Preliminary Version (incomplete translation)

Quine Mc Cluskey

Manual for Version 3.34

http://www.iapetus.ch/software

Table of Contents

A. Warranty

This Software is supplied "as is" and we expressly disclaim all warranties, express or implied, including (but not limited to) warranties of merchantability, fitness for a particular purpose, and non-infringement. There is no warranty that the computed results are correct.

You must accept the entire risk of downloading or using this software. If you do not agree to these terms do not use this software.

Or short: Use at your own risk.

B. License

This program is freeware.

This means you may use this software free of charge. You may copy and redistribute this software as long as you do this free of charge. You must not modify the software. If you redistribute the software you must and you must distribute *all* files belonging to the program. In case you got an incomplete copy you might download the newest version from:

http://www.iapetus.ch/software

C. Acknowledgments

R. Zürcher
Gfeller Telecommunications, Bern, Switzerland
gave the initial idea for this program.

H. Pfahlbusch
Hochschule Mittweida, Mittweida / Sachsen / Germany
for the algorithm used for the alternative form and for detailed bug reports and analysis.

1. Introduction

1.1. What does this program do?

This program calculates starting from a table of values the konjunktive or disjunktive minimized form. The result us shown on screen or written to a file. In case of more than one solution, all of them are calculated.

An integrated editor is used to write the table of values down. A versatile syntax supports a simplified notation. Several control commands added to the table allow to adjust the way the solution is calculated.

The program runs using Windows 98+ . The software is freeware. See chapter B.

1.2. Sources

The algorithm was developed in 1951 by the philosoph and mathematician Willard Van Orman Quine in collaboration with software programmers.

Some publications:

Quine,W. V. O.: "The problem of simplifying truth functions"
1952 : ibid. 17: 156 (vermutlich eine Sammlung in der CSFR...)

Quine,W. V. O.: "On cores and prime implicants of truth functions"
Nov. 1959 in American Mathematical Monthly 66: No. 9, S.755-760
Lancaster PA., ISSN 0002-9890

McCluskey, E. J.: "Minimisation of boolean Functions"
1956 in Bell System technical Journal, S.1417-1444.

Karnaugh, M.: "The map method for synthesis of combinational logic circuit"
Nov 1953 in Trans. AIEE Communications and Electronics. 72,
S.593-599

Veitch, E. W.: "A chart method for simplifying truth functions"
Mai 1952 in Proc. Assoc. for Computing Machinery, Pittsburgh.

2. How to launch the program

2.1. Installation

The program is contained in the file **QMC3xx.ZIP**. To install simply unpack the archive to a temporary directory and run SETUP.EXE.

For the editors the two true type fonts **MS LineDraw** or **Terminal** must be installed to get a correct display. See QMC homepage at http://www.iapetus.ch/software for locations where to download this font.

2.2. Start of Program

The program is launched as follows:

```
qmc3 [qmc_source_filename]
```

The optional source file (*.QMC) is loaded automatically into the editor. If present, a corresponding result file (*.OUT) is loaded into the result window.

The program can also be used in batch mode by adding the parameter --autorun :

```
qmc3 --autorun qmc_source_filename
```

or:

```
start /w qmc3 --autorun qmc_source_filename
```

The calculation starts automatically and the program exits after completion of the calculation.

The user interface language is auto detected, but can be forced by adding the parameter --lang :

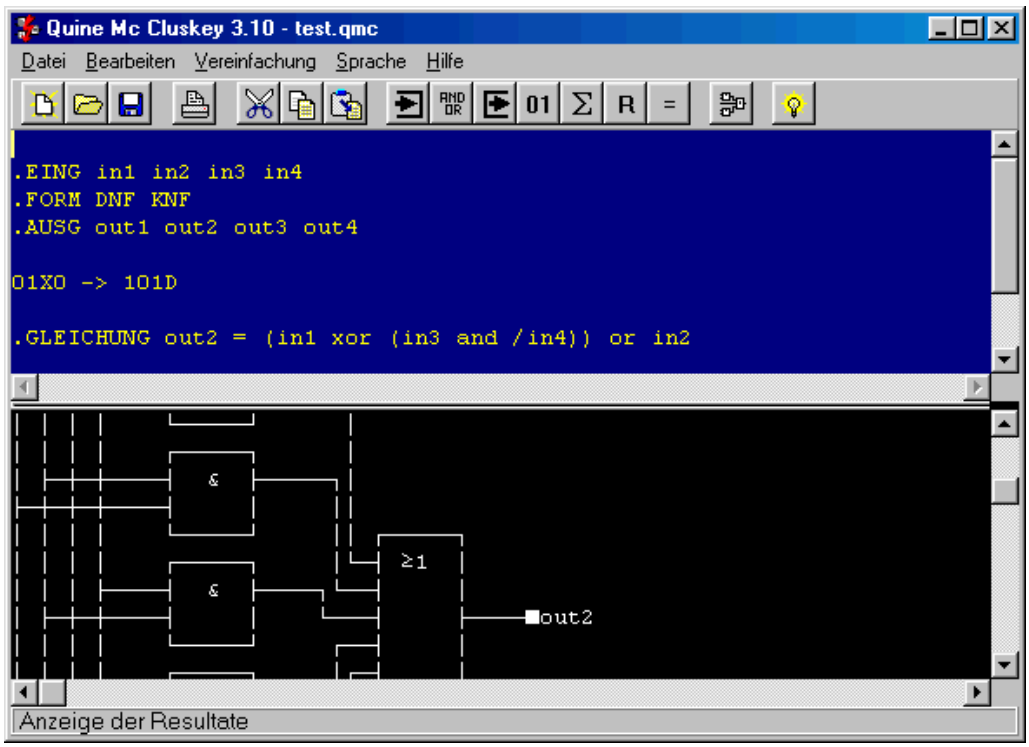
```
qmc3 --lang en [qmc_source_filename]
```

or:

```
qmc3 --lang de [qmc_source_filename]
```

2.3. Brief Overview of the User Interface

After lauch the program presents itself like this:



The interface consists of two parts: the upper blue source editor and the lower black result windows.

