Quine Mc Cluskey

Handbuch
für Version 3.24

http://www.iapetus.ch/software

Inhaltsverzeichnis

A. Gewährleistung

Wir schliessen ausdrücklich jede Gewährleistung für dieses Programm ("Software") aus. Die Software wird ihnen "so wie sie ist" zur Verfügung gestellt, ohne jede Gewährleistung irgendeiner Art, weder ausdrücklich noch konkludent, einschliesslich aber nicht eingeschränkt auf konkludente Gewährleistungen der Tauglichkeit, der Eignung für einen bestimmten Zweck oder des Nichtbestehens einer Rechtsverletzung. Das gesamte Risiko, das sich aus dem Verwenden oder der Leistung der Software entsteht, verbleibt bei Ihnen.

Oder kurz: Sie benutzen die Software auf eigenes Risiko!

B. Lizenzvereinbarungen

QMC ist Freeware.

Dies bedeutet, dass Sie die Software kostenlos benützen dürfen. Sie dürfen Sie Software auch kopieren und weitergeben, solange Sie dies auch kostenlos tun. Sie dürfen keine Änderungen an der Software vornehmen und müssen jeweils alle zum Programm gehörenden Dateien weitergeben. Falls Sie eine unvollständige Kopie erhalten haben, finden Sie die neueste Version unter:
http://www.iapetus.ch/software

C. Dank

Folgende Personen haben bei der Erstellung und Weiterentwicklung dieses Programmes wertvolle Unterstützung und Hinweise geliefert:

R. Zürcher
Gfeller Telecommunications, Bern CH
für den Anstoss zu diesem Programm.

H. Pfahlbusch
Hochschule Mittweida, Mittweida / Sachsen D
für den Algorithmus der "Alternativen Normalform" und für detaillierte Fehleranalysen.

1. Einführung

1.1. Was leistet das Programm?

Das Programm berechnet zu einer in einer Datei vorgegebenen Wertetabelle die konjunktive oder disjunktive minimale Normalform. Das Resultat dieser Rechnung wird auf den Bildschirm oder in eine Datei ausgegeben. Sollten mehrere gleichwertige minimale Normalformen existieren, so werden alle berechnet.

Die Wertetabelle ist einfach zu erstellen, da ein leistungsfähiger Befehlssatz eine vereinfachte Schreibweise unterstützt. Diverse Steuerbefehle erlauben eine weitgehende Kontrolle der Berechnung.

Das Programm ist unter Windows 98+ lauffähig. Das Programm ist **Freeware**. Siehe Kapitel B.

1.2. Quellen

Entwickelt wurde dieses Verfahren erstmals 1951 vom Philosophen und Logiker Willard Van Orman Quine in Zusammenarbeit mit einigen Programmierern.

Publikationen, die sich mit dem Vereinfachen von Logikgleichungen befassen:

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. Programmhandhabung

2.1. Installation

Das vollständige Programm ist in der Datei **QMC3xx.ZIP** gepackt. Zur Installation einfach das ZIP-Archiv in ein Verzeichnis entpacken und SETUP.EXE ausführen.

Für die beiden Editoren MUSS die True-Type Schriftart **MS LineDraw** oder **Terminal** installiert sein, sonst werden die Resultate falsch angezeigt. Siehe QMC Homepage unter http://www.iapetus.ch/software für Downloadmöglichkeiten.

2.2. Programmstart

Der Syntax zum Start des Programmes ist folgendermassen:

```
qmc3 [Quelldatei]
```

oder:

```
qmc3 --autorun Quelldatei
```

bzw:

