Langzeiterfassung von Messwerten in Forth

Claus Kühnel*

In der Langzeiterfassung von Messwerten müssen abzuspeichernde Daten komprimiert werden. Es existieren verschiedene Grenzwertverfahren, um die Datenmengen zu reduzieren. Eines davon wird hier angewendet und in Forth realisiert.

1. Einleitung

Die Erfassung und Überwachung von Daten mit niedriger Dynamik, wie sie unter anderem in der Umwelt- oder der Verfahrenstechnik vorkommen, erfordert in vielen Fällen eine Registrierung über einen längeren Zeitraum.

Bedingung für eine solche Erfassung ist nur, dass die betreffenden Sensorsignale in ein vom AD-Umsetzer verarbeitbares Spannungssignal umgeformt werden können. Dieses spezielle Thema soll im folgenden nicht weiter betrachtet werden. Die Applikationsschrift der Sensor- und Schaltkreishersteller liefern hier umfangreiche Unterstützung.

Mit der Langzeiterfassung von Messwerten taucht sofort das Problem der entstehenden Datenmengen auf. Im allgemeinen sind die erhobenen Daten auf einen externen Datenträger abzulegen, damit eine Archivierung und spätere Behandlung des Datenmaterials gesichert ist. Zwar ist das Abspeichern umfangreicher Datenmengen bei den heutigen Festplattenkapazitäten kein generelles Problem, sicher ist andererseits das Abspeichern weitgehend redundanten Datenmaterials nicht der Zweck der Übung.

Am Beispiel der Temperaturmessung soll das Anliegen verdeutlicht werden. Bei meteorologischen Erhebungen werden beispielsweise Tagesprofile (24 Stunden) der Temperatur von Luft und/oder Wasser aufgenommen. Wird alle 10 s ein Messwert erhoben, dann entstehen innerhalb von 24 Stunden immerhin 8640 abzuspeichernde Daten. Die Dynamik in einem solchen Tagesprofil ist aber durch wesentlich weniger Daten ohne Informationsverlust darstellbar. Sinnvoll erscheint es deshalb nur, die Daten abzuspeichern, die dann auch zur Charakterisierung des Messwertverlaufs herangezogen werden.

2. Komprimierung der abzuspeichernden Daten

In Systemen der Messwerterfassung werden eine Reihe intelligenter Grenzwertverfahren zur Reduzierung der erhobenen Datenmengen eingesetzt [1].

Die hier verwendete Variante der Datenreduzierung soll anhand von Bild 1 erklärt werden.

Es sei ein beliebiger Messwerteverlauf gegeben. Beginnend mit der Abspeicherung des ersten Messwertes wird ein Toleranzband symmetrisch zu diesem ersten Messwert festgelegt. Alle folgenden Messwerte, die innerhalb dieses Toleranzbandes liegen, werden nicht abgespeichert. Beim Auftreten eines ausserhalb des Toleranzbandes liegenden Messwertes erfolgt nunmehr dessen Abspeicherung sowie die erneute Festlegung des Toleranzbandes symmetrisch zu diesem Wert. Das Festlegen der Breite des Toleranzbandes ist abhängig von der messtechnischen Fragestellung und muss so erfolgen, dass nur signifikante Änderungen der Messgrösse zum Verlassen des Toleranzbandes führen können.

Auch hier soll wieder die Temperaturmessung zur Verdeutlichung der Aussage bemüht werden. Mit Hilfe eines integrierenden 12-Bit-AD-Umsetzers, wie er kommerziell als PC-Einsteckkarte angeboten wird, soll beispielsweise die Lufttemperatur im Bereich von –40 °C bis +60 °C überwacht werden. Die Auflösung (LSB – Least Significant Bit) des AD-Umsetzers beträgt unter den gegebenen Bedingungen 0,0244° (±0,5 LSB). Definiert man nun ein Toleranzband von ±5 ADU-Counts (11 LSB), dann muss sich die Temperatur zusätzlich zur Quantisierungsunsicherheit von 0,0244° (±0,5 LSB) um weitere 0,122° (5 LSB) vergrössern oder verkleinern, um eine Abspeicherung auszulösen. Zur Lösung der hier diskutierten Fragestellung ist die resultierende Auflösung von ≈0,1 °C sicher eine vernünftige Grössenordnung.