Using the menu “File” source code (*.QMC) is loaded into the source editor. If present, a corresponding result file (*.OUT) is loaded into the lower result window.

Text may be copied from the lower to the upper window using the clipboard.

The print command (menu “File” - “Print”) prints exclusively the content of the lower result window.

A toolbar allows fast access to often used functions.

The interface language is changed using LANG environment variable (overridable using the --lang command line switch). Currently supported languages are German and English.

2.4. Changelog

Neu in Version 3.34:

- Bug fix: Translations

Neu in Version 3.32:

- Bug fix: Printing on several pages
- Source ported to Lazarus
- Translations rewritten

New in Version 3.22:

- Bug fixed: Calculation using 16 inputs hanged
- New Feature: Input of truth table in hex-format.

New in Version 3.20:

- New feature: calculation forms NAND and NOR.
- English version of documentation

New in Version 3.18:

- Bug fixed: “Division by zero” - Error caused by the progressbar (!) when only one minterm could be found.

New in Version 3.16:

- Bug fixed: Wrong font in result window. Used font is now configurable.
- Error in output formatting fixed when using more than one calculation form simultaneously. .

Update:

- Chapter 5.4 Tips and Tricks expanded.

New in Version 3.14:

- Improved performance when defining more than 10 inputs
- Fixed error in manual (missing images) .

Update:

- Manual as PDF file.

New in Version 3.12:

- Bug fixed : Syntax error in source file was reported despite the syntax in the file was correct.

New in Version 3.10:

- Elimiation of the length limit of 255 chars in equations (.GLEICHUNG-command)
- Internationalization (de and en)
- --autorun parameter for batch mode.
- Bug fixed: The last line of a source file was ignored under certain conditions.

New in Version 3.02

- ANF calculation mode
- .PI-command

Version 2.05

- Last version for Windows 3.1

Version 1.0

- Original version for MSDOS

3. Source files

The to be simplified circuit is defined in the upper, blue edit window. There are only minimal requirements regarding the formatting of the text:

- One command only per source code line (exception: $-Command).
- Single words are separated by spaces.
- Everything is case insensitive.

See the following sections for an explanation of individual commands.

3.1. Structure

The order of the commands in the source file is arbitrary as long there is only one command per line. By placing a $-sign at the beginning of a line one command can also span over multiple lines.

Text blocks showing the syntax of the individual commands may be inserted in the upper edit window using the speed buttons (the icons) below the menu or using the menu itself.

3.2. Syntax

All commands are shown using the same syntax. Each command consists of up to four parts:

1. Mandatory parts:
   shown in **bold letters**. They must always be present.
2. Variant parts (A)
   Different possibilities separated by **|** . One of the possibilities must be chosen
3. Variant parts (B)
   A selection in **{ }**, separated by **|** . One or more of the possibilities must be chosen
4. Optional parts
   They may or may not be present. They are shown in parenthesis **[ ]**.
5. Parameter
   Strings consisting of numbers, formulas or names, indicated by a placeholder in **<>** - characters.

3.3. Comments

```
Syntax: ; [a comment]
```

Comments start at the semicolon and end at the line end.

Example:

```
; This is a comment
```

3.4. $-Sign

Syntax: **$** [continuation of last line]

The rule is one command per line. In case one line is too short, the next line might logically appended to the current line by starting the next line with a $ - char. The $-Char must be places at the very beginning of the line.

Example:

```
.EING first_input second_input third_input ; 3 inputs
$ fourth_input ; another input
```

this is equivalent with:

```
.EING first_input second_input third_input fourth_input
```

3.5. .FORM Command

Syntax: **.FORM** { DNF | KNF | ANF | NAND | NOR }

Symbol: 

Indicates witch normal form is to be used. This command is obligatory. More than one normal form may be specified.

DNF	Calculation using the disjunktive normal form
KNF	Calculation using the konjunktive normal form
ANF	Calculation using the alternative normal form
NAND	Disjunktive normal form with output using NAND-gates.
NOR	Konjunctive normal form with output using NOR-gates.

If the calculated result is too complicated, one might try another normal form. Often the complexity of the result and the computation time change dramatically when choosing another normal form.

Example

```
.FORM DNF ; disjunktive normal form
.FORM KNF ANF ; konjunktive and alternative normal form
```

3.6. .EING Command

Syntax: **.EING** [<EName1> [<EName2> ...]]]

Symbol: 

This command defines the names and the number of the input variables. This command is obligatory. Up to 16 input variables might be specified. the order of the names given is the same as the order of the inputs used in the table of values. To avoid ambguities with the operators used in expressions with the .GLEICHUNG-command the names of boolean operators (like and, or, xor, not, ..) should not be used as input variable names.