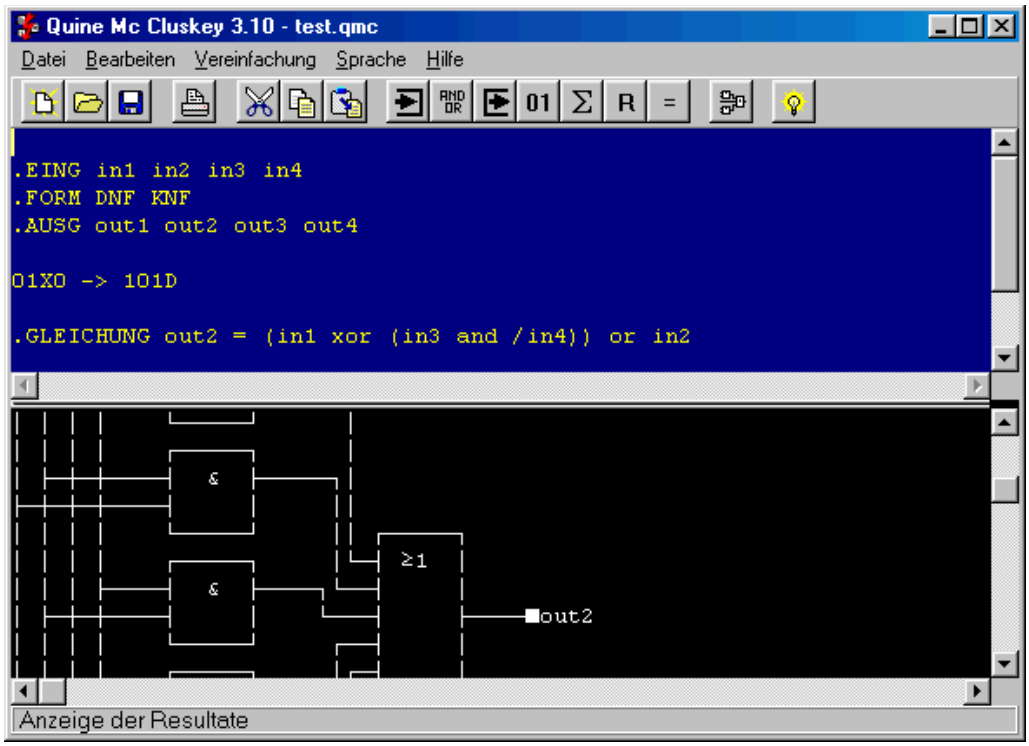
```
start /w qmc3 --autorun Quelldatei
```

Eine optional angegebene Quelldatei (*.QMC) wird beim Start automatisch in den Quelleditor geladen. Falls eine Resultatdatei (*.OUT) existiert, wird diese im Resultatfenster angezeigt.

Wird der Parameter --autorun zusammen mit einer Quelldatei, so wird wird die Quelldatei geladen und die Berechnung gestartet. Wenn die Berechnung fertig ist, wird das Programm beendet. Dies kann für den Betrieb im Batchmode verwendet werden.

2.3. Kurzbeschrieb der Oberfläche

Die Oberfläche bietet nach dem Start des Programmes folgendes Erscheinungsbild:



Die Oberfläche besteht aus zwei Bereichen, dem oberen, blauen Quelleditor und dem unteren, schwarzen Resultatfenster. Die Aufteilung zwischen den beiden Bereichen lässt sich durch Verschieben des Trennstriches verändern.

Über das Menü "Datei" lassen sich Quelltexte (*.QMC) in den Quelleditor laden. Wenn bereits eine entsprechende Resultatdatei (*.OUT) existiert, so wird diese ebenfalls geladen.

Die Zwischenablage-Funktionen können in beiden Editoren benutzt werden; im Resultateditor ist allerdings nur "Ausschneiden" möglich.

Der Druckbefehl druckt ausschliesslich den Inhalt des Resultateditors aus.

Eine Symbolleiste ermöglicht den schnellen Zugriff auf die Programmfunktionen.

Das Menü "Sprache" erlaubt die Einstellung der bevorzugten Interfacesprache. z.Z. wird Deutsch und Englisch unterstützt.

2.4. Neuerungen in dieser Version

New in Version 3.24:

- Bug behoben in der Schemaausgabe für die Berechnugnsformen NAND und NOR: Der Bug resultierte in inkorrekten Schemazeichungen. Die nun erzeugte Ausgabe ist aequivalent mit der Formelausgabe. Allerdings wird nun für sehr einfache Resultate, die nur aus Invertern und Nicht-Invertern (und keinen anderen Gattern) bestehen, in bestimmten Fällen ein nichtminimiertes Resultat ausgegeben; z.B. x=a wird als x=/(/a) angezeigt.

Neu in Version 3.22:

- Bug behoben: Berechnung mit 16 Eingangsvariablen blieb hängen
- Neues Feature: Eingabe von Wahrheitstabellen im Hex-Format.

Neu in Version 3.20:

- Neues Feature: Berechnungsformen NAND und NOR.

Neu in Version 3.18:

- Bug behoben: “Division durch null” - Fehler verursacht durch die Progressbar (!) wenn nur ein Minterm gefunden wird.