3. Beschreibung des Programmes

Als interaktive Hochsprache stellt Forth ein geeignetes Sprachkonzept (nicht nur) für Zwecke der Messwerterfassung und -verarbeitung dar. Hinzu kommt, dass moderne Forth-Versionen heute bereits mit einem Multi-Tasker versehen sind, dessen Anwendung sehr einfach ist.

Das Programm zur Langzeiterfassung und komprimierten Abspeicherung von Messwerten wurde in der MS-DOS-Version von Laxen & Perrys **F83** geschrieben. Es soll das Prinzip verdeutlicht werden, dessen Verständnis dann die Anpassung an konkrete Aufgabenstellungen erlaubt.

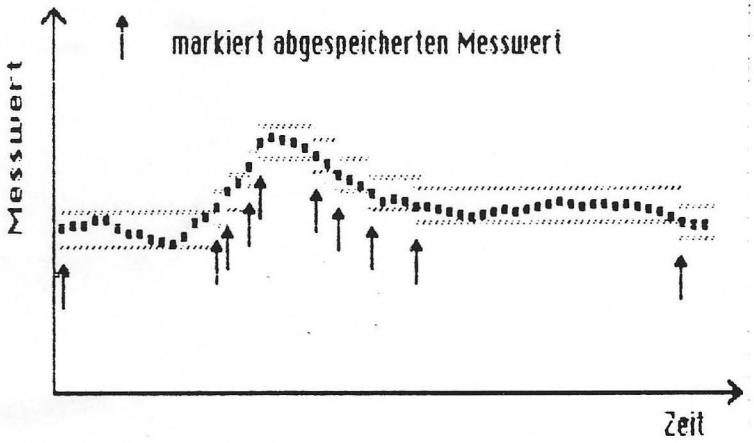

Bild 1 Grenzwertverfahren zur Reduzierung von Datenmengen in der Langzeit-Messwerterfassung.

* Dr. Claus Kühnel, ~~Zschertnitzerstr. 52, DDR-8020 Dresden~~

konkreten AD-Umsetzer. An dieser Stelle ist dann der konkrete Treiber zuzuweisen. Wie das vorgenommen wird, ist als Kommentar in Screen # 10 bereits enthalten. Die geschilderte Art der Vektorisierung ist stark verbreitet, aber nicht einheitlich. Die Grundlagen zur Vektorisierung sind in [3] zusammengestellt.

Neben einigen Variablen und einem kleinen Eingabepuffer zur Zahleneingabe ist in Screen # 11 das Wort **zahl_ein** zu finden. Mit diesem Wort soll eine halbwegs intelligente Eingabe formuliert werden. Das Wort erwartet eine Zahl auf dem Stack, die als Vorgabe gewertet wird. Im Fall einer gültigen Eingabe steht auf dem Stack dann der aktuell eingegebene Wert, ansonsten verbleibt dort der vorgegebene. Gibt man nun die Vorgabewerte geschickt vor, ist an vielen Stellen nur eine Bestätigung durch die RETURN-Taste vonnöten.

Von dieser Art der Zahleneingabe wird dann auch gleich im Wort **parameter_eingabe** in Screen # 12 Gebrauch gemacht. Die eingegebenen Werte werden dann sofort in der in Screen # 0 gezeigten Weise auf die Diskette geschrieben.

Die Worte **toleranz?, abspeichern** und **programm_ende** in Screen # 13 gehören nun schon zur Verarbeitung der erhobenen Daten.

Die eigentliche Messwerterfassung steht nun in Screen # 14. Der aktuelle Messwert wird erfasst und auf dem Bildschirm ausgegeben. Es schliesst sich die Ausgabe der Zahl der erhobenen Daten an, bevor überprüft wird, ob eine der Alarmgrenzen überschritten wird. Ein solcher Alarm macht sich hier durch ein akustisches Signal bemerkbar. Anschliessend wird die Einhaltung der Toleranz überprüft. Wird diese überschritten, erfolgt die Abspeicherung von Index und Messwert sowie die Neufestlegung des Toleranzbandes.