Examples

```
.EING A0 A1 WR ; we have 3 input variables: A0, A1, WR
```

3.7. .AUSG Command

Syntax: **.AUSG** [<AName1> [<AName2> ...]]]

Symbol: 

This command defines the names and the number of the output variables. This command is obligatory. Up to 16 output variables might be specified. the order of the names given is the same as the order of the inputs used in the table of values. To avoid ambiguties with the operators used in expressions with the .GLEICHUNG-command the names of boolean operators (like and, or, xor, not, ..) should not be used as output variable names.

Beispiele

```
.AUSG en Sel ; we have 2 output variables: en and Sel
```

3.8. .RESULTAT Command

yntax: **.RESULTAT** SCHEMA | GLEICHUNG | TABELLE

Symbol: 

Depicts the output format of the result. There are three possibilities:

GLEICHUNG	The result will be given as boolean equation (see below). This is the default value.
SCHEMA	The result will be given as digital circuit drawing, IEC symbols are used.
TABELLE	A complete truth table will be shown. All relations between .GLEICHUNG-commands, Don't care-output values and X-Symbols are resolved. This table may get very long!.

All output formats can be combined. At least one is needed.

When showing the result as equation the following notation is used:

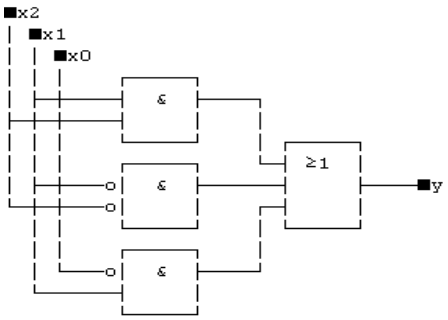
/A means: not A

A * B means: A and B

(A * /C) + (/A * B) means: (A and (not C)) or ((not A) and B)

(/A + B) * (C + /E) means: ((not A) or B) and (C or (not E))

Using the circuit drawing mode (SCHEMA) the result looks like this:



Special case: if an output turns out to be independent of all inputs (*log.1*) or (*log.0*) are assigned as results to represent logic 1 or logic 0 level. To avoid confusion the two strings shown in italics should not be used as names for input variables.

If the .RESULTAT-command is omitted the result is shown as equation.

Examplese:

```
.RESULTAT SCHEMA ; Resultat als circuit drawing
.RESULTAT SCHEMA GLEICHUNG TABELLE ; all possible types of output
```

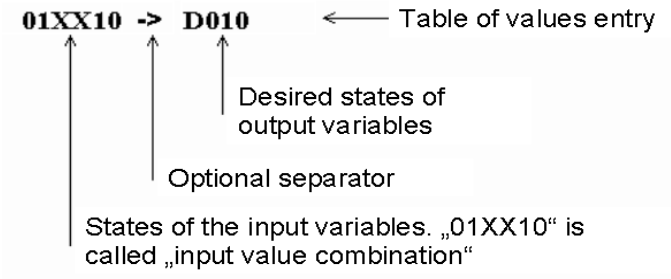
3.9. Entry in the Table of Values

Syntax: **0|1|X** [0|1|X [0|1|X [...]]] [->] **0|1|D** [0|1|D [...]]

or: ***Hexdigit*** [*Hexdigit* [...]]**h -> *Hexdigit*** [*Hexdigit* [...]]**h**
with Hexdigit = { 0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F }

Symbol: 

Explanation:



1 and **0** represent the logic level of the corresponding input variables. As many input variables as defined using the .EING command must be defined.

The character **X** represents both states, **1** and **0,** i.e., instead of defining an entry containing one **X**, two entries containing **1** and **0** at the position of the **X** can be defined. An entry with two **X** chars would be replaced by four entries with all possible combinations of **1** and *0* at the position of the two **X** (see examples).

The arrow sign **->** between input and output states is optional. It increases readability of the source and has no other purpose.

The letters **1**, **0** and **D** on the right side of the arrow define the desired output state for the input value combination. As many outputs as defined with the .AUSG command must be specified. The three possible letters have the following meaning:

1	The corresponding output is set to **1**.
0	The corresponding output is set to **0.**
D	The level of the corresponding output does not matter (**D**on't care). The solver will set the output to the logic level that results in a simpler solution.

The elements of an value table entry may be separated by space or tab characters to improve readability as long as one complete entry stays on one line.

Examples

```
01 -> 1 ; 1st output is set to 1 for 01 on the inputs

11xx1 -> d0 ; 1st output: Don't care,
            ;   the entry aplies to any state of the 3rd and 4th input
            ; 2nd output: log. 0
; this is the same as:
1100 1 -> d0
1101 1 -> d0
1110 1 -> d0
1111 1 -> d0 ; X characters expanded


xxxx -> 0111 ; any input combination results in
             ; log. 0 on th first output and log. 1 on outputs 2 to 4.
```

The last example is identical to:

```
.REST 0111 ; same effect as above if there is no table of values.
```

```
It is important to make sure that a certain input combination is not
assigned to 0 for an output in one entry and assigned to 1 for the same
output in an other entry. This happens easily when using the X-character. An
exception to this is that you may redefine an output set to Don't care.
```

Examples:

```
1111 -> 0
xxxx -> 1 ; not allowed, as 1111 was already set to log. 0. and is now
          ; redefined (hidden in xxxx) to log. 1.


But:

101010-> 1
xxxxxx -> d ; allowed, because log. 1 (or log. 0) has a higher priority
            ; than the don't care definition for 101010 hidden inside
            ; xxxxxx. The result of this two lines is a table filled with
            ; don't care entries except 101010 which is assigned to 1.
            ; This scheme allows a compact notation
.
```

Important:

A entry in the table of values has higher priority than a definition using the .GLEICHUNG command which itself has a higher priority than a definition using the .REST command.