Neu in Version 3.16:

- Fehlerhafter Zeichensatz im Resultatfenster korrigiert. Die Anzeige sollte nun in weniger Fällen eine fehlerhafte Anzeige der Schemaausgabe liefern (MS Linedraw Font).
- Ausgabefehler bei gleichzeitiger Berechnung mit mehreren Normalformen korrigiert.

Update:

- Kapitel 5.4 Tipps und Tricks erweitert.

Neu in Version 3.14:

- Wesentlich verbesserte Performance bei mehr als 10 Eingangsvariablen.
- Fehler im Handbuch korrigiert (fehlende Grafiken).

Update:

- Handbuch als PDF-Datei.

Neu in Version 3.12:

- Bugfix : Bestimmte Quelldaten erzeugten einen Syntax-Fehler, obwohl die Datei korrekt war.

Neu in Version 3.10:

- Aufhebung der Längenbegrenzung vom 255 Zeichen in Formeln (.GLEICHUNG-Befehl)
- Mehrsprachiges Interface (z.Z. Deutsch und Englisch)
- --autorun Parameter für den Batchbetrieb.
- Bugfix: Die letzte Zeile einer Quelldatei wurde unter bestimmten Bedingungen nicht ausgewertet.

Neu in Version 3.02

- ANF Berechnungsmodus
- .PI-Befehl

Version 2.05

- Letzte Version für Windows 3.1

Version 1.0

- Urversion für MSDOS

3. Quellcode

Die zu vereinfachende Schaltung wird im oberen, blauen Textfenster definiert. An die Formatierung des Textes werden nur minimale Anforderungen gestellt:

- Jeder Befehl befindet sich auf einer eigenen Zeile (Ausnahme s. $-Zeichen).
- Die einzelnen Wörter werden gewöhnlich durch Leerzeichen getrennt.
- Gross und Kleinschreibung spielen keine Rolle.

Die erlaubten Befehle werden im folgenden erläutert.

3.1. Gliederung

Die Befehle können in beliebiger Reihenfolge in der Quelldatei stehen. Es gilt generell: pro Befehl eine Zeile. Mit dem $-Befehl kann diese Regel durchbrochen werden.

Die Quelldateien können im eingebauten Editor erstellt werden. Für die verschiedenen Befehle sind im Menu Textblöcke abrufbar. Textblöcke können ebenfalls über die Symbolleiste eingefügt werden.

3.2. Darstellung der Befehle

Der Syntax der Befehle wird in einem einheitlichen Schema dargestellt. Ein Befehl besteht aus bis zu vier verschiedenen Teilen.

1. zwingende Teile eines Befehls
   Sie werden durch **Fettschrift** hervorgehoben. Sie müssen immer angegeben werden.
2. variante Teile (A)
   Sie werden durch das Zeichen **|** getrennt. Es muss eine der definierten Varianten angegeben werden
3. variante Teile (B)
   Eine Auswahl in { }, mit **|** Zeichen getrennt. Eine oder mehrere der definierten Varianten muss angegeben werden.
4. optionale Teile
   Sie können, aber müssen nicht angegeben werden. Sie stehen in eckigen Klammern **[ ]**.
5. Parameter
   Sie bestehen aus Zahlen,Formeln oder Namen, die definiert werden müssen. Der Platzhalter ist in grösser-kleiner Zeichen **<>** eingeschlossen.

3.3. Der Kommentar

```
Syntax: ; [ein Kommentar]
```

Kommentare werden durch einen Strichpunkt eingeleitet und gehen bis zum Zeilenende.

Beispiele:

```
; Dies ist ein Kommentar
```

3.4. Das $-Zeichen

Syntax: **$** [Fortsetzung der letzten Zeile]

Es gilt die Regel "pro Befehl eine Zeile". Wenn die Zeile zu kurz ist, kann die nächste Zeile an die vorhergehende angehängt werden, indem man ein $-Zeichen an den Anfang stellt. Das Beispiel veranschaulicht die Wirkung von Kommentaren.

Beispiele:

```
.EING erster_Eingang zweiter_Eingang dritter_Eingang ; 3 Eingänge
$ vierter_Eingang ; noch ein Eingang
```

dies ist identisch mit:

```
.EING erster_Eingang zweiter_Eingang dritter_Eingang vierter_Eingang
```

3.5. .FORM Befehl

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

Symbol: 

Dieser Befehl gibt an, welche Form berechnet werden soll. Dieser Befehl ist erforderlich. Es können mehrere Berechnungsformen gleichzeitig angegeben werden.