Das einzige Nutzerwort, was nach aussen hin eigentlich interessiert, ist das in Screen # 15 enthaltene Wort **doit.** Es organisiert die Initialisierung und Eingabe der Parameter, startet den Multi-Task-Betrieb und «erweckt» die Background-Task zum Leben. Die Messwerterfassung erfolgt in einer Endlos-Schleife, bis dass irgendwann die Betätigung einer beliebten Taste diese Schleife unterbricht. Die Aktualisierung der angezeigten Zeit geschieht durch die Background-Task im Verborgenen.

Nach Aufruf des Forth-Systems aus der Betriebssystemebene ist die Kompilierung des im File LZ.BLK abgelegten Quelltextes durch Eingabe von **open lz.blk 1 load** vorzunehmen. Das Wörterbuch des Forth-Systems wird um ganze 2865 Byte erweitert, wodurch die Kompaktheit des erzeugten Codes hinreichend dokumentiert sein dürfte.

Nach einer Wartezeit von ≈0,5 s startet das Programm selbst. Nach Programmende befindet man sich weiterhin im Forth-System.

Rettet man die Applikation als COM-File, wird man nach Programmende wieder in die Betriebssystemebene gelangen wollen. In Screen # 13 sind hierfür nur die Klammern um das im Wort **programm_ende** enthaltene **bye** zu löschen. Nach erneuter Kompilierung ist die Applikation mit **save-system lz.com** auf die Diskette zu schaffen.

Im Fehlerfall, z. B. **open error in...** bei nicht vorhandenem File, verbleibt man dennoch im Forth-System und kann interaktiv beispielsweise mit den Worten **drive?, a:, b:, c:, dir** hantieren. Ein Neustart der Applikation kann dann mit **doit** oder **boot** erfolgen.

Literatur

[1] Neidig M.:
Intelligente Grenzwertverfahren.
Elektronik, München 35 (1987) 2, S. 87 u. 88

[2] Zech R.:
Forth 83.
Franzis-Verlag München 1987

[3] Kühnel C.:
Vektorisierung in Forth.
Elektroniker, Aarau (1989) 5

```
      4
0 \ file_aktivieren                                  ck 28aug89
1
2 : file_aktivieren   ( -- )
3    [ dos ]
4    "         .DAT" fcb1 (!fcb)
5    ." Eingabe des Dateinamens [ohne Extension]: "
6    fcb1 1+ 8 expect              \ Eingabe Dateinamen
7    fcb1 1+ 8 upper               \ Umwandlung in Grossbuchstaben
8    fcb1 !files
9    open-file init_fcb
10 ;
11
12
13
14
15
```

```
      5
0 \ sound_on sound_off sound alarm                   ck 28aug89
1 hex
2 : sound_on    ( -- )   61 pc@ 03 or  61 pc! ;
3 : sound_off   ( -- )   61 pc@ FC and 61 pc! ;
4 decimal
5 : sound   ( u -- )
6    1 ?enough base @ >r decimal
7    sound_on
8    dup 37 < if drop 37 then
9    1193180. rot m/mod nip dup
10   [ hex ] B6 43 pc! FF and 42 pc! 100 / FF and 42 pc!
11   [ decimal ] r> base !
12 ;
13 : alarm   ( -- )
14    4000 sound 20 ms sound_off   \ veranderter Piepton
15 ;
```

```
      6
0 \ BIOS Timer                                       ck 25aug89
1
2 hex    46C constant counter   decimal
3
4 code ticks   ( -- u )
5    0 # ES mov   counter # BX mov   ES: 0 [BX] push   next
6 end-code
7
8 : timeout? ( ticks -- ticks f ) pause dup ticks - 0< ;
9 : till     ( n -- )        BEGIN  timeout? UNTIL  drop ;
10 : time    ( n -- time )   ticks + ;
11 : wait    ( n -- )        time till ;
12 : seconds  ( sec -- ticks )  18206 1000 */ ;
13 : minutes  ( min -- ticks )  1092 * ;
14
15
```

```
      7
0 \ time@ at?                                        ck 28aug89
1
2 hex
3
4 code time@   ( -- hs ss mm hh ) \ DOS Interrupt 21-2C
5    2C # AH mov   21 int   AH AH sub   DL AL mov   AX push
6    DH AL mov   AX push   AH DH mov   CL DL mov   CH AL mov
7    2push
8 end-code
9
10 code at?   ( -- col row )      \ DOS Interrupt 10-03
11    3 # AH mov   0 # BH mov   10 int
12    DH AL mov   AH AH sub   DH DH sub   2push
13 end-code
14
15 decimal
```

```
      12
\ parameter_eingabe                                  ck 25aug89

: parameter_eingabe   ( -- )
   0 6 at file_aktivieren
   0 6 at ." Daten werden in Datei " file? ." abgespeichert."
   0 8 at ." Erfassungstakt (sec.): " 1 zahl_ein takt !
   0 10 at ." Toleranzband (+/-): " 2 zahl_ein toleranz !
   toleranz @ takt @ doppelwort_schreiben
   0 12 at ." minimaler Alarmpegel: " 502 zahl_ein min_alarm !
   0 13 at ." maximaler Alarmpegel: " 522 zahl_ein max_alarm !
   min_alarm @ max_alarm @ doppelwort_schreiben
;
```

```
      13
\ toleranz? abspeichern programm_ende                ck 28aug89
: toleranz?   ( n -- f )
   dup lotol @ > swap hitol @ < and not
;
: abspeichern   ( n -- )
   dup index @ doppelwort_schreiben
   toleranz @ 2dup + hitol ! - lotol !
   1 anzahl +!
   0 20 at ." Zahl der abgespeicherten Daten: " anzahl ?
;
: programm_ende   ( -- )
   single dark beep
   [ hex ] ffff ffff doppelwort_schreiben [ decimal ]
   0 10 at index ? ." Daten erfasst." cr cr
   anzahl ? ." Daten in File " file? ." abgespeichert." cr cr cr
   [ dos ] file @ close ." Programm beendet." ( bye ) ;
```

