edit complex expressions because they are displayed in the textbook format as they are being typed and each element can be edited separately.

The Equation Writer can edit expressions, equations, lists, vectors, matrices and text. In this document the expression will be used as a blanket term.

The EQW can be used as a calculation environment because expressions can be evaluated inside the EQW. You can even execute programs and commands in the EQW. Expressions can be stored into variables or transferred to the Home entry-line. Expressions can be Copy/Pasted between the EQW and other applications.

The application includes an extensive API, which makes it possible to create advanced menu-systems/functions in the EQW. The API also includes an input function, which can be used by other programs to get input from users via the EQW instead of the usual input functions.

The Equation Writer is a fully integrated Flash Application. It is capable of working in all calculator modes including Split Screen mode. You can freely switch between the EQW and other applications in the same way as between the built in applications.





3.1 Conventions used in this document

In this document, it is assumed that you are familiar with the basic operations of your calculator.

Square brackets are used to indicate keys on the calculator. For example, the HOME button is written as [HOME]. The key-names used are those printed at or above the calculator buttons. For example the key [MATH] is a shifted key, which is reached by pressing [2nd][5].

[DIAMOND] = the green diamond key.

[SHIFT] = the shift key

[LEFT], [RIGHT], [UP] and [DOWN] = The cursor movement keys.

[*] = The multiplication key.

EQW = Equation Writer.

A single element = a variable, number, placeholder or fraction.

The screen shots used in this document is from the TI-92 Plus.

3.2 EQW Flash Application vs. EQW program

This section lists the differences between the Flash Application and the program version of the EQW. You can drop this section if you haven't used the program. In the Flash Application some key-functions have been moved other have been removed to make room for other functions.

New in the Flash Application is that it can work in split screen mode, that will say you can have the EQW in one part of the screen and another application in another part. The text size used by the editor can now be changed. The EQW has got a View Screen, which makes it possible to view larger expressions without having to use scroll through it.

The application contains a set of new API functions, which can be used to customise the EQW. The API can be used to create functions/menus, which works directly with the content of the EQW or send a series key-presses to the EQW (see API chapter 17).

The Flash Application has a number of new functions and menus. And some functions have been improved in the Flash version.

3.2.1 EQW Hotkeys

The EQW can define hotkeys, which can be used to activate the EQW from build-in applications.

[DIAMOND][(-)] Activate the EQW displaying the marked expression. An example of the use of this function could be to start EQW from the history display, with the cursor in the history display. This will transfer the history item into the EQW.

[DIAMOND][CLEAR] This just activates the EQW with its current content.

See: The Format menu [F1][C] 15.1



3.2.2 New editing functions

[SHIFT][UP]	Swap marked sub-expression with sub-expression to the left. This function does also swap arguments of functions/commands, items in lists/matrix and rows in matrix.
[SHIFT][DOWN]	Swap marked sub-expression with sub-expression to the right. This function does also swap arguments of functions/commands, items in lists/matrix and rows in matrix.
[SHIFT][LEFT]	Extends marked area to include adjacent sub-expression to the left.
[SHIFT][RIGHT]	Extends marked area to include adjacent sub-expression to the right.

3.2.3 Improved editing functions

[DEL]	Deletes any function/command and preserves the argument.
[PASTE]	Rows in matrices can now be copy/pasted.

3.2.4 Evaluating expressions

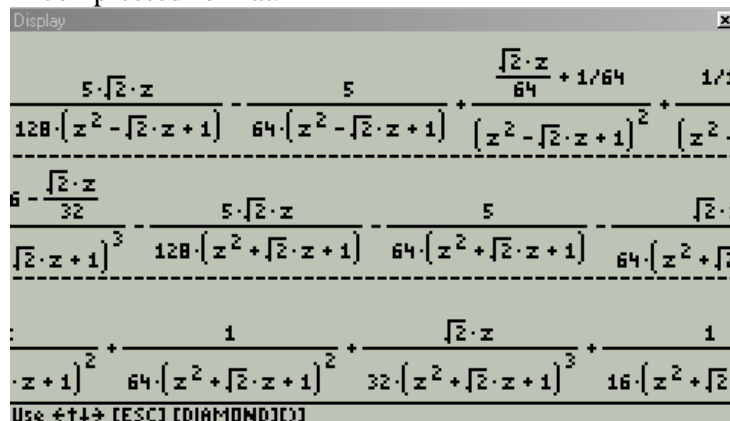
[ENTER]	Evaluates the entire expression in the EQW. The exit function has been moved to [ESC]. The [ENTER] key can as before be used to jump to placeholders.
[ENTRY]	Evaluates marked part of expression previous [F1]. The entry() function can be inserted using the Other menu.

3.2.5 Changing text size

[DIAMOND][)]	Decreases the font size used in the editor. Large text ⇒ Medium text Medium text ⇒ Small text Small text ⇒ Large text
--------------	--

3.2.6 View Screen

[F8]	Activates the View Screen. The View Screen displays the content of the editor in compressed format.
------	---



Above picture shows the TI-92 Plus View Screen displaying the result of $\text{expand}(z^2/(z^4+1)^3,z)$ using small fonts.



3.2.7 Moving cursor

The behaviour of the cursor [LEFT]/[RIGHT] has been changed to make it easier to move between sub-expressions.

3.2.8 Typing functions

1. It is no longer necessary to enter the starting parenthesis by build-in functions.
2. Custom functions can now be entered as function-name + starting parenthesis.

3.2.9 New menus

The line-menu has been replaced with a toolbar-menu.

Keys	Description
[MODE]	The system mode menu. The About screen has been moved to the Tools menu.
[APPS]	The system APPS menu. The custom menu has been move to [F5]
[DIAMOND] [APPS]	The system Flash Application menu.
[MEM]	The system memory menu.
[DIAMOND] [k] (TI-92+) / [DIAMOND] [EE] (TI-89)	The built-in keys display.



Key	Description												
[F1]	<p data-bbox="415 289 607 315">EQW Tools menu.</p> <div data-bbox="699 312 997 569" style="border: 1px solid black; padding: 2px; margin: 10px auto; width: fit-content;"> <pre> F1+ Tools 1:Open... +O 2:Save Copy As... +S 3:New... +N 4:View F8 5:Undo INS 6:To Home 7:Cut +X 8:Copy +C 9:Paste +U A:Delete + B:Clear Editor C:Format... +F </pre> </div> <p data-bbox="415 571 867 596">The Tools contains a couple of new functions</p> <table border="1" data-bbox="415 638 1282 1348"> <tr> <td data-bbox="415 638 634 743">Open</td> <td data-bbox="634 638 1282 743">Let you open a variable in same way as you would open a program in the Program Editor. The variable is opened unevaluated.</td> </tr> <tr> <td data-bbox="415 743 634 869">Save Copy As..</td> <td data-bbox="634 743 1282 869">Let you save the content of the EQW to a variable. The expression is saved unevaluated. This function let you create active variables. Active variables are first evaluated when used.</td> </tr> <tr> <td data-bbox="415 869 634 932">New</td> <td data-bbox="634 869 1282 932">Create new expression.</td> </tr> <tr> <td data-bbox="415 932 634 995">View</td> <td data-bbox="634 932 1282 995">View the content of the editor in multi-line Pretty Print.</td> </tr> <tr> <td data-bbox="415 995 634 1079">To Home</td> <td data-bbox="634 995 1282 1079">Transfers the marked part of an expression to the Home stack area.</td> </tr> <tr> <td data-bbox="415 1079 634 1348">Format</td> <td data-bbox="634 1079 1282 1348">Let you turn Auto Execution on/off. This mode setting controls the action performed when a function/command is selected in a menu. If Auto Execution is ON the selected function is executed and the result is inserted into the EQW. If Auto Execution is OFF the selected function is inserted into the expression. Installing of hotkeys. The font size used in the editor and in the View Screen can also be changed in via the Format dialog.</td> </tr> </table>	Open	Let you open a variable in same way as you would open a program in the Program Editor. The variable is opened unevaluated.	Save Copy As..	Let you save the content of the EQW to a variable. The expression is saved unevaluated. This function let you create active variables. Active variables are first evaluated when used.	New	Create new expression.	View	View the content of the editor in multi-line Pretty Print.	To Home	Transfers the marked part of an expression to the Home stack area.	Format	Let you turn Auto Execution on/off. This mode setting controls the action performed when a function/command is selected in a menu. If Auto Execution is ON the selected function is executed and the result is inserted into the EQW. If Auto Execution is OFF the selected function is inserted into the expression. Installing of hotkeys. The font size used in the editor and in the View Screen can also be changed in via the Format dialog.
Open	Let you open a variable in same way as you would open a program in the Program Editor. The variable is opened unevaluated.												
Save Copy As..	Let you save the content of the EQW to a variable. The expression is saved unevaluated. This function let you create active variables. Active variables are first evaluated when used.												
New	Create new expression.												
View	View the content of the editor in multi-line Pretty Print.												
To Home	Transfers the marked part of an expression to the Home stack area.												
Format	Let you turn Auto Execution on/off. This mode setting controls the action performed when a function/command is selected in a menu. If Auto Execution is ON the selected function is executed and the result is inserted into the EQW. If Auto Execution is OFF the selected function is inserted into the expression. Installing of hotkeys. The font size used in the editor and in the View Screen can also be changed in via the Format dialog.												
[F2]	The Algebra menu. This menu contains the same functions as the system Algebra menu except a couple of extra trigonometric functions in the Trig submenu. The extra trigonometric functions are included in the application, but has no special relation to the EQW.												
[F3]	The Calc menu contains the same functions as the system Calc menu.												
[F4]	The Other menu contains the same functions as the system Other menu.												
[F5]	The EQW Custom menu (executes the program main\eqwuser)												
[F6]	The Base menu contains the same functions as the system MATH\Base menu.												
[F7]	Activates the secondary toolbar menu.												
[F7][F2]	Number menu contains the same functions as the system MATH\Number menu.												
[F7][F3]	Angle menu contains the same functions as the system MATH\Angle menu.												
[F7][F4]	List menu contains the same functions as the system MATH>List menu.												
[F7][F5]	The Vector menu can be used to insert vectors (some types of vectors can not be inserted in any other way).												
[F7][F6]	Matrix menu contains the same functions as the system MATH\Matrix menu.												



4. Installing and uninstalling

4.1 Requirements

The EQW requires the calculator to operate on AMS v. 2.04 or newer versions.
To transfer of the program to the calculator you need TI-GRAPH LINK™ from Texas Instruments.