DNF	Berechnung der disjunktiven Normalform
KNF	Berechnung der konjunktiven Normalform
ANF	Berechnung der alternativen Normalform
NAND	Berechnung in der disjunktiven Form und Ausgabe mit NAND Gattern
NOR	Berechnung in der konjuktiven Form und Ausgabe mit NOR Gattern

Wenn das berechnete Resultat zu kompliziert ist, so sollte man die andere Normalform ausprobieren. Oft ergeben sich dabei überraschend einfache Resultate. Ebenso kann man vorgehen, wenn die Umsetztabellen überlaufen.

Beispiele

```
.FORM DNF ; disjunktive Normalform
.FORM KNF ANF ; konjunktive und alternative Normalform
```

3.6. .EING Befehl

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

Symbol: 

Dieser Befehl definiert die Anzahl und die Namen der Eingangsvariablen. Es können bis zu 16 Eingangsvariablennamen angegeben werden. Die Reihenfolge ist die gleiche wie für den Wertetabelleneintrag. Um Verwechslungen mit den beim .GLEICHUNG-Befehl verwendeten Operatoren auszuschliessen, sollten diese Operatoren nicht als Variablennamen verwendet werden

Beispiele

```
.EING A0 A1 WR ; Es wird mit 3 Eingangsvariablen gerechnet: A0, A1, WR
```

3.7. .AUSG Befehl

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

Symbol: 

Dieser Befehl definiert die Anzahl und die Namen der Ausgangsvariablen. Es können bis zu 16 Ausgangsvariablennamen angegeben werden. Die Reihenfolge ist die gleiche wie für den Wertetabelleneintrag. Um Verwechslungen mit den beim .GLEICHUNG-Befehl verwendeten Operatoren auszuschliessen, sollten diese Operatoren nicht als Variablennamen verwendet werden

Beispiele

```
.AUSG en Sel ; Es wird mit 2 Ausgangsvariablen gerechnet: en und Sel
```

3.8. .RESULTAT Befehl

Syntax: **.RESULTAT** { SCHEMA | GLEICHUNG | TABELLE }

Symbol: 

Dieser Befehl gibt an, wie das Resultat ausgegeben werden soll. Es gibt drei Möglichkeiten:

GLEICHUNG	Das Resultat wird in Form einer boolschen Gleichung ausgegeben (s. unten). Dies ist der Defaultwert.
SCHEMA	Das Resultat wird in Form eines Schemas ausgegeben, dabei werden Schaltsymbole nach IEC verwendet
TABELLE	Es wird eine vollständige Wertetabelle für alle Ein- und Ausgänge berechnet; d.h. alle Bezüge zwischen dem .GLEICHUNG-Befehl, Don't care-Ausgangszustand und X-Zeichen werden aufgelöst . Diese Tabelle kann bei vielen Eingangsvariablen sehr lang werden!.

Die Resultatformen können beliebig kombiniert werden, es muss jedoch mindestens eine angegeben werden.

Wenn das Resultat als Gleichung ausgegeben wird, so ist diese folgendermassen zu interpretieren:

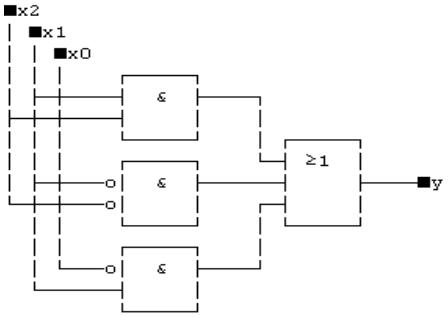
```
/A bedeutet: not A

A * B bedeutet: A and B

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

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

Im Modus Schema sieht das Ergebnis zum Beispiel so aus:



Spezialfall: Wenn ein Ausgang unabhängig von den Eingangsvariablen ist, dann wird der Ausgangsvariable eine logische 1 (*log.1*) oder eine logische 0 (*log.0*) zugeordnet. Diese beiden kursiven Bezeichnungen sollten daher, um eine Verwechslung zu vermeiden, nicht als Name für eine Eingangsvariable verwendet werden.

Wenn der .RESULTAT-Befehl nicht angegeben wird, so wird das Resultat als Gleichung ausgegeben.

Beispiele:

```
.RESULTAT SCHEMA ; Resultat als Schema
.RESULTAT SCHEMA GLEICHUNG TABELLE ; Alle möglichen
; Resultatformen
```

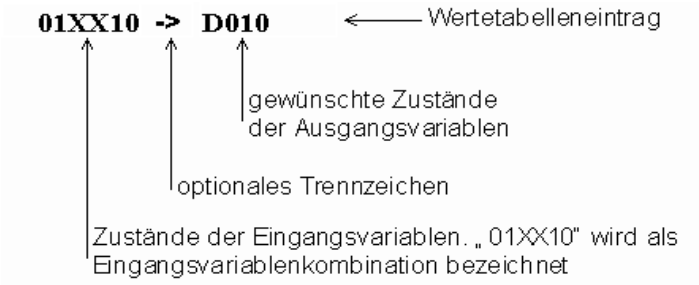
3.9. Der Wertetabelleneintrag

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

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

Symbol: 

Erläuterung der Begriffe:



1 und **0** stehen für die entsprechenden logischen Pegel der Eingangsvariablen. Es müssen soviele Eingangsvariablen angegeben werden, wie mit dem .EING Befehl definiert wurden.

Das Zeichen **X** steht für beide Zustände; d.h. anstelle eines Eintrags mit einem**X** können auch zwei Einträge mit einer **1** bzw. einer **0** anstelle des **X** in die Wertetabelle geschrieben werden. Ein Eintrag mit zwei **X** müsste natürlich durch vier Einträge mit **1** bzw **0** anstelle der **X** ersetzt werden (s. Beispiele).

Das Pfeil-Zeichen **->** ist ein optionales Trennzeichen zwischen den Ein- und Ausgängen. Es erhöht die Übersichtlichkeit. Die beiden Schreibweisen mit und ohne Pfeil sind von der Funktion her gleichwertig.

Die Buchstaben **1**, **0** und **D** am Ende der Zeile geben an, was die Eingangsvariablenkombination an den Ausgängen bewirken soll. Es müssen soviele Ausgangsvariablen angegeben werden, wie mit dem .AUSG Befehl definiert wurden. Dies sind die Bedeutungen der drei möglichen Zeichen:

1	Der Eintrag bewirkt eine logische **1** am entsprechenden Ausgang.
0	Der Eintrag bewirkt eine logische **0** am entsprechenden Ausgang.
D	Dies ist ein wählbarer Eintrag (**D**on't care), er darf am entsprechenden Ausgang sowohl eine logische Eins, wie auch eine logische Null bewirken (je nachdem was die einfachere Gleichung ergibt).

Es spielt übrigens keine Rolle, ob die einzelnen Zeichen des Wertetabelleneintrags durch Leerzeichen getrennt sind oder nicht; einzig die Reihenfolge ist entscheidend.

Beispiele

```
01 -> 1 ; bewirkt eine log. 1 am 1. Ausgang

11xx1 -> d0 ; 1. Ausgang: Don't care Kombination,
            ; Zustand der 3. und 4. Eingangsvariable
            ; beliebig. 2. Ausgang: bewirkt eine
            ; log. 0 am 2. Ausgang, sonst wie
            ; 1. Ausgang.

1100 1 -> d0
1101 1 -> d0
1110 1 -> d0
1111 1 -> d0 ; identisch mit vorigem Beispiel, aber
             ; ohne X ausgedrückt.

xxxx -> 0111 ; jede Variablenkombination erzeugt
             ; log. 0 am 1. Ausgang und log. 1 am
             ; 2. bis 4. Ausgang.
```

Das letzte Beispiel ist aequivalent zum Befehl

```
.REST 0111 ; gleiche Bedeutung wie oben wenn keine
           ; Wertetabelle angegeben wird.
```

Es muss darauf geachtet werden, dass nicht in einem Eintrag eine bestimmte Variablenkombination log. 0 und in einem anderen Eintrag die gleiche Kombination log. 1 an einem Ausgang erzeugen soll. Dies kann bei Verwendung des **X**-Zeichens leicht passieren. Einzig Don't care Einträge können so überstimmt werden. zB:

```
1111 -> 0
xxxx -> 1 ; ist unzulässig, da 1111 zuerst als log. 0, dann
          ; in xxxx versteckt als log. 1 definiert wurde.