Forth 83 Model

```
      14
\ wert_erfassen var_init                             ck 28aug89

: werte_erfassen   ( -- )
   adu
   0 16 at ." aktueller Messwert: " dup .
   1 index +!              \ Index incrementieren
   0 18 at ." Zahl der erhobenen Daten: " index ?
   dup min_alarm @ max_alarm @ between not if alarm then
   dup toleranz? if abspeichern else drop then
   takt @ seconds wait
;
: var_init   ( -- )          \ Initialisierung der Variablen
   index off          anzahl off
   hitol off          lotol off
;
```

```
      15
\ doit                                               ck 28aug89

: doit   ( -- )
   dark cr ." **** Langzeiterfassung von Messwerten ****"
   var_init parameter_eingabe
   65 0 76 2 rahmen
   15 24 at ." Abbruch der Erfassung durch beliebige Taste "
   multi .time wake            \ Multi-Task-Betrieb
   begin
      werte_erfassen           \ Messwerterfassung bis zur
   key? until                  \ Betaetigung einer Taste
   programm_ende
;

' doit is boot                 \ Selbststart des COM-Files
                               \ sonst nur DOIT
```

Tabelle 1 Listings zur Langzeit-Messwerterfassung in Forth («Screen 1 ... 15).

File ständig überwacht. Ist die Kapazität erschöpft, erfolgt die Vergrösserung des Files um 1 KByte.

Screen # 4 enthält das Wort **file_aktivieren,** welches den Namen des Datenfiles erwartet und dieses File auch eröffnet. Die Extension DAT wird automatisch ergänzt.

Um einen vom Systembeep abweichenden Alarmton erzeugen zu können, wurden die Worte in Screen # 5 mit aufgenommen. Diese Worte sind an MS-DOS gebunden, werden aber nicht unbedingt benötigt. In einem solchen Fall wäre das Wort **alarm** durch die Definition **: alarm beep ;** zu ersetzen.

In Screen # 6 sind die Bestandteile zur Nutzung der Timerfunktionen enthalten. Gelesen wird hier die Zelle 40:6C, die durch den Ticker ständig erhöht wird. Mit den so definierten Worten lassen sich beliebige Wartezeiten erzeugen. Eine Zeitverzögerung von 5 s erreicht man beispielsweise durch **5 seconds wait.**

Mit dem Wort **time@** in Screen # 7 wird über den DOS-Interrupt 21-2C die DOS-Zeit auf den Stack gelegt. Die aktuelle Cursorposition wird im Wort **at?** mit dem DOS-Interrupt 10-03 ermittelt. Diese beiden Worte sind, wie auch schon das Wort **ticks,** Code-Definitionen, die mit dem F83-Assembler erzeugt wurden. An diesen Stellen gibt es bei den verschiedenen Forth-Versionen mitunter beträchtliche Unterschiede.