4.2 Files

There are two different versions of the EQW. One for TI-89 and one for TI-92Plus:

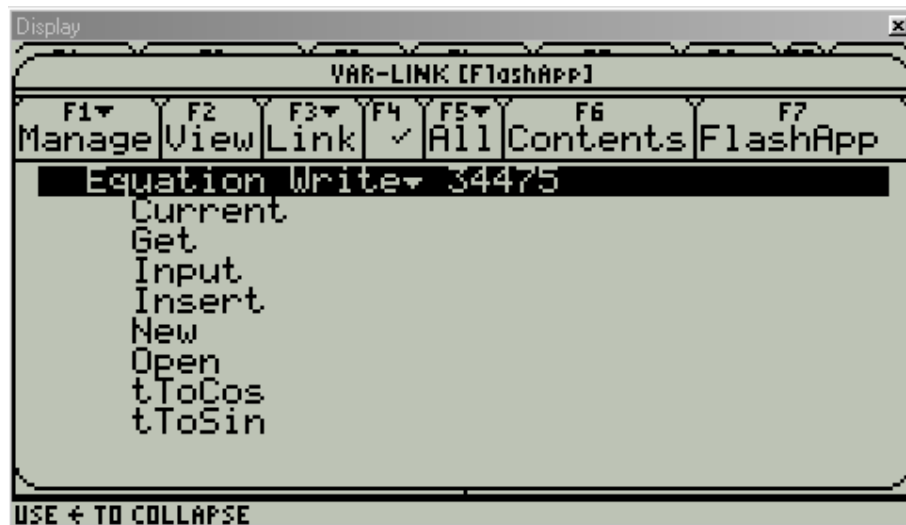
TI-89: EQW89.89k
TI-92Plus: EQW92.9xk

4.3 Transferring files

To transfer the program to the calculator using TI-GRAPH LINK™:

1. Activate the “Link” menu
2. Select the “Send Flash Software...” submenu.
3. Select the “Application and Certificate...” submenu-item.
4. Select the appropriate file.
5. Press the button “Add”.
6. Press the button “OK” to start transferring files.

When the application is installed it will become visible in several menus.



Pressing [F7] in the VAR-LINK menu will show the installed Flash Applications. The above picture shows the Equation Writer with the Basic extensions that are included in the application.

Pressing [F6] when “Equation Writer” is selected will show the version and copyright information.



4.4 Uninstalling the EQW

1. Remove eventually installed EQW hotkeys. Hotkeys are removed by selecting Tolls\Format... ([F1][C]) in the EQW and setting Keyboard Shortcut to OFF.
2. Press [QUIT] to exit the EQW
3. Press [VAR-LINK] to activate the menu.
4. Press [F7] to go to the Flash page.
5. Highlight "Equation Writer" in the list by selecting it.
6. Press [F1] to activate the "Manage" menu.
7. Press [1] or [ENTER] to select the "Delete" item.
8. Press [ENTER] to delete the application.

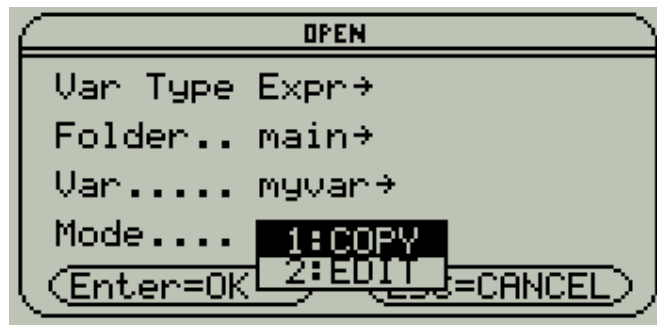
5. Starting and exiting

The EQW has three different opening/starting methods.

1. Current
2. Open
3. New

Opening the editor with the Current method will let you continue editing from the point where you left off.

The Open method will let you copy/edit the content of an existing variable. The editor will be opened with the content of the selected variable.



The New method will open a blank editor.



5.1 Starting the EQW

The EQW can be activated in three different ways.

5.1.1 Via the Flash Application menu

1. Press [DIAMOND][APPS] or [APPS][ENTER] to activate the Flash Application menu.
2. Select "Equation Writer" in the menu.
3. Select opening method:
 - Current* Continue editing an open expression/variable.
 - Open* Open a variable for editing. Selecting Open will activate a window where the variable to edit can be chosen (see Opening and Saving variables).
 - New* Open the EQW ready for entering a new expression.

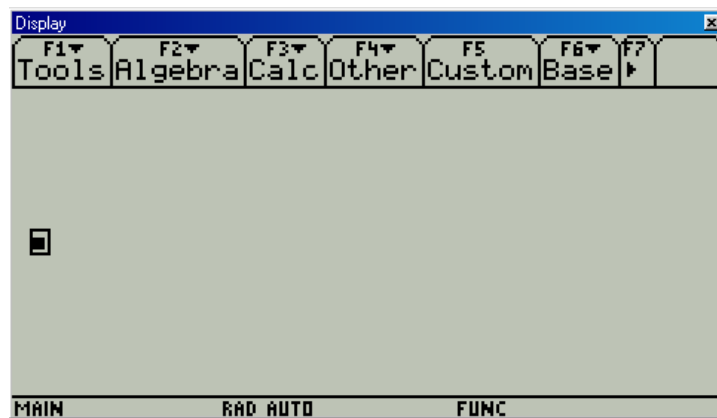
5.1.2 Via the API

The EQW can also be started from the Entry Line or from the program by executing an API program.

- EQW.Current() Continue editing of an open expression/variable (see API functions).
- EQW.New() Open the EQW ready for entering a new expression (see API functions).
- EQW.Open(VAR) Open the variable VAR for editing (see API functions).

5.2 The start-up screen

When the EQW is started the screen will look as follows:



The menu-line at the top of the screen shows the available pull down menus. The black square in the centre of the screen is a placeholder. Placeholders are used to show where a text can be entered. The rectangle surrounding the placeholder is the cursor.

5.2.1 Via keyboard shortcuts

The EQW is able to install hotkeys, which can be used to activate the EQW. These hotkeys are by default not installed. For information on how to install hotkeys see: 15.1 The Format menu [F1][C]



5.3 Exiting the EQW

The EQW is automatically exited when another application is activated. Even though one exits the EQW, the content of the editor is not lost. The content of the editor will be back if you start the EQW again with the Current method (see Starting the EQW). The following keys can be used to switch to another application from the EQW:

[Y=]	Switch to the Y= editor.
[WINDOW]	Switch to the Window editor
[GRAPH]	Switch to the Home screen.
[TblSet]	Switch to the Table Set editor.
[TABLE]	Switch to the Table editor.
[APPS]	The system application menu.
[2nd][APPS]	Switch to the previous application.
[DIAMOND][APPS]	The system Flash Application menu.
[MODE]	Change the current application in the mode window.
[QUIT]	Quit to Home.

5.3.1 Exiting and passing the content of the editor to Home.

Pressing [ESC] exits the editor and makes it pass its content to the entry-line of the Home application. The content will be inserted at the cursor position in the entry-line. The way it works is similar to Copy/Paste to the entry-line except it does not involve the Clipboard.

6. Entering expressions

Entering an expression in the EQW is fairly straightforward. However, there is one thing, which you must get used to, i.e. the method of entering parentheses. Only in very rare cases do you enter parentheses in the EQW. In fact, the EQW does not like parentheses and will flash an error message if you try to enter them.

In the EQW the cursor is the parentheses. In this program you would mark the expression in the EQW where you normally would insert parentheses and the EQW will then insert parentheses, if necessary. Every operation in the EQW is applied to the marked part of an expression. The EQW automatically marks different levels of sub-expressions when using the [UP] / [DOWN] cursor keys. Pressing [UP] marks the sub-expression above. Pressing [DOWN] marks the sub-expression below. This may sound a little cryptic, but in fact it is very simple.



6.1 Placing parenthesis using the cursor.

Entering: $a - b + c$

Keys	Result in EQW	Comments
		The black square is the placeholder. A placeholder indicates that a value is expected and not yet entered. The outlined box around the placeholder is the cursor. The cursor indicates where the next operation will be applied. Entering a value will replace the content enclosed by the cursor. The cursor is outlined when it is at the lowest level of an expression.
[a]		Entering a character will make the cursor change form to a left arrowhead. The arrowhead cursor indicates that you are in the character mode and the next character entered will be placed at the position of the arrowhead. Pressing [BACKSPACE] in the character mode deletes the character to the left of the arrowhead.
[-]		Pressing an operator will exit the character mode, add a new placeholder and move the cursor to it. The next argument can now be typed in.
[b]		
[UP]		Pressing [UP] exits the character mode and moves the cursor one level up. The cursor is now displayed as an inverse text. This indicates that it is no longer at the lowest level of the expression. The cursor as always indicates which part of the expression the next operation will be applied to. In this case to $(a - b)$ and that is what we want, because we want to enter $(a - b) + c$.
[+]		Pressing [+] adds a new placeholder where the value to add can be entered. The parentheses around $(a - b)$ are not shown, because they are implied by the order of operators. Parentheses are only shown in the EQW when they are a mathematical necessity.
[c]		The expression is now completed and can be returned to the entry line by pressing [ESC].

Entering: $a - (b + c)$

Keys	Result in EQW	Comments
[a] [-][b]		The first part is the same as in the previous example, just enter “ $a - b$ ”. Outlined and left arrow cursors indicate that the next operation applies to that element alone. This is what we want in this case, since we want to add ‘ c ’ to ‘ b ’.
[+]		Pressing [+] adds a new placeholder as in the previous example, but this time there also appear parentheses around b and the placeholder. The parentheses are shown because the operators do not imply the calculation order.
[c]		



Entering: $a - b/c + d$

Keys	Result in EQW	Comments
[a][−][b]	$a - b$	
[/][c]	$a - \frac{b}{c}$	This is not much different from the previous example. This time, however, it is the division's operator that is applied to 'b' instead of the addition operator.
[UP]	$a - \frac{b}{c}$	Pressing [UP] marks "b/c", but that is not what we want, because pressing [+] now will add 'd' to "b/c".
[UP]	$a - \frac{b}{c}$	Pressing [UP] again marks "a-b/c". Expressions can be divided to smaller units; each of them represents a cursor level. In this case the expression can be split up into the single elements a, b and c. They are at the lowest level. It can also be divided into two sub-expressions "b/c" and "a - (b/c)". They are at a higher level, because they contain more than one simple element. The first sub-expression above 'c' is "b/c". The next level above "b/c" is "a - (b/c)", because it includes both the sub-expressions "b/c" and the single element 'a'. If the expression "a - (b/c)" had been a part of another expression then there would have been further levels above it. Each time [UP] / [DOWN] is pressed the cursor moves one level up/down and mark the expression at that level.
[+][d]	$a - \frac{b}{c} + d$	Pressing [+] applies correctly the addition to the entire expression.