Abbreviated writing form using hex digits::

Input value combinations and output values consisting of only the letters **1**, **0** may be entered as hex numbers. Hex numbers are recognized by the the suffix **h** or **H**:

```
01111 -> 0101
10000 -> 1100

ist identical to:

0Fh -> 5h
10h -> Ch
```

If *n* input variables and *m* output variables are used, the *n* least significant bits of the input variable combination and the *m* least significant bits of the output variable definition are considered. Missing leading bits are filled with 0.

Example:

For 5 input variables, the following lines are equivalent:

```
01111 -> 0101
Fh  -> 5h       ; 5th bit is set to 0
0Fh -> 5h       ; 3 leading bits are ignored
8Fh -> 5h       ; idem.
```

3.10. .REST Command

Syntax: **.REST 0|1|D** [0|1|D [...]]

or: **.REST *Hexdigit*** [*Hexdigit* [...]]**h**
with Hexdigit = { 0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F }

Symbol: 

All combinations of input values not listed in the table of values may be assigned to a specific output value using this command. The possible parameters are the same as those used in chapter 3.9 for the output part of the table of values entry. Instead of using this command one could also add all missing entries to the table of values and assign the desired output values to those table of value entries. The .REST command requires exactly the same number of parameters as outputs are defined with the .AUSG command.

Important:

This command does not make sense in combination with the .GLEICHUNG command. The . GLEICHUNG command has always higher precedence as it is always possible to evaluate an equation for all possible input combinations.

If neither .GLEICHUNG or .REST are used all undefined input combinations will have don't care (D) as desired output level assigned. e.g:

Examples:

```
.REST D0D ; all not listed input combinations of the first and third
          ; output are set to  don't care whereas the second output
          ; defaults to log. 0.
```

Similar to the truth table entries, states consisting only of **1**and **0** can be written in hex form. Hex numbers are recognized by the suffix **h** or **H**:

Example:

```
.REST 1111110

ist identical to:
.REST 7Eh
```

3.11. .GLEICHUNG Command

Syntax: **.GLEICHUNG <Output> =<Equation>**

This command is an extension of the .REST command. Instead of assigning a fixed value to not defined table of value entries an expression is evaluated and its result used instead. The .GLEICHUNG-command defines an expression for one single output variable. If an expression is to be used for more than one output more than one . GLEICHUNG commands must be used.

An expression consists of input variable names, constants and operators. Infix notation is used.

Two **constants** are defined that may be used at any place within an expression:

1	Constant for logic level 1.
0	Constant for logic level 0.

Valid **operators** are:

( )	Parentheses, may be used to get other evaluation order
/ , NOT	Inverts the following expression or variable.
* , AND	And operator
NAND	And operator followed by inversion
+ , OR	Or operator
NOR	Or operator followed by inversion
XOR, EXOR	Anticoincidence operator
XNOR, EXNOR	Coincidence operator

The priority of the operators is decreasing from top to bottom. Note that operator names may not be used as input or output variable names.

Examples:

```
.EING a0 a1 a2 a3 ; 4 inputs
.AUSG en ; 1 ouptut
.GLEICHUNG en = ( (not a0) and (not a2) ) or a3
   ; for every not in the table of values defined entry
   ; this expression will be evaluated.
   ; In the above example all parentheses
   ; could be omitted due to operator priority.

0000 -> 1 ; as exception to the above expression, the output should be
1111 -> 1 ; log. 1 if all inputs are identical
```

The table of values has always higher precedence than an expression. The expression is only evaluated when there is no entry in the table of values for a specific input value combination.

The following two .GLEICHUNG commands substitute the .REST command mentioned below:

```
; .GELICHUNG version:
.GLEICHUNG Ausgang1 = 1 ; 1st output always = 1
   ; No expression for 2nd output. It will thus default to don't care.
.GLEICHUNG Ausgang3 = 0 ; 3rd output is set to 0
```

This is the same as above:

```
; .REST version
.REST 1D0 ; Same effect as above
```

Important:

The .GLEICHUNG command has higher precedence than the .REST command. If there is neither a .REST command nor a .GLEICHUNG command for an output, input value combinations not defined in the table of values will have don't care assigned.

3.12. .PI Command

Syntax: **.PI EXAKT|SCHNELL|HEURISTISCH**

This command controls how prime implicants are selected to create the minimized final solution. This command is optional. If not specified the selection method defaults to SCHNELL (fast mode). The possible parameters of the command are listed below:

EXAKT	A complete calculation by multiplying all bit-fields of the prime implicants is done. This may take a very long time but in return all possible solutions are found.
SCHNELL, HEURISTISCH (Default)	The selection is done using a heuristic algorithm. This algorithm is fast but has some drawbacks: It is not sure that the best minimized solution is found. The algorithm may output several solutions that are not absolutely minimal (i.e. solutions that contain more terms than necessary). Under certain circumstances the search for solutions produces large amounts of redundant (identical) solutions due to the way the algorithm traverses the solution space. Also, the complete traversal of the solution space may take much longer than with the EXAKT but solutions are already available prior the end of the traversal. Because of that the search for solutions may be interrupted in this mode.