In Screen # 8 erfolgt schliesslich die Definition der Zeitausgabe als Background-Task. Damit eine flackerfreie Anzeige garantiert ist, verhindert das Wort **single** während der Ausgabe der Uhrzeit eine Task-Umschaltung (Single-Task-Betrieb). Erst wenn die Zeit komplett ausgegeben ist, kann der Multi-Task-Betrieb wieder aufgenommen werden. Tiefergehende Informationen zum Multi-Task-Betrieb kann der interessierte Leser in [2] finden.

In Screen # 9 wird mit den im Zeichensatz vorhandenen Grafiksymbolen ein Rahmen definiert.

In Screen # 10 ist ein sehr einfacher (Pseudo-)Zufallszahlengenerator [2] enthalten, mit dessen Hilfe das Wort **test** definiert wurde. Das der Messwerterfassung dienende Wort **adu** wurde nach **test** vektorisiert. Das macht sich für den Programmtest sehr gut und erfordert keinen

Da bei einer Langzeiterfassung zwangsläufig auch der Bildschirm die ganze Zeit in Betrieb ist, wurden die Bildschirmausschriften im Umfang weitgehend reduziert. Auf eine komfortable Gestaltung des Bildschirm-Layouts wurde deshalb verzichtet.
Um bei einer Langzeiterfassung eine gewisse Orientierung und bei seltenen Abtastungen (z. B. 1/min) als Indikator für die laufende Erfassung auch etwas Leben auf dem Bildschirm zu erzeugen, wurde in der rechten, oberen Bildschirmecke eine sich im Sekundentakt ändernde Zeitanzeige vorgesehen.