As it can be seen from above examples, does the use of cursor keys alter the way expressions are build and the purpose of parenthesis is replaced by marking sub-expressions.



6.2 Entering functions/commands/programs

Functions, as operators, are always applied to the marked part of an expression. A function can have one or more arguments. Optional arguments are added by marking the function and pressing [,]. Optional arguments are removed by marking the argument and pressing [BACKSPACE]. Pressing [RIGHT] / [LEFT] moves the cursor to the next/previous argument.

Example: $\int(\sin(2 \cdot t), t, 0, 1)$

Key	Result in EQW	Comments
[2][*][t]	$2 \cdot t$	Entering the argument to sin(). You could also insert sin() first and then the argument.
[UP]	$\sin(2 \cdot t)$	Marking the expression by pressing [UP] applies [SIN] to "2*t" and not only to t.
[SIN]	$\int \sin(2 \cdot t)$	After sin() is applied it is also included in the marked area. This is because sin() now has taken over the top level. Pressing [DOWN] would mark "2*t" again. Functions remain marked if no further arguments are required otherwise? The cursor jumps to the first missing argument.
[∫]	$\int \sin(2 \cdot t) dt$	Applying the integral makes three new placeholders appear. One for each argument the function requires.
[t]	$\int \sin(2 \cdot t) dt$	Pressing [RIGHT] makes the cursor jump to the next argument. Pressing [LEFT] makes the cursor jump to the previous argument.
[RIGHT]	$\int \sin(2 \cdot t) dt$	You do not need to fill out the limits of the integral, because they are optional arguments. Optional arguments can be deleted by pressing [BACKSPACE]. Deleting one of the limits of the integral removes both limits.
[0]	$\int_0 \sin(2 \cdot t) dt$	It may be a little confusing to press [RIGHT] to jump to the next argument when the argument is above or to the left. The [ENTER] key may be helpful in this case. Pressing [ENTER] when there are placeholders in the expression makes the cursor jump to the first placeholder in the expression. Entering a value and pressing [ENTER] makes the cursor jump to the next placeholder. Please note that when there are no more placeholders left [ENTER] will trigger the evaluation of the entered expression.
[RIGHT] [1]	$\int_0^1 \sin(2 \cdot t) dt$	Now the integral is completed and ready to be evaluated by pressing [ENTER] or to be returned to the entry line by pressing [ESC].



Example: $\lim(f(t), t, 0, -1)$

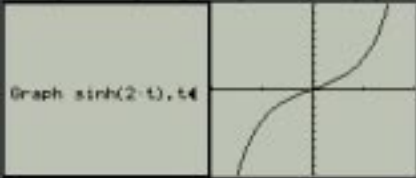
Key	Result in EQW	Comments
[f][\square]	f(\square)	Typing the function name and a starting parenthesis enters an undefined function. The starting parenthesis tells the program that it is a function and not a variable. You could type in the full function "f(t)" that is also accepted by the EQW.
[RIGHT]	f(\square)	Pressing a cursor key triggers the evaluation of the input. The EQW will recognise that an undefined function has been entered and provides the closing parenthesis and a placeholder.
[t][UP]	f(t)	The argument 't' is entered and the function is marked by [UP].
[F3][3]	lim f(t) $\square \rightarrow \square$	The limit function is selected from the EQW menu. It could also have been selected from the [MATH] or [CATALOG] menu. Functions selected from menus are also applied to the marked part of the expression.
[t][RIGHT][0]	lim f(t) $t \rightarrow 0$	The arguments are entered as in the previous examples.
[,]	lim f(t) $t \rightarrow 0^+$	The direction of a limit is an optional argument. Optional arguments are per definition added to a function by selecting the function and pressing [,]. But in most cases, it is enough just to press [,] from anywhere inside the function. The direction of a limit is a special case of an optional argument. Normally optional arguments are added as placeholders, but by limits it appears as a small plus. You cannot enter anything in the direction field, but you can toggle the direction by pressing [(-)].
[(-)]	lim f(t) $t \rightarrow 0^-$	The direction of a limit is toggled by pressing [(-)].



6.2.1 Typing functions/commands/program names

Functions, commands and program names can be typed instead of selecting them from menus.

Example: Graph $\sinh(2 \cdot t)$, t

Key	Result in EQW	Comments
[G][r][a][p][h]	Graph	Built-in commands and functions must be typed exactly as they appear in the [CATALOG] menu with uppercase and lowercase letters.
[SPACE]	Graph	Commands must be terminated by a space.
[RIGHT]	Graph	Pressing a cursor key inserts the commands with an appropriate number of placeholders.
[2][*][t]	Graph 2·t	Typing argument to sinh().
[UP]	Graph 2·t	Marking the argument to sinh(). Typed in functions/commands/program names are as always applied to marked part of expression.
[s][i][n][h]	Graph sinh	Typing in the function. Notice that the argument “2·t” has disappeared. The program doesn’t know that you are entering a function and assumes that you are just replacing the expression. When the EQW recognises that a function or command has been entered it will return the argument.
[(]	Graph sinh(Opening parenthesis must terminate functions and program names. However, the terminating parenthesis is not necessary by built-in functions.
[UP]	Graph sinh(2·t)	Marking the expression, because we need to add the variable to graph. Notice that the “2·t” has reappeared as the argument to sinh().
[,]	Graph sinh(2·t),	Pressing [,] adds an optional argument to the marked command.
[t]	Graph sinh(2·t),t	The expression is now completed and can be executed by pressing [ENTER] or returned to the entry line by pressing [ESC].
[ENTER]		



6.3 Entering a Matrix

Pressing \square inserts a 1x1 matrix around the marked part of expression. A matrix in the EQW can be marked at three levels:

1. The entire matrix.
2. A row in the matrix.
3. An item in a row.

A column is added to the end of a matrix by marking a row and pressing [,]. A row is added to the bottom of a matrix by marking a row and pressing [;] or by marking the entire matrix and pressing [;].

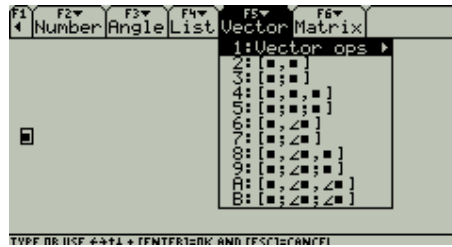
Entering: [a,b;c,d]

Key	Result in EQW	Comments
	\square	Starting with a blank editor this time. It is not necessary to clear the editor, because a matrix can be inserted anywhere in an expression.
[[]]	\square	The matrix is inserted.
[a]	[a]	Typing first variable.
[,]	[a \square]	Adding a column. The rules above are the general rules for altering the size. In most cases you do not need to mark a row/matrix before adding rows/columns. If the cursor is not marking a function or is inside a function then pressing [,]/[;] from any place inside the matrix will add a column/row.
[b]	[a b]	Typing second variable.
[;]	\square \square	Adding a row.
[c]	[a b c \square]	Typing third variable.
[RIGHT]	[a b c \square]	Pressing [LEFT]/[RIGHT] moves the cursor to previous/next item in a matrix.
[d]	[a b c d]	Typing fourth variable and the matrix is completed.

6.4 Entering vectors



The only way to enter polar/cylindrical/spherical vectors in the EQW is by selecting them from the Vector menu. Pressing [F7][F5] activates the Vector menu.



The Vector menu contains row and column vectors in four different formats. The formats are rectangular, polar, cylindrical and spherical. Selecting a vector in the menu replaces the marked expression with vector.

Entering: [5,∠60°,∠45°]

Keys	Result in EQW	Comments
[F7][F5][A]		Inserting a spherical vector.
[ENTER]		Pressing [ENTER] to move the cursor to the first placeholder.
[5]		Typing the first co-ordinate.
[RIGHT]		Moving the cursor to the next co-ordinate.
[6][0][°]		Entering the second co-ordinate.
[RIGHT]		Moving cursor to the last co-ordinate.
[4][5][°]		Entering the third co-ordinate.

6.5 Entering a list

Pressing [[]] inserts a one element list around the marked part of an expression. The list can be marked at two levels:

1. The entire list.
2. An item in the list.

An item is added to the end of a list by marking the entire list and pressing [,].

Entering: {a, b, c}

Key	Result in EQW	Comments
		Starting with a blank again. It is not necessary to clear the editor, because a list can be inserted anywhere in the expression.
[[]]		The list is inserted.
[a]		Typing first variable.



[,]	{a }>	Adding an item to the list. The rules above are the general rules for altering the size. In most cases you do not need to mark the list before adding items. If the cursor is not marking a function or is inside a function then pressing [,] from any place inside the list will add an item to the end of list.
[b]	{a b<<}	Typing second variable.
[,]	{a b }>	Adding a new item to the list.
[c]	{a b c<<}	Typing third variable and the list is completed.

6.6 Entering a string.

The EQW is not suitable for editing string, but it is capable of doing it. A dedicated text editor is more suitable for editing text.

Pressing [“] inserts a list at cursor position. If the cursor marks an expression then the expression is converted to a string. A list can be marked at two levels:

1. The entire string.
2. The text in the string

Entering: “t+cos(t)”

Key	Result in EQW	Comments
	{ }	Starting with a blank again. It is not necessary to clear the editor, because a string can be inserted anywhere in an expression.
[“]	“{ }”	The string is inserted. Deleting the placeholder inside the quotation marks using [BACKSPACE] creates an empty string.
[t]	“t<<”	Typing the first character.
[+]	“t+<<”	Pressing an operator or function keys inside a string inserts text instead of performing an operation.
[COS]	“t+cos(<<”	Inserting “cos(“.
[t)]]	“t+cos(t)<<”	Typing the last two characters and the string is completed.



6.7 Inserting subscription

Pressing [DIAMOND][[]] applies subscription to the marked part of an expression.

Entering: a[3,2]

Keys	Result in EQW	Comments
		Starting with a blank again. It is not necessary to clear the editor, because a subscription can be inserted anywhere in an expression.
[a]	a	Typing the variable. The arrowhead indicates that operations will be applied to this single object alone.
[DIAMOND][[]]	a	Applying subscription to variable 'a'.
[3]	a[3	Typing the first index.
[,]	a[3,	The second index is an optional argument to subscription, so it is added by pressing [,].
[2]	a[3,2	Typing the second index and the expression is completed.

6.8 Entering Degree Minutes Seconds (DMS)

Keys	Result in EQW	Comments
[4][5][°]	45°	Enter degrees.
[,]	45°'	Press [,] to add the optional argument Minutes.
[2][2]	45°22'	Enter minutes.
[,]	45°22'"	Press [,] to add the optional argument Seconds.
[1][5]	45°22'15"	Enter seconds.