Aber:

101010-> 1
xxxxxx -> d ; ist zulässig, da log. 1 (oder log. 0) gegenüber
            ; dem don't care Eintrag Vorrang hat. 101010 wird
            ; hier als log. 1 und der Rest als don't care
            ; definiert. Dadurch wird eine einfache
            ; Schreibweise möglich.
```

Wichtig:

Der Wertetabelleneintrag hat gegenüber einem .GLEICHUNG und einem .REST -Befehl Vorrang

Abgekürzte Schreibweise mit Hexziffern:

Eingangsvariablenkombinationen und Ausgangsvariablenzustände die nur aus den Buchstaben **1**, **0** bestehen können auch in Hexadezimalschreibweise angegeben werden. Hexadezimalzahlen werden am Suffix **h** oder **H** erkannt:

```
01111 -> 0101
10000 -> 1100

ist identisch mit:

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

Bei *n* Eingangsvariablen und *m* Ausgangsvariablen werden jeweils die *n* niedrigsten Bits der Eingangsvariablenkombination und die *m* niedrigsten Bits der Ausgangsvariablenkonbination berücksichtigt. Fehlende führende Bits werden mit 0 aufgefüllt.

Beispiel:

Bei 5 Eingangsvariablen sind folgende Angaben identisch:

```
01111 -> 0101
Fh  -> 5h      ; 5. Bit wird mit 0 ergänzt
0Fh -> 5h      ; 3 führende Bits werden ignoriert
8Fh -> 5h      ; dito.
```

3.10. .REST Befehl

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

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

Symbol: 

Alle nicht in der Wertetabelle aufgeführten Variablenkombinationen können mit diesem Befehl auf einen bestimmten Zustand gelegt werden. Die Bedeutung der möglichen Parameter ist gleich der im Kapitel "3.9. Der Wertetabelleneintrag" verwendeten Suffixe für die Wertetabelleneinträge. Dieser Befehl wirkt wie wenn alle fehlenden Einträge in die Tabelle aufgenommen würden. Es müssen genau so viele Parameter angegeben werden, wie Ausgangsvariablen definiert sind.

Wichtig:

Dieser Befehl kann nicht sinnvoll gleichzeitig mit dem .GLEICHUNG-Befehl verwendet werden, da der .GLEICHUNG Befehl Vorrang hat. Wenn weder dieser Befehl noch ein .GLEICHUNG-Befehl für eine bestimmte Ausgangsvariable verwendet wurde, so werden die nicht aufgeführten Einträge als don't care definiert.

Beispiel:

```
.REST D0D ; nicht aufgeführte Einträge sind beim 1. und
          ; 3. Ausgang don't care Einträge, beim 2. Ausgang
          ; sollen sie log. 0 ergeben.
```

Analog zu den Wahrheitstabelleneinträge können Zustände die nur aus den die nur aus den Buchstaben **1**, **0** bestehen auch in Hexadezimalschreibweise angegeben werden. Hexadezimalzahlen werden am Suffix **h** oder **H** erkannt:

Beispiel:

```
.REST 1111110

ist identisch mit:
.REST 7Eh
```

3.11. .GLEICHUNG Befehl

Syntax: **.GLEICHUNG <Ausgangsvariable> =<Formel>**

Dieser Befehl wirkt ähnlich wie der .REST-Befehl. Anstelle eines festen Wertes für nicht aufgeführte Wertetabelleinträge wird eine Formel ausgewertet und deren Resultat dem Eintrag zugeordnet. Ein .GLEICHUNG-Befehl enthält jeweils eine Formel für eine Ausgangsvariable; d.h. bei mehreren Ausgangsvariablen muss dieser Befehl gegebenenfalls mehrmals verwendet werden.

In der Formel werden die Eingangsvariablen mit Konstanten und Operatoren verknüpft und das Ganze dann einem Ausgangsvariablen zugeordnet. Es gilt die normale Infix-Notation.

Es sind zwei **Konstanten** definiert:

1	Konstante für logisch 1.
0	Konstante für logisch 0.

Gültige **Operatoren** sind:

( )	Klammern. Sie können verwendet werden, um die Operatorenpriorität zu umgehen
/ , NOT	Invertiert die nachfolgende Variable oder den nachfolgenden Klammerausdruck.
* , AND	Und-Verknüpfung
NAND	Und-Verknüpfung mit anschliessender Invertierung des Resultats.
+ , OR	Oder-Verknüpfung
NOR	Oder-Verknüpfung mit anschliessender Invertierung des Resultats.
XOR, EXOR	Exklusiv-Oder-Verknüpfung oder Antivalenz-Verknüpfung
XNOR, EXNOR	Aequivalenz-Verknüpfung

Die Priorität der Operatoren nimmt in der Tabelle von oben nach unten ab. Die Operatoren können natürlich nicht als Variablennamen (.EING / .AUSG-Befehl) verwendet werden!