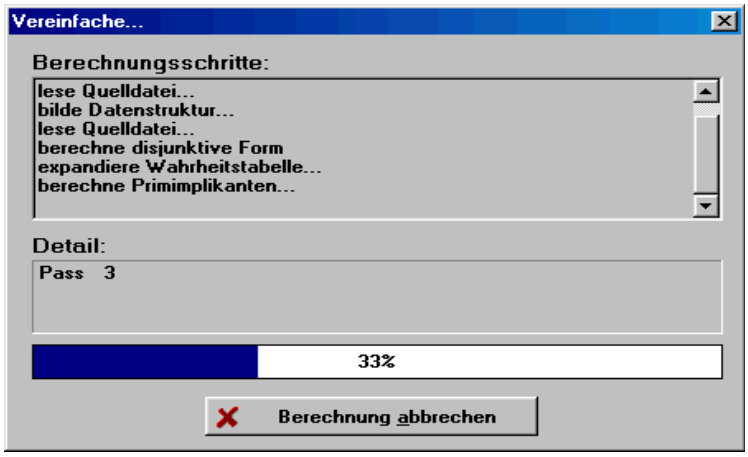
4. Results

The result of the calculation is shown in the result window. If the source code has already been saved to a file, the result of the calculation is written to a file with the same name but the extension '.OUT'.

4.1. Calculation of Results

Symbol: 

The calculation is started using the menu item "Calculation". The progress of the calculation is shown in the calculation window:



The calculation may be aborted any time by pressing "abort calculation'. Not however that in this case no results may have been written to the result file yet.

4.2. Calculation Steps

A typical case is shown below:

initializing memory...

reading source file...

This builds the internal tables needed for the calculation:

building data structure...

reading source file...

calculating in konjunktive mode

in the following step all value table entries containing X are expanded:

expanding truth table...

The calculation of the prime implicants follows. The percentage shows how much of the current pass is already done. The number of passes varies depending on the structure of the table of values. The prime implicant calculation may take a long time!

calculating prime implicants...

pass 1 100%
pass 2 100%
pass 3 100%
...

6 minterms and 5 prime implicants found

Some prime implicants may be redundant. These are eliminated in the next step. Depending on algorithm selected some or all solutions are found. Sometimes the same solution is found several times, in this case the redundant solution is deleted by a post processing filter. The filter also deletes solutions that are not minimal.

searching for redundant prime implicants...

(fast heuristic mode)

Finally writing the results out to a file:

writing solution to result file...

No other solutions for output number 1.

This sequence is repeated for each output. The collected solutions are then loaded into the result window.

4.3. Limitations

The main limitations are the maximum number of input variables and the size of the internal tables. Version 3.x allows for 16 input variables. The prime implicant tables allow to store about 65000 prime implicants. To make use of this table size at least 256MB RAM are needed (otherwise the program gets very slow due to swapping). The minimum memory needed to run the program is about 32MB.

Certain input configurations result in an excessive long calculation time. In this case it is favorable to change the calculation mode (.FORM command) to something else.

5. Problems

5.1. Error messages

Wärend der Berechnung auftretende Fehler werden in das Resultatfenster geschrieben. Fehlerhafte Quelltextzeilen werden mit einem ^ auf der Folgezeile markiert.

In den folgenden Listen sind alle Fehlermeldungen (mit entsprechenden Erklärungen) in alphabetischer Reihenfolge enthalten:

5.2. Errors in the source file

Diese Meldungen erscheinen beim Einlesen der Quelldatei. Wenn keine Resultatdatei angegeben wurde, so erscheint die fehlerhafte Zeile auf dem Bildschirm. Wurde eine Resultatdatei angegeben, so weist das Caret-Symbol (**^**) in der Resultatdatei auf die fehlerhafte Quelltextzeile.

Anzahl Eingänge ist undefiniert

Grund : Das Programm hat einen Wertetabelleneintrag gefunden, bevor mit dem .EING Befehl die Anzahl der verwendeten Eingänge definiert wurde. Eventuell fehlt der .EING Befehl sogar.

Abhilfe : Der .EING Befehl muss vor der Wertetabelle in der Quelldatei stehen.

Berechnungsform ist undefiniert

Grund : Das Programm hat einen Wertetabelleneintrag gefunden, bevor die mit dem .FORM Befehl die Berechnungsform definiert wurde. Diese Information ist aber zur korrekten Interpretation der Tabelle unerlässlich. Möglicherweise wurde der .FORM Befehl vergessen.

Abhilfe : Der .FORM Befehl muss vor der Wertetabelle in der Quelldatei stehen.

Die Quelldatei ist fehlerhaft.

Grund : Die Quelldatei konnte nicht vollständig interpretiert werden.

Abhilfe : In der Resultatdatei ist der Fehler näher erläutert. Existiert eine solche nicht, so steht die Erläuterung weiter oben auf dem Bildschirm.