7. Moving cursor

The cursor can be moved in two different Moving Modes.

1. Moving between conjugate sub-expressions.
2. Jumping between single elements of an expression (numbers, variables, fractions and placeholders).

The first method is typically used when an expression is entered/modified. The second method may often help when moving the cursor to a certain place in a complex expression or when one just wants to change some values. The EQW automatically switches between these moving methods, therefore, it is usually not necessary to control the moving mode.



7.1 Controlling the Moving Mode:

- [DOWN] If the cursor is outlined then the Moving Mode is switched to second else the cursor is moved one level down.
- [UP] Moves the cursor one level up and the Moving Mode is switched to the first.
- [2nd][LEFT] Moves the cursor to the beginning of the line and the Moving Mode is switched to second. That will say the cursor is moved to the first simple element found in the expression.
- [2nd][RIGHT] Moves the cursor to the end of the line and the Moving Mode is switched to second. That will say the cursor is moved to last simple element in the expression.
- [2nd][DOWN] Moves the cursor to the first simple element found inside the marked expression and the Moving Mode is switched to second.
- [2nd][UP] Moves the cursor to the highest level (marks the entire expression) and the Moving Mode is switched to first.

	Moving Mode 1	Moving Mode 2
[LEFT]	If there is a sub-expression to the left then the cursor is moved to sub-expression. If there is no sub-expression to the left the cursor is moved up one expression level.	Moves cursor to the next single element to the left. The cursor stops at the first single element in the expression.
[RIGHT]	If there is a sub-expression to the right then the cursor is moved to sub-expression. If there is no sub-expression to the right the cursor is moved up one expression level.	Moves the cursor to the next single element to the right. The cursor stops at the last single element in the expression.



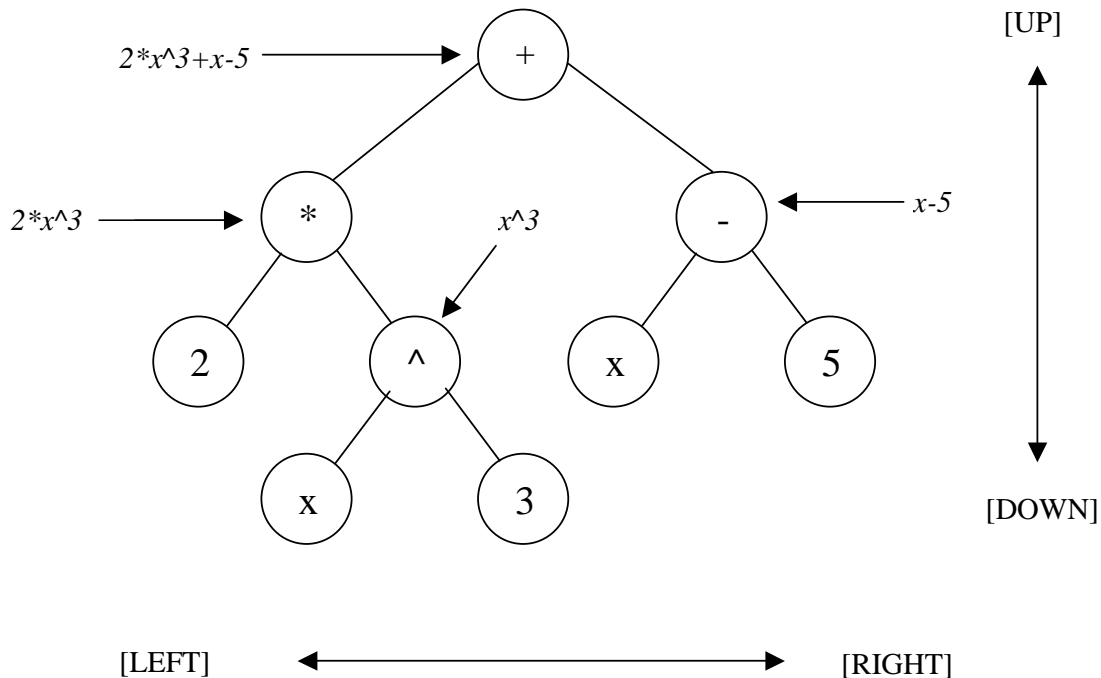
8. Editing an expression

8.1 Marking sub-expressions

As you probably already know are almost all operations applied to the marked part of an expression. The way an expression is marked can be visualised by a tree where each node represents a cursor position. The marked area is always the tree seen below a node. When the cursor is moved it jumps between the nodes in the tree. The [UP] / [DOWN] cursor keys moves vertical in the tree. The [LEFT] / [RIGHT] cursor keys moves horizontal in the tree.

If $2*x^3+x-5$ is entered with following key sequence then the movement tree will look as below:

[2][*][x][^][3]
 [UP][UP]
 [+]
 [x][-][5]



Now above expression in the EQW

Key	Result in EQW	Comments
	$2 \cdot x^3 + x - 5$	The expression has been entered. The cursor is at the top node of the movement tree and the entire expression is marked.

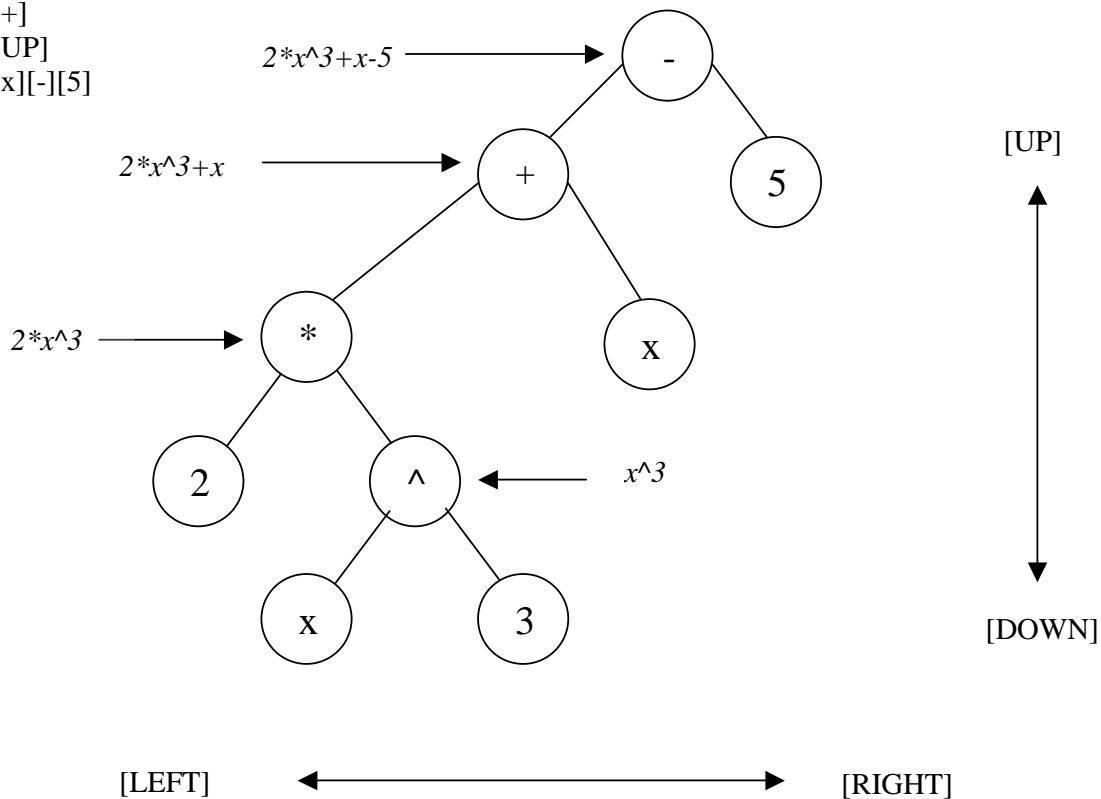


[DOWN]	$2 \cdot x^3 + x - 5$	The cursor is moved down to the second node from the top in the left side of the tree. Pressing [DOWN] moves always the cursor along the path to the left in a ramification.
[DOWN]	$2 \cdot x^3 + x - 5$	Pressing [DOWN] one more time moves the cursor to the bottom left node in the movement tree. The cursor is now at the end of this branch and can not be moved further down. This can be seen from that the cursor shape has changed to an outlined box.
[RIGHT]	$2 \cdot x^3 + x - 5$	Pressing [RIGHT] marks the second factor.
[RIGHT]	$2 \cdot x^3 + x - 5$	Pressing [RIGHT] again marks the next term.
[UP]	$2 \cdot x^3 + x - 5$	Pressing [UP] moves the cursor to the node above the x-node to the right. As it can be seen in the movement tree is the node above 'x' the subtract operator.

The shape of a movement tree and the default marking of sub-expressions depends on how an expression is entered.

If $2 \cdot x^3 + x - 5$ from previous example is entered with following key sequence then the movement tree will look as below:

[2][*][x][^][3]
 [UP][UP]
 [+]
 [UP]
 [x][-][5]





8.2.1 Deleting functions leaving the argument: [DEL]

Key	Result in EQW	Comments
	$\sin(2 \cdot t) + (\tan(t^2))^2$	Changing tan() to cos().
[RIGHT] [DOWN]	$\sin(2 \cdot t) + (\tan(t^2))^2$	Marking tan()
[DEL]	$\sin(2 \cdot t) + (t^2)^2$	Deleting tan(). Notice that the argument "t^2" remains.
[COS]	$\sin(2 \cdot t) + (\cos(t^2))^2$	Inserting cos() in the place of tan().

8.2.2 Deleting optional arguments: [BACKSPACE]

Key	Result in EQW	Comments
	$\int_a^b \tan(t) dt$	Deleting optional arguments by marking argument and pressing [BACKSPACE].
[BACKSPACE]	$\int \tan(t) dt$	By integrals both limits are deleted at the same time.

8.2.3 Deleting single characters: [BACKSPACE]

Key	Result in EQW	Comments
	$aa + bk + \square$	Changing "bk" to "bb"
[LEFT]	$aa + \square + cc$	Marking "bk". The outlined cursor indicates that the item is a single element. This means that pressing [BACKSPACE] will activate character-editing mode.
[BACKSPACE]	$aa + b\blacktriangleleft + cc$	The black arrowhead indicates that you are in character-editing mode.
[BACKSPACE]	$aa + b\blacktriangleleft + cc$	In the character-editing mode [BACKSPACE] deletes the character to the left of the arrowhead.
[b]	$aa + bb\blacktriangleleft + cc$	The correction is completed.