Beispiele:

```
.EING a0 a1 a2 a3 ; 4 Eingangsvariablen
.AUSG en ; 1 Ausgang
.GLEICHUNG en = ( (not a0) and (not a2) ) or a3
   ; en wurde für alle nicht in der Wertetabelle
   ; vorkommenden Einträge wird diese Gleichung verwendet.
   ; Wegen der Operatorenpriorität könnten in diesem Beispiel
   ; alle Klammern weggelassen werden.

0000 -> 1 ; Wenn alle Eingänge gleich sind, soll en log.1 werden
1111 -> 1 ; Diese Einträge stellen quasi die Ausnahmen für
   ; obige Formel dar.
```

Die Wertetabelle hat immer Vorrang. Die Gleichung wird nur ausgewertet, wenn kein entsprechender Eintrag in der Wertetabelle vorkommt.

Die folgende Kombination von .GLEICHUNG-Befehlen wirkt gleich wie der .REST-Befehl darunter. Die Reihenfolge der .GLEICHUNG-Befehle ist übrigens egal:

```
.GLEICHUNG Ausgang1 = 1 ; Der 1. Ausgang soll 1 ergeben
   ; Der 2. Ausgang hat keine Gleichung; er ergibt daher
   ; don't care.
.GLEICHUNG Ausgang3 = 0 ; Der 3. Ausgang soll 0 ergeben
```

und hier das gleiche mit dem .REST-Befehl:

```
.REST 1D0 ; Gleiche Bedeutung wie oben
```

Wichtig:

Dieser Befehl kann NICHT sinnvoll gleichzeitig mit dem .REST-Befehl verwendet werden, da der .GLEICHUNG-Befehl Vorrang hat. Wenn weder ein .REST-Befehl noch dieser Befehl für eine bestimmte Ausgangsvariable verwendet wurde, so werden die nicht in der Wertetabelle aufgeführten Einträge als don't care definiert.

3.12. .PI Befehl

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

Dieser Befehl bestimmt die Art und Weise wie aus den berechneten Primimplikanten die minimal notwendigen ausgesucht werden. Dieser Befehl ist optional, wird er nicht angegeben, dann gilt die Einstellung SCHNELL. Die Parameter haben folgende Bedeutung:

EXAKT	Es wird eine vollständige Berechnung durch Ausmultiplizieren der Bitfelder der Primimplikanten vorgenommen. Dies dauert eventuell sehr lange und braucht viel Speicher, dafür werden sicher alle Lösungen gefunden.
SCHNELL, HEURISTISCH (Default)	Die Auswahl wird mit einem heuristischen Verfahren vorgenommen. Dieses findet sehr schnell Lösungen, aber es ist nicht sicher, dass die minimale Lösung schnell gefunden wird. Unter Umständen werden zuerst viele nicht minimale Lösungen ermittelt. Das Verfahren liefert manchmal die minimalste Lösung nicht. Im Gegensatz zum exakten Verfahren kann die Suche nach weiteren Lösungen abgebrochen werden - es werden dann die bereits gefundenen Lösungen ausgegeben.

4. Resultate

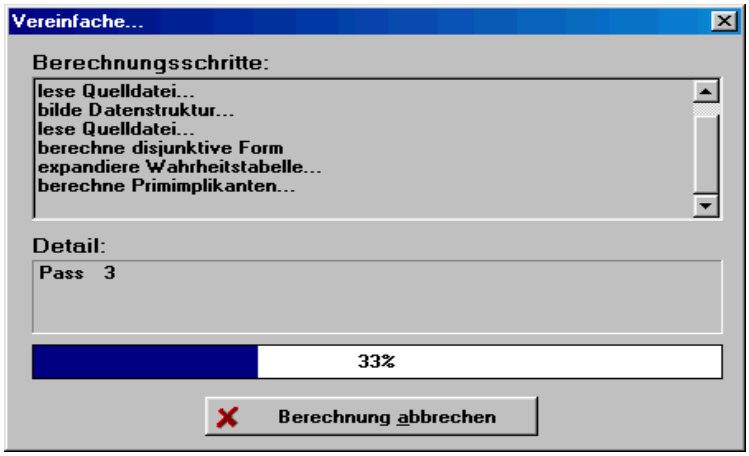
Die Ausgabe erfolgt in das Resultatfenster. Wenn der Quelltext schon in einer Datei gespeichert wurde, so wird das Resultat zusätzlich in eine Datei desselben Namens aber mit der Endung '.OUT' gespeichert..