Einige Bemerkungen zum Listing sollen den Aufbau des Programmes erläutern. Der Kommentar-Screen (Screen # 0) enthält die Beschreibung der Organisation des erzeugten Datenfiles. Wichtig für eine Reproduktion des Wertverlaufes ist vor allem die Kenntnis der Taktrate. Die erfassten Werte werden schliesslich sequentiell abgespeichert. Wenn die Messwerterfassung durch das Betätigen einer beliebigen Taste abgebrochen wurde, dann wird das Fileende (EOF) noch durch zwei Worte FFFFFFFF(H) gekennzeichnet. Eine Routine zum Auslesen der Daten kann sich dann daran orientieren.
Screen # 1 dient, wie häufig praktiziert, als Loadscreen für die anderen Bestandteile des Files. Die Compilierung wird kommentiert, und anschliessend erfolgt der automatische Start der Applikation. Beim Abspeichern wurde hier nicht vom üblichen Blockkonzept ausgegangen, sondern ein Record von vier Bytes zur Aufnahme von zwei 16-Bit-Einträgen definiert.
In Screen # 2 wird der File-Control-Block (FCB) dann auch entsprechend initialisiert. Das Datenfile, welches durch die Applikation beschrieben werden soll, muss auf der Disk bereits vorhanden sein. In **F83** wird beispielsweise durch Eingabe von **1 create-file name.dat** ein File NAME.DAT mit einer Kapazität von 1 KByte erzeugt. Die vorgegebene Speicherkapazität ist unkritisch, da diese bei Erfordernis vergrössert wird.
Diese Vergrösserung bewirkt das Wort **re_init_fcb** in Screen # 3. Beim Abspeichern mit Hilfe des Wortes **doppelwort_schreiben** wird der freie Speicherplatz im

```
      0
 0 \ LZ.BLK                                          ck 28aug89
 1
 2 Das Programm LZ.BLK erlaubt die Erfassung von Messwerten ueber
 3 einen laengeren Zeitraum und deren komprimierte Abspeicherung.
 4
 5 Format des erzeugten Datenfiles:
 6 ! Takt ! Toleranz ! max. Alarm ! min. Alarm !
 7 ! Index1 ! Wert1 ! Index2 ! Wert2 !... ! FFFF ! FFFF !
 8
 9 Jeder Eintrag hat eine Breite von 16 Bit. Abgespeichert werden
10 stets zwei Werte.
11 Erstellung eines selbststartenden COM-files durch
12 Eingabe von "save-system lz.com".
13
14 Autor: Claus Kuehnel, Zschertnitzer Str. 52, 8020 Dresden
15 Datum: 28. 8.1989

      1
 0 \  Load-Screen                                    ck 28aug89
 1
 2 cr
 3
 4 cr .( Langzeit-Erfassung wird compiliert...) cr
 5
 6 2 15 thru
 7
 8 cr .( Langzeit-Erfassung ist compiliert.)
 9
10
11 500 ms
12
13 boot
14
15

      2
 0 \ record init_fcb @record# @maxrec#               ck 28aug89
 1
 2 2variable record              \ Index Wert
 3
 4 : init_fcb   ( -- )
 5   record [ dos ] set-dma
 6   4 file @ 14 + !             \ Record zu 4 Bytes
 7   0 file @ record# !          \ akt. Record = 0
 8   file @ dup 16 + @ 4 / swap maxrec# !
 9 ;
10 : @record#   ( -- n )
11   [ dos ] file @ record# @    \ aktueller Record
12 ;
13 : @maxrec#   ( -- n )
14   [ dos ] file @ maxrec# @    \ maximaler Record
15 ;

      3
 0 \ re_init_fcb doppelwort_schreiben                ck 28aug89
 1 : re_init_fcb   ( -- )
 2   b/buf file @ 16 + +!        \ Datei um 1 kByte erweitert
 3   file @ dup 16 + @ 4 / swap [ dos ] maxrec# !
 4   @record#
 5   b/buf 4 / 0 do              \ Loeschen der erweiterten Datei
 6     bl bl record 2! [ dos ] file @ rec-write
 7     1 file  @ record# +!
 8   loop
 9   file @ record# !
10 ;
11 : doppelwort_schreiben   ( 16b1 16b2 -- )
12   record 2! [ dos ] file @ rec-write   \ Record schreiben
13   1 file  @ record# +!                 \ Incr. akt. Record
14   @record# @maxrec# = if re_init_fcb then
15 ;

      8
 0 \ merker .time                                    ck 28aug89
 1
 2 variable merker   merker off
 3
 4 background: .time             \ Background-Task definieren
 5 begin pause single            \ Single-Task-Betrieb
 6   time@ 2 pick merker @ <>
 7   if at? >r >r 67 1 at
 8     s>d <# # # #> type ascii : emit \ Stunden
 9     s>d <# # # #> type ascii : emit \ Minuten
10     dup merker !                  \ Merker aktualisieren
11     s>d <# # # #> type            \ Sekunden
12     r> r> at
13   else 2drop drop
14   then drop multi         .      \ Multi-Task-Betrieb
15 again ;

      9
 0 \ rahmen                                          ck 28aug89
 1 201 constant loe        187 constant roe
 2 200 constant lue        188 constant rue
 3 205 constant hor        186 constant ver
 4
 5 : rahmen   ( links oben rechts unten -- )
 6   4 ?enough
 7   3 pick 3 pick at loe emit
 8   over 3 pick at roe emit
 9   3 pick over at lue emit
10   over over at roe emit
11   over 4 pick 1+ ?do i 3 pick at hor emit loop
12   over 4 pick 1+ ?do i over at hor emit loop
13   dup 3 pick 1+ ?do 3 pick i at ver emit loop
14   dup 3 pick 1+ ?do over i at ver emit loop
15   2drop 2drop ;

     10
 0 \ seed random test adu                            ck 28aug89
 1
 2 variable seed   here seed !       \ Initialisierung
 3
 4 : random   ( n1 -- n2 )
 5   seed @ 259 * 3 + 32767 and dup seed !
 6   32767 */
 7 ;
 8 : test   ( -- n )                 \ beliebiges Testwort
 9   500 24 random +
10 ;
11
12 defer adu
13
14   test is adu
15

                    Forth 83 Model

     11
 0 \ takt toleranz min_alarm max_alarm index anzahl  ck 28aug89
 1 variable takt           variable index
 2 variable toleranz       variable anzahl
 3 variable min_alarm      variable hitol
 4 variable max_alarm      variable lotol
 5
 6 create buff 6 allot           \ Puffer fuer Zahleingabe
 7
 8 : zahl_ein   ( n1 -- n2 )     \ Eingabe bzw. Bestaetigung
 9   1 ?enough dup ." [ " . " ] ? "
10   buff 1+ 5 blank
11   buff 1+ 4 expect buff number?
12   if drop swap drop
13   else 2drop then
14 ;
15
```