8.2.4 Deleting parts of an expression: [BACKSPACE]

Key	Result in EQW	Comments
	$\frac{8 \cdot t}{t^2 + 1} + \sin(5 \cdot t)$	Removing the denominator completely. The denominator is marked. [BACKSPACE] and [CLEAR] will both delete a marked expression and preserve the placeholder.
[BACKSPACE]	$\frac{8 \cdot t}{\square} + \sin(5 \cdot t)$	Only [BACKSPACE] can delete a placeholder alone.
[BACKSPACE]	$8 \cdot t + \sin(5 \cdot t)$	Deleting a placeholder also removes the operator.



8.2.5 Deleting rows and columns of a matrix [BACKSPACE]

Key	Result in EQW	Comments
	<pre> a b c d e f g h i </pre>	Before a row can be deleted it must be marked.
[UP]	<pre> a b c d e f g h i </pre>	Pressing [UP] when an element in a matrix is marked or [DOWN] when the entire matrix is marked marks a row.
[RIGHT]	<pre> a b c d e f g h i </pre>	The cursor is moved to the previous/next row by pressing [LEFT] / [RIGHT].
[BACKSPACE]	<pre> a b c g h i </pre>	Pressing [BACKSPACE] deletes the marked row.
[DOWN][RIGHT]	<pre> a b c g h i </pre>	If one wants to delete the second column, first, the cursor is moved to an element in the second column.
[CLEAR]	<pre> a b c g h i </pre>	Then the element is cleared. When there is a placeholder in a column, the column can be deleted by deleting the placeholder.
[BACKSPACE]	<pre> a b g i </pre>	Deleting the column.

8.3 [Copy], [Cut] and [Paste]

You can copy/paste expressions between the EQW and other applications. Expressions can also be copy/pasted inside the EQW. These functions can be used to reshuffle an expression.

Keys	Description
[CUT] or [F1][6]	Cuts the marked part of an expression only preserving the placeholder.
[COPY] or [F1][7]	Copies the marked part of an expression.
[PASTE] or [F1][8]	Replaces the marked part of an expression with a previous copied or cut expression.

Note: only valid expressions can be pasted into the EQW.



8.4 Undoing operations [INS]

The EQW can undo any operation except single-character editing (i.e. the cursor is an arrowhead). Up to ten operations can be undone.

Example:

Key	Result in EQW	Comments
	$8 \cdot t + \sin(5 \cdot t)$	Regretting the deletion of the denominator in the previous example I can undo the operation.
[INS]	$\frac{8 \cdot t}{\square} + \sin(5 \cdot t)$	The placeholder appears again.
[INS]	$\frac{8 \cdot t}{t^2 + 1} + \sin(5 \cdot t)$	Pressing [INS] one more time has undone the deletion in the previous example.

9. Evaluating expressions

The EQW has three different key-functions for evaluating an expression inside the EQW and a menu-function for evaluating expressions from the Home environment.

Keys	Description
[ENTER]	Evaluates the entire expression. The format of the result depends on the current mode-settings of the calculator. This is the same as marking the entire expression and pressing [ENTER]. If there are placeholders in the expression, the cursor is moved to the first placeholder and no evaluation takes place. This can be used to let the program move the cursor and just fill in the empty fields.
[ENTRY]	Evaluates the marked part of an expression. The format of the result depends on the current mode-settings of the calculator. If there are placeholders in the marked expression the cursor is moved to the first placeholder and no evaluation takes place.
[DIAMOND][ENTER]	Evaluates the marked part of an expression in an approximate mode. If there are placeholders in the marked expression the approx() function is inserted and no evaluation takes place.
[F1][5]	Transfers the marked part of an expression to the Home environment and executes the expression from there. The result will not be returned to the EQW, but can be retrieved by evaluating ans(1) in the EQW.



Evaluating: Solve($x^2+2x-1=0,x$) using [ENTER] and [DIAMOND][ENTER]

Keys	Result in EQW	Comments
	$x^2 + 2 \cdot x - 1$	The expression is entered, but we still need to create an equation and insert the solve() function.
[=]	$x^2 + 2 \cdot x - 1 =$	Applying the equal operator to the marked expression to create an equation.
[0]	$x^2 + 2 \cdot x - 1 = 0$	Entering the right side of the equation.
[UP]	$x^2 + 2 \cdot x - 1 = 0$	Marking the equation, so that solve() can be applied to the equation.
[F2][1]	$\text{solve}(x^2 + 2 \cdot x - 1 = 0, \text{ })$	Selecting solve() from the Algebra menu. Notice that the cursor is moved to the placeholder for a second argument, so it is ready for entering the argument.
[x]	$\text{solve}(x^2 + 2 \cdot x - 1 = 0, x)$	Entering the variable to solve for.
[ENTER]	$x = -(\sqrt{2} + 1) \text{ or } x = \sqrt{2} - 1$	If you want to evaluate the entire expression [ENTER] will do the job. When the evaluation is completed the result appears in the EQW ready for further manipulations. Let us now find the approximate value of the first result and preserve the second result exact.
[DOWN]	$x = -(\sqrt{2} + 1) \text{ or } x = \sqrt{2} - 1$	Marking the first result.
[DIAMOND] [ENTER]	$x = -2.41421 \text{ or } x = \sqrt{2} - 1$	Evaluating the marked expression in an approximate mode.



1. Expanding “(t+5)^2+2” in the denominator of: (t-3)^2/((t+5)^2+2)^2 using [ENTRY]
2. Transferring marked expression to Home.

Keys	Result in EQW	Comments
		Marking the sub-expression. Pressing [ENTRY] evaluate/simplifies the marked part of an expression in using current mode-settings.
[ENTRY]		The sub-expression is simplified.
[F1][5]		The content of the EQW is not changed by this operation. Only the content of Home is changed.
		This picture shows the result of this operation seen from Home. The marked expression is transferred to Home and executed from there in the same way as when you normally enter expressions and evaluate them, but you remain in the EQW.

10. Opening/Recalling variables

Variables opened using one of the EQW opening methods are not simplified before being displayed in the EQW. Variables can be opened using different methods:

Opening method	Result
Via the Open method in the Flash Application menu.	Variables opened for editing are updated. The editor is cleared. The selected variable is either *recalled or **opened for editing.
Via the Open method in the EQW Tools menu.	Variables opened for editing are updated. The editor is cleared. The selected variable is either *recalled or **opened for editing.
Via the keyboard function [RCL]	Replaces the marked expression with the content of a variable.
Using the API Open method.	Variables opened for editing are updated. The editor is cleared. **Variable is opened for editing.
Using the API Input method.	Variables opened for editing are updated. The editor is cleared. **Variable is opened for editing.

* Variables recalled are just copied into the EQW.

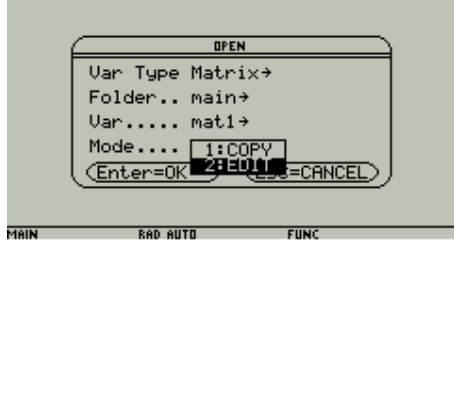
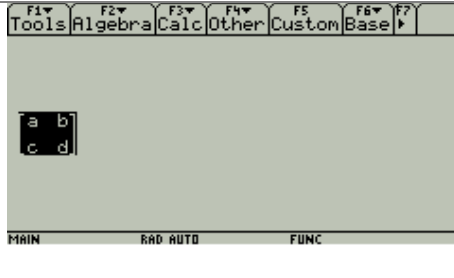
** Variables opened for editing are automatically updated with the content of the EQW by exit of the EQW.



Example: Opening a variable for editing via the Flash Application menu.

Keys	Result	Comments
		Storing a matrix into a variable, so we have a variable to work with. Now let us edit the “mat1” variable in the EQW.
[DIAMOND] [APPS]		The Flash Application menu pops up. In this example the menu only contains the Equation Writer, but it might contain more items if there are more Flash Applications installed.
[RIGHT]		Press [RIGHT] to activate the Equation Writer Open menu. The Open menu gives you three options for opening the EQW.
		Current Lets you continue editing of an expression in the EQW.
		Open Lets you pick a variable to edit.
		New Opens a blank editor.
[DOWN] [ENTER]		<p>Selecting the Open method. The EQW Open dialog is displayed.</p> <p><i>Var Type</i> is a submenu where the type of variable to open is selected.</p> <p><i>Folder</i> is a submenu where the folder to pick the variable in is selected.</p> <p><i>Var</i> is a submenu containing all the variables of the selected type in the selected folder.</p> <p><i>Mode</i> is a submenu where the opening method is selected.</p>
[RIGHT] [DOWN] [DOWN]		<p>The EQW can edit four types of variables:</p> <ol style="list-style-type: none"> Expressions (an example of an expression could be (a+b)/c). Lists Matrix Strings <p>In this case, we want to edit a matrix, so we select Matrix in the menu.</p>



<p>[ENTER] [DOWN] [DOWN] [DOWN]</p>		<p>When Matrix is selected the names of all variables in the selected folder containing matrix appears in the <i>Var</i> submenu. In this case, the folder main and the variable mat1 are automatically chosen, so we do not need to change them, but the opening mode needs to be changed. The Opening Mode can either be COPY or EDIT. If COPY is selected then the EQW is opened with a copy of the content of the variable. The variable will remain unchanged even though you make changes to the content in the EQW. If EDIT is selected then the EQW is opened with the content of the variable. The variable will be updated by the EQW so it reflects the changes made in the EQW.</p>
<p>[ENTER] [ENTER]</p>		<p>The EQW is opened displaying the content of the variable</p>



11. Saving expressions

The contents of the EQW can be stored to a variable in three ways:

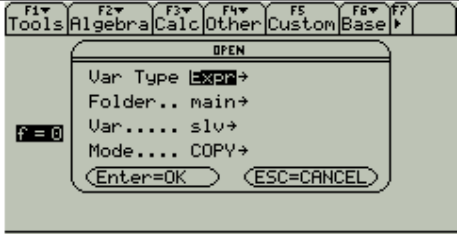
1. Using the [STO] operator in the EQW and evaluating expressions. The [STO] operator evaluates expressions and the result is stored into a variable.
2. Selecting “Save Copy As...” in the Tools menu. The “Save Copy As...” function stores the contents of the EQW unevaluated into a variable.
3. Opening a variable for editing via the Tools menu. Changes made to variables opened this way are automatically saved when the EQW is exited, another variable is opened or a new expression is created via New. The expression is not evaluated before saving.