4.1. Berechnung der Resultate

`Symbol:` 

Durch Wahl des Menupunktes Berechnung wird die Vereinfachung gestartet. Während der Ausführung werden die ablaufenden Aktionen angezeigt. Die folgende Beschreibung setzt voraus, dass das Verfahren Quine-McCluskey bekannt ist.

Der Fortschritt der Berechnung wird in diesem Dialogfeld angezeigt:



Die Berechnung kann jederzeit mit "Berechnung abbrechen" beendet werden.

4.2. Ablauf der Berechnung

Zuerst wird die interne Datenstruktur aufgebaut.

bilde Datenstruktur...

Welcher Ausgang wird berechnet?

Berechnung für 1. Ausgang

Als nächstes wird die Quelldatei eingelesen.

lese Quelldatei...

Eine Meldung zeigt die Berechnungsform an...

Berechnung für DNF

Beim nun folgenden Schritt werden alle Wertetabelleneinträge mit X-Zeichen durch die ausgeschriebene Form des Tabelleneintrags ersetzt:

expandiere Wertetabelle...

Nun werden die Primimplikanten gebildet. Die Prozentanzeige gibt an, wieviel des jeweiligen Durchgangs (pass) schon berechnet ist. Die Anzahl Durchgänge ist von Fall zu Fall verschieden. Es kann sehr lange dauern!

berechne Primimplikanten...
pass 1 100%
pass 2 100%
pass 3 100%

Redundante Primimplikanten werden im nächsten Schritt eliminiert. Da das Endresultat von der Reihenfolge der Elimination abhängt, werden mehrere Anläufe genommen; nämlich soviele wie überhaupt möglich sind. Auf diese Art werden alle Lösungen gefunden. Ein nachgeschaltetes "Filter" löscht Lösungen, die nicht absolut minimal sind.

suche redundante Primimplikanten...

pass 1
pass 2
pass 3 Lösung ist redundant <- Das Filter hat zugeschlagen !

Am Schluss werden alle verbliebenen Resultate in die Resultatdatei oder auf den Bildschirm ausgegeben.

Schreibe Resultat nach DEC.OUT...
nicht minimale Lösung gelöscht... <- Filterwirkung
Schreibe Resultat nach DEC.OUT...

Der beschriebene Vorgang wiederholt sich für jede Ausgangsvariable. Die Resultate werden anschliessend in das schwarze Editorfeld geladen.

4.3. Grenzen

Die Grenzen liegen vor allen in der Anzahl erlaubter Variablen und in der Grösse der intern verwendeten Tabellen. In der Version 3.x sind 16 Variablen erlaubt. Die Tabellen zur Berechnung der Primimplikanten fassen ca. 65000 Einträge. Die Tabelle zur Ermittlung der redundanten Primimplikanten fasst maximal 65000 Primimplikanten. Die genannten Tabellengrössen sind nur voll ausnutzbar, wenn 256 MByte oder mehr Speicher verfügbar ist. Der minimale Speicherbedarf beträgt ca. 32 MByte. Wenn wenig RAM verfügbar ist, so werden intensiv temporäre Zwischendateien erzeugt, was einen markanten Geschwindigkeitseinbruch ergibt.

5. Wenn es nicht funktioniert

5.1. Fehlermeldungen

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. Fehler im Quelltext

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. Fehler während der Berechnung

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 und 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. Anwendungsbeispiele

6.1. Beispiel 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. Beispiel 2

In einem Mikroprozessorsystem soll ein Freigabesignal für eine externe Einheit erzeugt werden. Dabei soll folgende Wertetabelle gelten:

Eingänge notReset En WR A0	Ausgang Freigabe
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

Die Tabelle wird in eine Quelldatei umgesetzt. Durch geeigneten Einsatz des **X**-Zeichens konnten ein paar Einträge gespart werden. Diese Einsparung hat aber keinen Einfluss auf die Berechnung, sie ist etwas für Schreibfaule (Das Beispiel ist etwas zu einfach und zu klein, um die Mächtigkeit des X-Zeichens zu demonstrieren).

Das Resultat aller Bemühungen ist die Quelldatei: DEMO2.QMC.