Eintrag enthält eine ungültige Anzahl Ausgangsvariablen

Grund : Ein Wertetabelleneintrag enthält nicht die mit dem .EING Befehl definierte Anzahl Eingangsvariablen.

Abhilfe : Tabelleneintrag vervollständigen bzw. korrigieren.

Eintrag enthält eine ungültige Anzahl Eingangsvariablen

Grund : Ein Wertetabelleneintrag enthält nicht die mit dem .EING Befehl definierte Anzahl Eingangsvariablen.

Abhilfe : Tabelleneintrag vervollständigen bzw. korrigieren.

Nicht definierter Eintrag in der Wertetabelle

Grund : Dem Eintrag konnte keinen Ausgangszustand zugeordnet werden. Es wurde eine Ausgangsvariable vergessen.

Abhilfe : Eintrag durch den Suffix (die Suffixe) E,N oder D ergänzen.

unbekannter Befehl

Grund : Die betreffende Zeile enthält einen unbekannten Befehl. Vermutlich ist ein Befehl falsch eingetippt worden

Abhilfe : Befehl richtig eingeben!

unbekanntes Zeichen in der Wertetabelle

Grund : Die betreffende Zeile enthält ein ungültiges Zeichen. Vermutlich ist die Zeile falsch eingetippt worden

Abhilfe : Zeile richtig eingeben!

ungültige Anzahl Ausgänge

Grund : Es wurde keine gültige Anzahl Ausgänge angegeben.

Abhilfe : gültige Zahl angeben.

ungültige Anzahl Eingänge

Grund : Es wurde keine gültige Anzahl Eingänge angegeben.

Abhilfe : gültige Zahl angeben.

ungültige Berechnungsform (DNF, KNF, ANF, NAND oder NOR ist erlaubt)

Grund : Beim .FORM Befehl wurde eine ungültige Option angegeben.

Abhilfe : DNF, KNF, NAND, NOR oder ANF als Option angeben.

ungültige Option (D, E oder N ist erlaubt)

Grund : Beim .REST Befehl wurde eine ungültige Option angegeben.

Abhilfe : D, E oder N als Option verwenden.

ungültige Suchmodus (SCHNELL, HEURISTISCH, EXAKT sind erlaubt)

Grund : Beim .PI Befehl wurde eine ungültige Option angegeben.

Abhilfe : SCHNELL, HEURISTISCH, EXAKT als Option verwenden.

ungültiger Ausgabemodus (GLEICHUNG und/oder SCHEMA sind erlaubt)

Grund : Beim .RESULTAT Befehl wurde eine ungültige Option angegeben.

Abhilfe : SCHEMA und/oder GLEICHUNG als Option verwenden.

ungültiges Zeichen in der Wertetabelle

Grund : Die Wertetabelle enthält eine ungültige Zeile

Abhilfe : Die Zeile enthält ungültige Zeichen, z.B. irgendwelche Buchstaben. Sie muss korrigiert werden.

zuviele Eingänge angegeben (max *nn*)

Grund : Es wurden mehr als *nn* Eingänge angegeben.

Abhilfe : Es dürfen maximal *nn* Eingänge angegeben werden, sonst passen die Umsetztabellen nicht in den Speicher.

zuviele Ausgänge angegeben (max *nn*)

Grund : Es wurden mehr als *nn* Ausgänge angegeben.

Abhilfe : Es dürfen maximal *nn* Ausgänge angegeben werden. Sollten mehr als 16 Ausgänge nötig sein, so kann man die Quelldatei aufteilen.

zuwenig Ausgänge angegeben (min 1)

Grund : Es wurde weniger als 1 Ausgang angegeben.

Abhilfe : Es muss mindestens 1 Ausgang angegeben werden.

zuwenig Eingänge angegeben (min 1)

Grund : Es wurde weniger als 1 Eingang angegeben.

Abhilfe : Es muss mindestens 1 Eingang angegeben werden.

5.3. Error messages during calculation

Diese Fehler treten während der Berechnungsphase auf. Die Meldungen erscheinen in der Resultatdatei um schwarzen Editorfenster.

Berechnung durch Benutzer abgebrochen

Grund : Die Berechnung wurde durch den Benutzer abgebrochen.

Abhilfe : -

Fehler in Quelltext: Kombination doppelt!

Grund : Es wurde die selbe Zeile zweimal in der Wertetabelle angegeben.

Abhilfe : Eine Zeile löschen.

Quelldatei <Name> nicht gefunden

Grund : Das Programm konnte die Quelldatei nicht finden.

Abhilfe : richtigen Dateinamen angeben.

relevante Eingangskombination doppelt angegeben : XXXXX

Grund : Eine nicht - Don't care Eingangskombination wurde gleichzeitig als log.1 und als log.0 am Ausgang definiert.

Abhilfe : Dies kann leicht bei umfangreichen Wertetabellen und bei Verwendung des X Zeichens vorkommen.
z.B.. 1XXXX -> 0
XXXX1 -> 1
Nach Expandieren der beiden Zeilen treten Kombinationen auf, die in beiden Zeilen enthalten sind und somit nicht eindeutig zugeordnet werden können. Man beachte, dass wenn z.B. die *untere* Zeile eine Don't care Kombination ist, so wird kein Fehler ausgegeben. Die doppelt vergebene Kombination würde automatisch als der *oberen* Zeile zugehörig betrachtet. Dies vereinfacht oft die Wertetabelle. Die doppelte Kombination ist zu entfernen.