Expressions stored by any of these methods can be retrieved unevaluated by use of the Open function in the Tools menu or Flash Application menu. The API Open method or the [RCL] function can also be used to recall an unevaluated variable.

Using method 2 or 3 to store into a variable give you total control over what is stored and how it is stored. It is possible via method 2 and 3 to store expressions, which are not valid outside the EQW. An example of a non-valid expression could be a matrix containing a list or another matrix. The EQW does not check whether or not an expression is valid outside the EQW before it is saved, because expressions stored by the EQW are always valid for the EQW. This means that even though an expression is invalid outside the EQW it can always be opened and edited by the EQW and potential errors corrected.

Expressions stored by method 2 and 3 are first evaluated when used. This can be used to create variable with special properties.

11.1 Example: Differences between STO and “Save Copy As..”.

Keys	Result in EQW	Comments
	<code>solve(f = 0, t) → slv</code>	The differences between the storing methods are best shown by a comparison. Let us study the ordinary method of storing into a variable first.
[ENTER]	<code>f = 0</code>	Evaluating the expression. Now let us open the variable to see what the store operator really has stored into the variable.
[F1][1]		There pops up the Open window and the variable-type, folder, name and opening mode can be selected. In this case we just want to see what is stored into the variable “slv”, so a copy is fine.
[ENTER]	<code>f = 0</code>	As suspected the variable only contains “f=0”.



		<p>Now let us try to do the same using the EQW “Save Copy As.” method. It is not necessary to mark the function, because “Save Copy As.” always stores the entire expression.</p>
[F1][2]		<p>This opens the “SAVE COPY AS” window. The Type-field shows which type of variable the expression will be saved as. The Folder-field shows in which folder the variable will be stored. The name of the variable to store the expression into must be typed in the Variable field.</p>
[DOWN] [s][1][v]		<p>Entering the variable-name “slv2”.</p>
[ENTER] [ENTER]		<p>Pressing [ENTER] twice to verify the input and the window is closed. The expression is now stored into the variable “slv2”. The contents of the EQW remain unchanged by this operation. Let us now open the variable to see what really was stored this time.</p>
[CLEAR]		<p>Clearing the EQW just to make sure that it is really the contents of the variable we get (it is not necessary, because the editor is automatically cleared when a variable is opened).</p>
[F1][1] [ENTER]		<p>Opening the variable “slv2” via the Tools menu. As expected, the variable does contain the saved expression. Let us now try to use the created variables from the Entry line.</p>
		<p>If the variables are evaluated from the Entry line then they appear to be identical.</p>



	<p>It is first by substitution of the variable 'f' that the difference becomes visible. The reason for this difference is easy to find by looking at the contents of the variables. Slv contains "f=0" Slv2 contains "solve(f=0,t)" What happens is that the variable 'f' is replaced by "t^2-1" in both expressions and the resulting expressions are evaluated.</p>
--	---

11.2 Viewing the content of a variable.

Due to the EQW being able to store unevaluated expressions the real content of a variable is not always revealed by evaluation of the variable (see 11.1). The content of a variable can be viewed in the VAR-LINK menu or by opening the variable in the EQW.

Press [F6] in the VAR-LINK menu to view the content of a variable.

12. Creating forms

You can create/use forms within the EQW. A form can be any kind of expression where one or more placeholders are not filled in. A form can either be a stored expression or a string containing an expression. You create a form in exactly same way as a normal expression. The only difference is that you do not fill in all placeholders. The placeholder you preserve can then be filled in later when the form is used.

The Vector menu [F7][F2] is an example of how forms can be used in the EQW:

All of the items in the Vector menu are forms.



12.1 Using stored expressions as forms

A form is created by:

1. Entering the expression in the EQW
2. Storing the expression into a variable using Save Copy As in the Tools menu.

This type of form is recalled by opening the variable using Open in the Tools/Flash Application menu. The API Open/Input methods can also be used to open the forms. A form can be inserted into the EQW by recalling the variable using [RCL].

12.2 Using strings as forms

A form is created by:

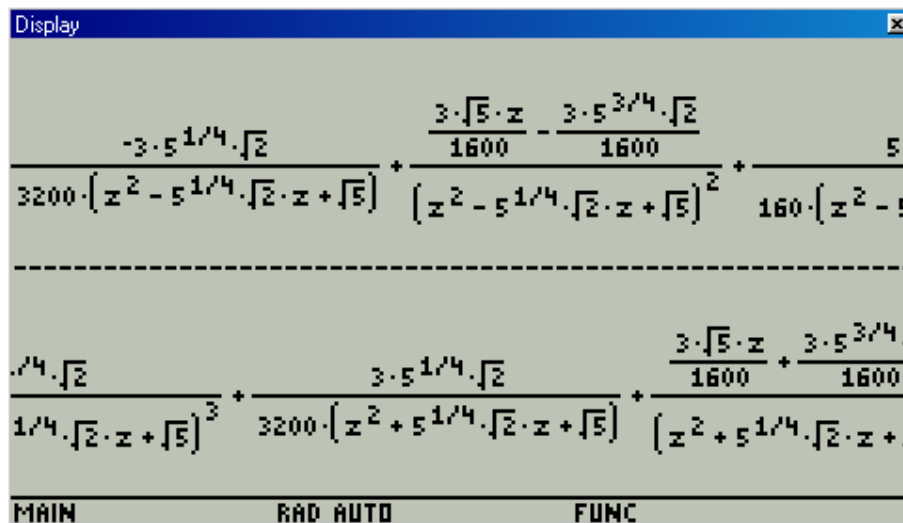
1. Entering the expression in the EQW
2. Mark the expression
3. Press ["] to convert the expression to a string

The result of this is a string-form. String-forms can only be inserted into the EQW by use of the API Insert method. This type of form is very suitable for use in menus.

13. The View Screen [F8]

The View Screen displays as much as possible of the current expression in the EQW on a single screen. The View Screen will always use the entire screen area for displaying. Long expressions are wrapped and displayed on multiple lines.

The View Screen is activated by pressing [F8] or by selecting View in the Tools menu.





Key functions in the View Screen:

Key	Description
[ESC]	Return to the editor.
[LEFT]	Scroll expression to the left.
[RIGHT]	Scroll expression to the right.
[DOWN]	Scroll expression down. If the View Screen displays more than one line then pressing [DOWN] will scroll one line down.
[UP]	Scroll expression up. If the View Screen displays more than one line then pressing [UP] will scroll one line up.
[2nd][LEFT]	Jump to the start of expression.
[2nd][RIGHT]	Jump to the end of expression.
[DIAMOND][)]	Decreases the font size of the displayed expression. Large text ⇒ Medium text Medium text ⇒ Small text Small text ⇒ Large text The amount of text, which can be displayed at a time on the View Screen, depends highly on the selected text size. The text size of the View Screen is independent of the text size in the editor. Selecting Format in the Tools menu and changing the “View Text Size” is an alternative way to set the text size of the View Screen.

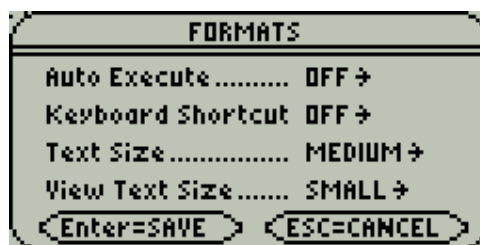
14. Local and global variables

The TI OS defines two types of variables Local and Global. Local variables are variables declared with the Local statement, all other variables are global variables. When a variable is entered in the EQW by default it is interpreted as a global variable.

Local variables are often used in programs and functions. The OS does not allow undefined local variables in expressions, but the EQW does. A variable can be converted to a local/global in the EQW by marking a variable and pressing [DIAMOND][(-)]. If the marked variable is a global one, then it is converted to a local. If it is a local variable then it is converted to a global. This function makes it possible to create stored expressions containing undefined local variables for use in programs.

15. Customising the EQW

15.1 The Format menu [F1][C]





15.1.1 Auto Execute

Auto Execute control the behaviour of the menus in the EQW. If Auto Execute is set to OFF then picking a function/command in a menu will lead to that a chosen function/command is inserted in the expression. If Auto Execute is set to ON then the function/command is evaluated if possible and the result are inserted. It is only by build-in functions and commands an evaluation is attempted and only if the function/command can be evaluated with one argument. Auto Execute is default set to OFF.

Example: Behaviour of the function factor()

Keys	Result in EQW	Comments
	$\frac{1728}{11}$	Starting with a random fraction. Auto Execute is set to OFF.
[F2][2]	$\text{factor}\left(\frac{1728}{11}\right)$	Selecting the factor function from the Algebra menu.
	$\frac{1728}{11}$	Starting with a random fraction again. Auto Execute is set to ON.
[F2][2]	$2^6 \cdot 3^3$ $\frac{\quad}{11}$	Selecting again the factor function from the Algebra menu. As it can be seen from the picture, this time the result of the factorisation is inserted and not the function.

15.1.2 Keyboard Shortcut

The EQW is able to install hotkeys, which let you access the EQW by a single key-press from within other applications. Setting Keyboard Shortcut to ON in the Format menu will install hotkeys in following applications. Keyboard Shortcut is default set to OFF.

Application Name
Home
Y= Editor
Graph
Table
Data/Matrix
Program Editor
Text Editor
Numeric Solver

IMPORTANT: As long as Keyboard Shortcut is set to ON the EQW or Flash applications installed before the EQW can not be uninstalled. This means that you have to set Keyboard Shortcut to OFF before you can delete the EQW or other Flash applications installed before the EQW. Flash Applications installed after the EQW are not influenced by the Keyboard Shortcut.



The hotkeys defined by the EQW are two key-combinations, which are not used by the system.

[DIAMOND][(-)] Copies the marked expression to Clipboard. Opens the EQW for editing of a new expression. Pasting the content of the clipboard into the EQW. If nothing is marked then the EQW will be started with the current content of the Clipboard. If the marked is not a valid expression or the clipboard contains an invalid expression then this command will result in an error in the EQW.

[DIAMOND][CLEAR] Continues editing of an open expression/variable. This is the same as selecting “Current” in the Flash Application menu.

Example: Passing an entry from the Home history area into the EQW.

Keys	Result in EQW	Comments
		This picture shows the Home screen. The cursor is moved to the second answer in the History area.
[DIAMOND][(-)]		The EQW is displayed and the marked expression from the Home history area has been pasted into the editor.

15.1.3 Text Size/View Text Size

Text Size/View Text Size controls the font size used in the editor/View Screen. The text size can be set to SMALL or MEDIUM or LARGE. Pressing [DIAMOND][)] can also be used to change the text size.

SMALL = 4x6 pixel fonts (variable size font)

MEDUM = 6x8 pixel fonts (fixed size font)

LARGE = 8x10 pixel fonts (fixed size font)



15.2 Using the standard Custom menus

Most standard Custom menus can be used with the EQW. Though attempts to insert an invalid expression via a Custom menu will result in an error in the EQW.

The built-in Custom menu has two examples of menu items, which are not valid in the EQW. These are:

[F2][6] "Define f(x) ="
[F3][4] "solve(and , {x,y})"

[CUSTOM] Turns the active Custom menu on/off.

To find out more about standard Custom menus look up "Custom menu" in your calculator manual.

15.3 EQW keyboard programs

Ten key-combinations in the EQW are reserved for customising. These keys do nothing else but execute programs with certain names if they exist. All EQW keyboard programs must be placed in the MAIN folder. There are basically no rules for what type of programs can be used as keyboard programs.

Key	Program names
[DIAMOND][1] – [DIAMOND][9]	EQWPrgm1() – EQWPrgm9()
[F5]	EQWUser()

The API program "EQW.Insert" (see 17.4EQW.Insert) can be used in connection with the keyboard programs to create functions and menus in the EQW.

Examples of creating programs which work with the EQW:

1. Defining [DIAMOND][2] as log() key. The program must be placed in the main folder.

```
:eqwprgm2()
:Prgm
:eqw.insert("log()")
:EndPrgm
```

2. Defining [DIAMOND][3] as a popup menu with active commands. The program must be placed in the main folder. An active command is not inserted in the EQW. Each menu item will perform an operation on the marked part of the expression in the EQW and replace marked with the result.

```
:eqwprgm2()
:Prgm
: Local var
: 0->var
: PopUp {"cSolve x","cZeros x", "nSolve x","expand x","factor x"},var
: if var=1
: eqw.insert(cSolve(eqw,x))
: if var=2
```



```

: eqw.insert(cZeros(eqw,x))
: if var=3
: eqw.insert(nSolve(eqw,x))
: if var=4
: eqw.insert(expand(eqw,x))
: if var=5
: eqw.insert(factor(eqw,x))
:EndPrgm

```

3. A simple example of an EQW custom menu. Pressing [F5] activates the menu. The program must be placed in the main folder. The program creates a Toolbar with one menu. Picking one of the units in the menu will multiply the unit to the marked expression.

```

:eqwuser()
Prgm
Toolbar
Title "Units"
Item "m/s",u1
Item "m/s^2",u2
Item "ft/s^2",u3
EndTBar
Return
Lbl u1
Eqw.insert(eqw*_m/_s)
Return
Lbl u2
Eqw.insert(eqw*_m/_s^2)
Return
Lbl u3
Eqw.insert(eqw*ft/_s^2)
Return
EndPrgm

```

16. EQW keys and functions

The EQW support all key-function normally available so the Key Map found in the calculator manual is also valid for the EQW. There are a few keys, however, which have a special meaning in connection with the EQW; these are the [INS] and [ENTRY] keys. All other keys have the same or similar function as normal.

16.1 Cursor movements

Key	Description
[UP]/[DOWN]	Move one level up/down in the expression
[LEFT]/[RIGHT]	Move to the left/right in the expression
[2nd][UP]	Move to the top level of the expression (mark everything).
[2nd][DOWN]	Move to the lowest level of the marked (sub-)expression.
[2nd][LEFT] / [2nd][RIGHT]	Move to the beginning/end of expression.



16.2 Scroll expression

Key	Description
[DIAMOND][LEFT] / [DIAMOND][RIGHT]	Scroll expression left/right
[DIAMOND][UP] / [DIAMOND][DOWN]	Scroll expression up/down.

16.3 Expression evaluation

Key	Description
[ENTER]	Evaluate entire expression.
[ENTRY]	Evaluate marked (sub-) expression.
[DIAMOND][ENTER]	Evaluate marked (sub-) expression in approx. Mode.

16.4 Expression editing

Key	Description
[BACKSPACE]	<p>If cursor is inverse Delete marked expression</p> <p>If cursor is outlined Enter character editing mode</p> <p>If cursor is at left arrowhead Delete character to the left</p> <p>If placeholder is an optional argument to a function Delete optional argument</p> <p>If placeholder is the only argument to a function Delete placeholder and function</p> <p>If placeholder is an argument to an operator Delete placeholder and operator</p> <p>If placeholder is in column of a matrix Delete column</p> <p>If marked is a row of matrix Delete row</p> <p>Or with other words: [BACKSPACE] will delete almost anything.</p>
[DEL]	Deletes marked function and leave the argument.
[CLEAR]	Deletes the marked expression and leave placeholder.
[DIAMOND][CLEAR]	Deletes the entire expression.
[,]	Add optional argument to marked function. Add item to marked list. Add column to marked row of a matrix. Add row to marked matrix.
[SHIFT][UP]	<p>Expressions: Swap marked expression with the expression to the left.</p> <p>Functions/commands/lists/matrix: Swap marked argument with the argument to the left.</p>
[SHIFT][DOWN]	<p>Expressions: Swap marked expression with the expression to the right.</p> <p>Functions/commands/lists/matrix: Swap marked argument with the argument to the right.</p>
[INS]	Undo last operation (up to ten operations).
[COPY]	Copy marked expression.
[CUT]	Cut marked expression
[PASTE]	Paste marked expression



16.5 Marking expression

[SHIFT][LEFT]	Extend marked area to include the expression to the left.
[SHIFT][RIGHT]	Extend marked area to include the expression to the right.

16.6 Operators

Key	Description
[STO]	Insert store operator
[+], [-], [*], [/], [^], [√]	Insert arithmetic operator
[=], [≠], [<], [>], [≤], [≥]	Conditional operators
[∠], [>], [[]]	Angel, unit conversion and with operators
[x ⁻¹]	Insert reciprocal (TI-92+ only)
[°], [!]	Degree, factorial operators
[′], [-]	Differential and negation operator.

16.7 Functions

Key	Description
[SIN], [COS], [TAN], [SIN ⁻¹], [COS ⁻¹], [TAN ⁻¹]	Trigonometric functions
[LN], [e ^x]	Logarithm functions
[∫], [d]	Integral and differentiation
[Σ]	Summation (TI-92+ only)
[ANS]	ans()

16.8 Matrix

Key	Description
[[]]	Insert matrices
[.]	Add column to marked matrix.
[;]	Add row to marked matrix
[DIAMOND][[]]	Add subscription to marked matrix.
[BACKSPACE]	If placeholder marked Delete column If row marked Delete row
[SHIFT][UP]	If matrix element marked Swap marked element with element to the left. If matrix row marked Swap marked row with the row above.
[SHIFT][DOWN]	If matrix element marked Swap marked element with element to the right. If matrix row marked Swap marked row with the row below.



16.9 Lists

Key	Description
[{]	Insert list
[DIAMOND][[]]	Add subscription to marked expression.
[.]	Add element to marked list.
[BACKSPACE]	If placeholder marked Delete element
[SHIFT][UP]	Swap marked element with element to the left.
[SHIFT][DOWN]	Swap marked element with element to the right.

16.10 Strings

Key	Description
[“]	Convert marked expression to a string

16.11 Menus and windows

Key	Description
[MATH], [CATALOG], [CHAR], [MODE], [MEM], [VAR-LINK], [APPS], [DIAMOND][APPS]	System menus.
[UNIT]	System unit menu. Units are multiplied to marked expression.
[CUSTOM]	Toggle the custom menu on/off.
[DIAMOND][K]	System symbol table
[F1] (EQW primary menu)	Tools
[F2] (EQW primary menu)	Algebra
[F3] (EQW primary menu)	Calculus
[F4] (EQW primary menu)	Other
[F5] (EQW primary menu)	Custom. Executes “main\eqwuser()”.
[F7] (EQW primary menu)	Activates EQW secondary menu.
[F1] (EQW secondary menu)	Activates EQW primary menu.
[F2] (EQW secondary menu)	Number menu
[F3] (EQW secondary menu)	Angle menu
[F4] (EQW secondary menu)	List menu
[F5] (EQW secondary menu)	Vector menu
[F6] (EQW secondary menu)	Matrix menu

16.12 Keyboard programs

Key	Description
[DIAMOND][1] - [DIAMOND][9]	Executes keyboard programs “main\eqwprgm1” - “main\eqwprgm9”.



16.13 Switching application

Key	Description
[Y=]	Switch to the Y= editor.
[WINDOW]	Switch to the Window editor
[GRAPH]	Switch to the Home screen.
[TblSet]	Switch to the Table Set editor.
[TABLE]	Switch to the Table editor.
[2nd][APPS]	Switch to the previous active application.

16.14 Other keys

Key	Description
[RCL]	Recall content of a variable. The content of a variable replaces marked expression.
[QUIT]	Quit to Home.
[ESC]	Send the contents of the EQW to the entry line and quit to Home.
[F8]	Switches to the View Screen.
[DIAMOND][(-)]	Converts marked variable to a Local/Global variable.
[DIAMOND][)]	Decreases the text size of the displayed expression. Large text ⇒ Medium text Medium text ⇒ Small text Small text ⇒ Large text



17. API

The Application Programming Interface is a group of programs, which extends the calculators built-in TI-Basic language. One purpose of these programs is to make it possible for Basic programs to use the EQW for input/output. Another purpose is to make it possible to use Basic programs for customising and extending the EQW.

17.1 EQW.Current

17.1.1 EQW.Current()

Type: Program.
Argument: non.
Description: Displays the current content of the EQW.
Example: EQW.Current()

17.2 EQW.Get

17.2.1 EQW.Get()

Type: Function.
Argument: non.
Description: Returning the marked expression/sub-expression from the EQW.

Note: If the marked expression is not valid then an error will occur.

Example: If the content of the EQW look like this:

A calculator display showing a fraction $\frac{a+b}{c}$. A cursor is positioned over the numerator $a+b$.

EQW.Get() [ENTER] a+b

A calculator display showing a fraction $\frac{a+b}{c}$. A cursor is positioned over the denominator c .

EQW.Get() [ENTER] c

Program example: Following EQW keyboard program defines the key [DIAMOND][1]. The program adds the marked expression in the EQW to the start of the list main\eqwsave. The program must be placed in the MAIN folder.

```
:eqwprgm1()
:Prgm
: if GetType(main\eqwsave) ≠ "LIST"
:      { }→main\eqwsave
: augment({EQW.Get()},main\eqwsave) → main\eqwsave
:EndPrgm
```



17.3 EQW.Input

17.3.1 EQW.Input()

Type: Program.
 Argument: non.
 Description: Pauses the program, displays the current content of the EQW and lets you manipulate the content. The program continues when [ESC] or [QUIT] is pressed.