Resourcenmangel

Grund : Das Programm konnte nicht genügend Ressourcen wie Speicher, Dateien etc. finden. Diese Meldung erscheint in Begleitung mit einer anderen Fehlermeldung (s. dort).

Abhilfe : -

Stack.Newstack: too many stacks defined

Grund : interner Fehler

Abhilfe : Kontakt mit Programmentwickler aufnehmen

Stack.Dispstack: stack already disposed

Grund : interner Fehler

Abhilfe : Kontakt mit Programmentwickler aufnehmen

Stack.Pop: stack empty

Grund : interner Fehler

Abhilfe : Kontakt mit Programmentwickler aufnehmen

Stack.Pop/Push/Pick/StackSize: stack undefined

Grund : interner Fehler

Abhilfe : Kontakt mit Programmentwickler aufnehmen

Stack.Push: stack overflow

Grund : interner Fehler

Abhilfe : Kontakt mit Programmentwickler aufnehmen

Stack.Pick: too few elements on stack

Grund : interner Fehler

Abhilfe : Kontakt mit Programmentwickler aufnehmen

Überlauf der internen Tabelle

Grund : Die interne Umsetztabelle ist zu klein (zu viele Eingangsvariablen und eine zu komplizierte Wertetabelle).

Abhilfe : Weniger Eingangsvariablen verwenden, andere Berechnungsform (DNF oder KNF) ausprobieren. Wertetabelle vereinfachen.

Überlauf der Primimplikantentabelle

Grund : Die Primimplikantentabelle ist zu klein (es entstanden mehr als xxxxx Primimplikanten).

Abhilfe : Weniger Eingangsvariablen verwenden, andere Berechnungsform (DNF oder KNF) ausprobieren. Wertetabelle vereinfachen.

unable to allocate XXXX bytes

Grund : Das Programm konnte XXXX Bytes Speicher nicht anfordern.

Abhilfe : Auf diese Meldung folgt die Meldung zuwenig Speicher. (s. dort).

zuwenig Speicher

Grund : Der verfügbare Speicher reicht nicht aus.

Abhilfe : Speicher freigeben oder einen neuen PC kaufen :).

5.4. Tips and Tricks

Falscher Font in der Resultstanzeige

Grund : MS Linedraw konnte vom Resultatfenster nicht geladen werden.

Abhilfe : Für die beiden Editoren MUSS die True-Type Schriftart **MS LineDraw** oder **Terminal** installiert sein, sonst werden die Resultate falsch angezeigt. Wenn diese Schriftart unter einem leicht anderen Namen bereits auf dem System vorhanden ist, so kann der entsprechend Font im Menü Optionen->Schriftarten ausgewählt werden.
MS LineDraw ist auf diversen FTP Servern unter dem Namen gc0651.exe verfügbar (siehe z.B. http://www.google.com mit **gc0651.exe** als Suchbegriff).

Exception beim Start des Programmes

Möglicher Grund : Kein Standarddrucker installiert.

Abhilfe : Standarddrucker installieren. Falls kein Drucker vorhanden ist, einen Dummydrucker definieren.

Die Berechnung dauert sehr lange

Grund : Es wurde eine komplexe Wahrheitstabelle zum Vereinfachen definiert.

Abhilfe : Berechnungsform ändern, z.B. DNF statt KNF. Das so erhaltene Resultat kann eventuell von Hand in die gewünschte Form umgewandelt werden.

6. Applications

6.1. Example 1

Es soll eine Parity-Check Logik gebildet werden. Dazu sei folgende Wertetabelle gegeben :

Eingänge D0 D1 D2 D3 D4	Ausgang Parität
0 0 0 0 0	0
0 0 0 0 1	1
0 0 0 1 0	1
0 0 0 1 1	0
0 0 1 0 0	1
0 0 1 0 1	0
0 0 1 1 0	0
0 0 1 1 1	1
0 1 0 0 0	1
0 1 0 0 1	0
0 1 0 1 0	0
0 1 0 1 1	1
0 1 1 0 0	0
0 1 1 0 1	1
0 1 1 1 0	1
0 1 1 1 1	0
1 0 0 0 0	1
1 0 0 0 1	0
1 0 0 1 0	0
1 0 0 1 1	1
1 0 1 0 0	0
1 0 1 0 1	1
1 0 1 1 0	1
1 0 1 1 1	0
1 1 0 0 0	0
1 1 0 0 1	1
1 1 0 1 0	1
1 1 0 1 1	0
1 1 1 0 0	1
1 1 1 0 1	0
1 1 1 1 0	0
1 1 1 1 1	1

Setzt man diese Tabelle in eine Quelldatei um, so erhält man die Demonstrationsdatei DEMO1.QMC :

Zuerst Titel und Kommentare:

```
; Demonstrationsdatei 1
;Parity - Check Logik
;MWI
```

Es folgen die Steueranweisungen. Es muss im Minimum die Anzahl Eingänge (1) und die Berechnungsform (2) definiert werden. Die Variablennamen sind optional, sonst werden einfach die ersten Buchstaben des Alphabets verwendet. Das Resultat wird als Gleichung ausgegeben, da nichts anderes definiert wurde.

```
.eing D0 D1 D2 D3 D4 (1) ; 5 Eingänge
.form DNF (2) ; disjunktive Normalform
.ausg Resultat ; 5 Datenleitungen
.rest 0 ; nicht angegebene Kombinationen -> 0
```