If the user press [ESC] then the system variable ok is set to 1 and, the EQW will return the resulting expression using one of two different methods depending on the set-up of the editor.

1. If a variable is open for editing in the EQW then the resulting expression will be stored in the variable. The variable may either have been opened using the EQW.Open method or by use of the “Open...” menu option and setting opening Mode to edit.
2. If no variable is open for editing in the EQW then the resulting expression will be stored in the clipboard and the key-press [PASTE] is stored in the keyboard buffer.

If the user press [QUIT] the variable ok is set to 0.

Note: It is possible to enter an expression in the EQW, which are not valid. The EQW will not check whether or not an expression is valid before it is stored in a variable. A variable containing an invalid expression will result in an error when evaluated.

Program example 1: Following program emulates the program version of the EQW. The program will accept different types of arguments (see Insert for a description of the argument types). The edited expression is returned via the clipboard to the entry-line.

```
:eqw(expr1)
:Prgm
:eqw.new() ; clear the editor and make sure
; that the result will be returned
; via the clipboard.
:eqw.Insert(expr1) ; insert the expression to edit in
; the EQW
:eqw.Input() ; let the user edit the expression
:EndPrgm
```

```
Example:
eqw(x^2+1) [ENTER] ; Starts the EQW displaying
; x^2+1. The argument is
; evaluated/simplified before it is
; displayed.
eqw("2+3+4+5") [ENTER] ; Starts the EQW displaying
; 2+3+4+5. The argument is
```



; not evaluated/simplified before
; it is displayed.

Program example 2: Following program allows the user to edit an expression, and stores the resulting expression in the variable 'result'.

```
:editexpr1()
:Prgm
:delvar result ; clear old result
:eqw.Open(result) ; Open the variable 'result' for
; editing.
:eqw.Insert(cSolve(ÿ*x^2+ÿ*x+ÿ=0,x)) ; insert the expression to edit in
; the EQW
:eqw.Input() ; let the user edit the expression
:If ok=1 Then ; if the user pressed [ESC] then
:.... ; do something with the result
:EndIf
:EndPrgm
```

17.3.2 EQW.Input(VAR)

Type: Program.
Argument: Path-name of variable to use for input. If the variable does not exist, it will be created.
Description: Pauses the program, closes eventually open variables in the EQW, displays the EQW with the content of the variable VAR and lets the user manipulate the content. The program continues when [ESC] or [QUIT] is pressed.
The content of the variable VAR can be of type expression, string, matrix or list. The expression may contain 'ÿ' (char(255)), which will be interpreted as placeholders.
If [ESC] is pressed the system variable ok is set to 1 and the content of the EQW will be stored in the variable VAR.
If [QUIT] is pressed the variable ok is set to 0 and the content of the variable VAR is undefined.

Note: It is possible to enter an expression in the EQW, which are not valid. The EQW will not check whether or not an expression is valid before it is stored in a variable. A variable containing an invalid expression will result in an error when evaluated.

Program example: Following program allows the user to enter an expression, and stores the resulting expression in the local variable 'result'.

```
: editexpr2()
:Prgm
```



```

:Local result
:eqw.Input(result)
:If ok=1 Then ; if the user pressed [ESC] then
:.... ; do something with the result
:EndIf
EndPrgm

```

17.4 EQW.Insert

17.4.1 EQW.Insert(expression)

Type: Program.
Argument: expression.
Description: Evaluates/simplifies expressions, displays the current contents of the EQW and replaces the selected text with a simplified expression.

There are two variable names, which are reserved by the EQW. These variables have a special meaning in expressions.

Variable name	Comments
'ÿ' = char(255)	'ÿ' is inserted in the EQW as a placeholder for a missing argument. Expressions containing 'ÿ' are not evaluated/simplified before they are inserted in EQW.
Eqw	Variables with the name "eqw" are replaced by the marked expression/sub-expression in the EQW before the expression is evaluated.

Note: Expressions cannot contain Basic commands.

Example: If the content of the EQW looks like this:

$$(\sin(t))^2 + (\cos(t))^2 = 0$$

Execute: EQW.Insert(tan(t)^2)

Result in the EQW:

$$(\sin(t))^2 + (\tan(t))^2 = 0$$

Program example: Following EQW keyboard program defines the key [DIAMOND][1]. The program solves the marked equation in EQW with respect to 'x' and inserts the result at the cursor position. The program must be placed in the MAIN folder.

```

:eqwprgm1()
:Prgm
:eqw.insert(solve(eqw,x))

```

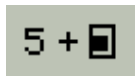


:EndPrgm

17.4.2 EQW.Insert(expression-string)

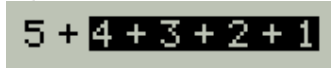
Type: Program.
 Argument: expression-string.
 Description: Converts the expression-string to an expression, displays the current content of the EQW and replaces the selected text with an expression. An expression-string can contain any valid expression including Basic commands.
 Note: Expression-strings are not simplified.

Example: If the content of the EQW looks as follows:



Execute: EQW.Insert("4+3+2+1")

Result in the EQW:

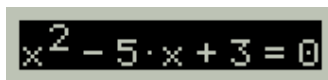


17.4.3 EQW.Insert(command-string)

Type: Program.
 Argument: command-string.
 Description: Displays the current content of the EQW and inserts the appropriate input-form for the command/function in the command-string with selected text as the first argument. A command-string is a string, which contains the exact name of a function or command. A command-string may contain the name of any build-in command, function or operator. A command-string may also contain the name of an undefined function; user defined functions/programs or Basic extensions.

Commands-strings are case sensitive and build-in commands and function must be written exactly as they appear in system menus (Catalog and Math menus). The safest way to create a command string is to pick the command/function from a system menu. Some examples of command-strings: "cSolve(", "&", "Graph ", "and", "OneVar ", "^-1", "=", "x(", "myfolder\.myprog(", "EQW.tToCos(".

Example: If the content of the EQW looks like this:



Execute: EQW.Insert("cSolve(")



Result in the EQW:

```
cSolve(x2 - 5·x + 3 = 0, □)
```

17.4.4 *EQW.Insert(key-string)*

Type: Program.
Argument: key-string.

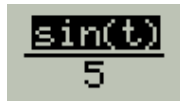
Description: Displays the current content of the EQW and sends the key-string as a series of key-presses to the application. A string is threaded as a key-string when the first character in the string has the numeric code 14. The identifier character is not sent. Characters with a numeric code between 14 and 31 are translated to other key-press before they are sent. You can find a list of all character codes in your calculator manual. You can get a character with a given numeric code from `char(#)`.



Key translation table

Character codes	Key press
14	[DIAMOND][ENTER]
15	[ENTER]
16	[ENTRY]
17	[2nd][LEFT]
18	[2nd][RIGHT]
19	[2nd][UP]
20	[2nd][DOWN]
21	[LEFT]
22	[RIGHT]
23	[UP]
24	[DOWN]
25	[BS]
26	[DIAMOND][CLEAR]
27	[CLEAR]
28	[DEL]
29	[SHIFT][LEFT]
30	[SHIFT][RIGHT]
31	Menu function: To Home

Example 1: If the content of the EQW is as follows:



Execute: `EQW.Insert(char(14)&char(27))` (this will invoke [CLEAR])

Result in the EQW:





Example 2: This macro will insert the definite integral from zero to infinity with the selected as argument and let you fill in the variable of integration.

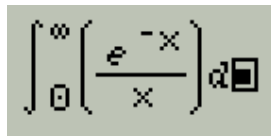
Char(14)&"[(" &char(24) &char(24) &char(23) &char(24) &char(22)&char(22)&"0" &char(22)&"∞" &char(23) &char(16)

If the content of the EQW look like this:



Execute: EQW.Insert(Char(14)&"[(" &char(24) &char(24) &char(23) &char(24) &char(22)&char(22)&"0" &char(22)&"∞" &char(23) &char(16))

Result in EQW:



17.5 EQW.New

17.5.1 EQW.New()

Type: Program.
 Argument: non.
 Description: Closes eventually open variables, clears the editor and displays the editor.

Example: EQW.New()

17.6 EQW.Open

17.6.1 EQW.Open(VAR)

Type: Program.
 Argument: Path-name of variable to open. If the variable does not exist, it will be created.

Description: Closes eventually open variables and opens the variable VAR for editing in the EQW. If no folder is given then current folder will be used.



Note:

If the variable-name is invalid or the variable-name is reserved an error will occur.

Example:

`EQW.Open(folder\myvar)`



18. Troubleshooting

18.1 The EQW will not mark the nominator/denominator of a fraction.

The EQW handles two types of numerical fractions.

18.1.1 Unevaluated numerical fractions

Numerical fractions entered in the EQW are threaded as common divisions where nominator and denominator are separate entities.

18.1.2 Evaluated fractions

Evaluated numerical fractions are fractions, which are result of an evaluation by the OS. The OS converts all numerical fractions to single entities. The EQW will not convert it back by it self. This type of fraction can be split into nominator and denominator by pressing [BACKSPACE][DOWN].

18.2 How to enter an empty string

An empty string is created in two steps:

1. Insert the string by pressing [“].
2. Delete the placeholder between the quotation marks using [BACKSPACE].

18.3 The cursor disappears when inside an empty string

The cursor does not disappear; it does just not enclose anything (the string is empty). The cursor is in this case displayed as a line between the quotation marks. The cursor indicates that you still can enter text into the string even without a placeholder.

18.4 The EQW will not delete a placeholder

The EQW will not allow deletion of a placeholder if the resulting expression becomes invalid.

18.5 How to change the direction of a limit

See the example with limit in the section: Entering functions/commands/programs 6.2.

18.6 Error when trying to use a variable created by the EQW

It is possible to enter an expression in the EQW, which is not valid and the EQW does not check whether or not an expression is valid before it is stored in a variable. A variable containing a non-valid expression will result in an error when evaluated. Opening the variable in the EQW and correcting the expression will solve the problem. Variables containing a non-valid expression can only be opened using the EQW.open method or the “Open...” menu option.

18.7 Unable to delete the EQW or applications installed before the EQW

The EQW or apps installed before the EQW can not be uninstalled as long as the EQW have keyboard shortcuts activated. To solve this problem turn EQW keyboard shortcuts off. Keyboard shortcuts are removed by selecting Tolls\Format... ([F1][C]) in the EQW and setting Keyboard Shortcut to OFF.