Nun folgt die weiter oben gezeigte Wertetabelle:

```
; Wertetabelle :
0 0 0 0 1 -> 1
0 0 0 1 0 -> 1
0 0 1 0 0 -> 1
0 0 1 1 1 -> 1
0 1 0 0 0 -> 1
0 1 0 1 1 -> 1
0 1 1 0 1 -> 1
0 1 1 1 0 -> 1
1 0 0 0 0 -> 1
1 0 0 1 1 -> 1
1 0 1 0 1 -> 1
1 0 1 1 0 -> 1
1 1 0 0 1 -> 1
1 1 0 1 0 -> 1
1 1 1 0 0 -> 1
1 1 1 1 1 -> 1
; 1 am Schluss angegeben -> Eintrag wird am
; Ausgang log. 1 ergeben Die Zustände mit 0 müssen
; nicht spezifiziert werden (s. .REST-Befehl)
```

Und nun die vom Programm erzeugte Resultatdatei: Sie enthält als erstes eine Kopie der Quelldatei. Dann folgen die errechneten Resultate (es können durchaus mehrere minimale Lösungen existieren).

```
Quine Mc Clusky V 2.0 Nr. #20100690-001
Copyright (c) 1989,1996 by M.Wieser, http://www.iapetus.ch
Berechnung für 1. Ausgang
+++ Quelldatei DEMO1.QMC :
------------------------------------------------------------------
; Demonstrationsdatei 1
;Parity - Check Logik
;MWI
.eing 5 ; 5 Eingänge
.form DNF ; disjunktive Normalform
.namen D0 D1 D2 D3 D4 Resultat ; 5 Datenleitungen
.rest 0 ; nicht angegebene Kombinationen -> 0
; Wertetabelle :
0 0 0 0 1 -> 1
0 0 0 1 0 -> 1
0 0 1 0 0 -> 1
0 0 1 1 1 -> 1
0 1 0 0 0 -> 1
0 1 0 1 1 -> 1
0 1 1 0 1 -> 1
0 1 1 1 0 -> 1
1 0 0 0 0 -> 1
1 0 0 1 1 -> 1
1 0 1 0 1 -> 1
1 0 1 1 0 -> 1
1 1 0 0 1 -> 1
```

```
1 1 0 1 0 -> 1
1 1 1 0 0 -> 1
1 1 1 1 1 -> 1
; 1 am Schluss angegeben -> Eintrag wird am
; Ausgang log. 1 ergeben Die Zustände mit 0 müssen
; nicht spezifiziert werden (s. .REST-Befehl)
+++ Ende der Quelldatei DEMO1.QMC
------------------------------------------------------------------
+++ errechnete minimalisierte Normalform(en) :
------------------------------------------------------------------
Resultat = (/D0 * /D1 * /D2 * /D3 * D4) +
           (/D0 * /D1 * /D2 * D3 * /D4) +
           (/D0 * /D1 * D2 * /D3 * /D4) +
           (/D0 * /D1 * D2 * D3 * D4) +
           (/D0 * D1 * /D2 * /D3 * /D4) +
           (/D0 * D1 * /D2 * D3 * D4) +
           (/D0 * D1 * D2 * /D3 * D4) +
           (/D0 * D1 * D2 * D3 * /D4) +
           ( D0 * /D1 * /D2 * /D3 * /D4) +
           ( D0 * /D1 * /D2 * D3 * D4) +
           ( D0 * /D1 * D2 * /D3 * D4) +
           ( D0 * /D1 * D2 * D3 * /D4) +
           ( D0 * D1 * /D2 * /D3 * D4) +
           ( D0 * D1 * /D2 * D3 * /D4) +
           ( D0 * D1 * D2 * /D3 * /D4) +
           ( D0 * D1 * D2 * D3 * D4)
------------------------------------------------------------------
Keine weiteren Lösungen für 1. Ausgang
```

Wie ersichtlich ist, konnte das Programm diese Wertetabelle nicht in eine einfachere Logikschaltung überführen. Dies liegt daran, dass für diese Wertetabelle keine minimalere disjunktive Form als die voll ausgeschriebene Form existiert. Das heisst aber nicht, dass es nicht möglich ist, hier etwas zu vereinfachen. Die Berechnung mit der alternativen Form bringt folgendes Resultat:

```
Resultat = D0 xor D1 xor D2 xor D3 xor D4
```

6.2. Example 2

In a microprocessor system an enable signal must be generated for an external unit according to the following table:

Inputs notReset En WR A0	Output Enable
0 0 0 0	0
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	0
0 1 0 1	1
0 1 1 0	0
0 1 1 1	Don't care
1 0 0 0	1
1 0 0 1	Don't care
1 0 1 0	1
1 0 1 1	Don't care
1 1 0 0	Don't care
1 1 0 1	Don't care
1 1 1 0	Don't care
1 1 1 1	0

The table is transformed into a source code file. The source is found in the file DEMO2.QMC. Note the use of the X-character in the file to get a more compact notation of the table (not that the X character is really useful for this example, the example is a bit too simple to show the power of the X-character) .