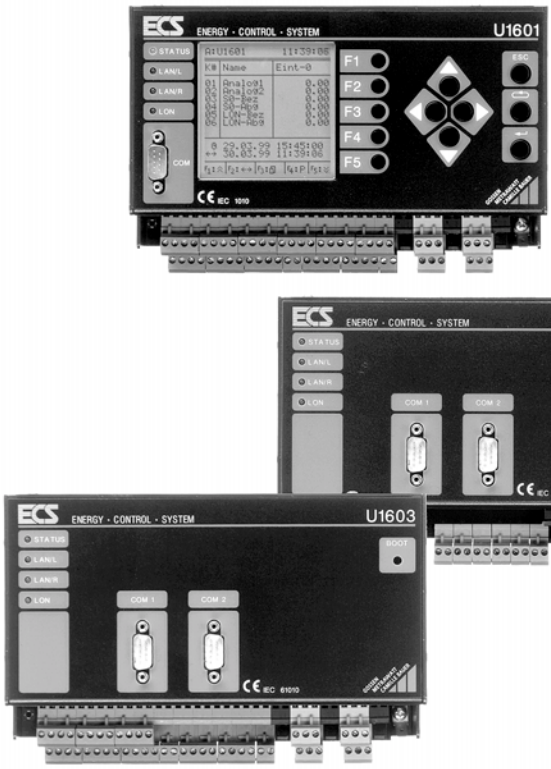
Bedienungsanleitung

GOSSEN METRAWATT

ENERGY · CONTROL · SYSTEM

ECL-Interpreter und Befehlsreferenz

3-348-870-01
3/6.02



Inhalt Seite

Stand der ECS-Betriebssoftware: V2.47

1 ECL-Interpreter

1.1 Einführung

Der ECL-Kommando-Interpreter (ECL = **E**nergy **C**ontrol **L**anguage) stellt die logische Schnittstelle zwischen einer Summenstation und dem PC (Hostrechner) oder Terminal dar. Physikalisch werden die Signale über eine serielle RS-232-Schnittstelle übertragen.

Der Informationsaustausch mit weiteren durch das ECS-LAN verknüpften Summenstationen kann so erfolgen, als ob die adressierte Station direkt mit dem PC oder Terminal gekoppelt wäre.

Die Kommunikation erfolgt mit Klartextbefehlen, das Ausgabeformat lässt sich ganz nach Wunsch datenbank- oder benutzerspezifischen Belangen anpassen. Die einzelnen Befehle lassen sich aneinanderreihen und die Abarbeitungsreihenfolge bestimmten Bedingungen unterwerfen, so dass damit eine komplette Programmiersprache zur Verfügung steht. Diese Programmiersprache nennen wir ECL- Energy Control Language.
ECL ist eine Mischung aus FORTH und BASIC. Wer sich mit der **u**mgekehrten **p**olnischen **N**otation (UPN) der HP-Taschenrechner auskennt und etwas Programmiererfahrung in BASIC hat, wird keine Schwierigkeiten mit der neuen Sprache haben.

Warum, so werden Sie sich vielleicht fragen, benötigt das ECS überhaupt eine “Hi-Level“ Programmiersprache? Einerseits verfügen die Summenstationen über virtuelle Kanäle, deren Definition (besonders bei geräteübergreifenden Bezügen) eine verständliche Notation verlangt, und andererseits ist die effiziente Programmierung der Relaisausgänge oder anderer Vorgänge nur mit einer ausgewachsenen Programmiersprache möglich. Stellen Sie sich nur vor, Sie müssen der
Summenstation 'B' erklären:

> Relais 1 der Summenstation 'B' schaltet ein, wenn die Summe der Momentanleistungen der Kanäle 1 bis 5 im Gerät 'A' plus der Kanäle 8 und 17 im Gerät 'G5' größer als 125 kW ist.

Wir schreiben (eingeloggt in die Station B:) :

```
<B> A:Pmom – 1..5, G5:Pmom – 8+17, +,125,>,IF,Rel 1=1, ELSE,Rel 1=0
```

Zur Verdeutlichung analysieren wir die einzelnen Teile dieser Befehlsfolge:
Wie oben bereits erwähnt, reihen wir Befehle aneinander. Daraus entstehen dann Befehlsfolgen. Anders als in BASIC bescheren uns diese Befehlsfolgen aber keine neuen semantischen (Semantik: die Bedeutung) Probleme, denn das Zusammenspiel der einzelnen Befehle erfolgt über den klar definierten Parameter-Stack. Dieser Stack ist ein sogenannter Keller-Speicher, d.h. man entnimmt Elemente in der umgekehrten Reihenfolge, als sie hinzugefügt wurden.

Beispiel:
Wir fügen die Elemente 1, 5, 8 und 17 in dieser Reihenfolge dem Stack hinzu. Entnehmen wir Elemente, erhalten wir zuerst die 17, dann die 8, und die 5, und schließlich die 1.
Jeder Befehl fügt “sein Ergebnis“ dem Stack hinzu oder entnimmt Elemente dem Stack. Der Additionsbefehl '+' nimmt beispielsweise 2 Elemente vom Stack, addiert diese beiden und fügt das Ergebnis dem Stack hinzu.

Der Ausgabebefehl '!' entnimmt ein Element dem Stack und „druckt“ es aus. Also:

2,5,+,! : gibt ’7’ aus (Addition von 2 + 5)
17.5;–4;3;*;+;! : gibt ’5.5’ aus (Multiplikation – 4 * 3 = – 12, dann Addition 17,5 + – 12 = 5,5)

Innerhalb eines Befehls gilt die vertraute, von BASIC bekannte Art:

Funktionsname (Argument1, Argument2; ...)

Wir erlauben uns hier einen syntaktischen Klimmzug. Klammern um die Argumente, genauso wie die Kommas zwischen den Argumenten, können (und müssen sogar) weggelassen werden.

- Das Trennzeichen zwischen Funktionsname (ECL-Befehl) und Argument (Parameter), sowie zwischen den Argumenten untereinander ist das Leerzeichen.
- Zwischen den Befehlen vereinbaren wir das Komma oder das Semikolon als Trennzeichen.

ECL-Befehl Parameter1 Parameter2 ... = Zuweisung1 Zuweisung2 ...

Jetzt erscheint obiges Beispiel schon klarer.
Der erste Befehl lautet also:

```
A:Pmom – 1..5
```

A: deutet dem Befehlsinterpreter an, diesen aktuellen Befehl an das Gerät A weiterzuleiten (gemäß unseres Beispiels nehmen wir an, dass wir gerade mit dem Gerät B kommunizieren). Das Ergebnis (hier die Summe der Momentanleistungen der Kanäle 1 bis 5) jedoch wird dem Gerät B zurückgeschickt und auf den Stack “gepusht“ (dem Kellerspeicher hinzugefügt).Der zweite Befehl

,

```
G5:Pmom – 8+17
```

“pusht“ also die Summe der Momentanleistungen des Kanals 8 und 17 von Gerät 'G5' auf den Stack

```
+, 125, >
```

Der dritte Befehl '+' addiert beide Momentanleistungssummen, anschließend “pusht“ der vierte Befehl '125' einfach die Zahl 125 auf den Stack.
Der fünfte Befehl '>' vergleicht die gebildete Summe mit 125 (..Summe > 125). Ist der Vergleich positiv, d.h. ist die ..Summe wirklich größer als 125, wird einfach eine 1 auf den Stack gepusht, ansonsten eine 0 (Null).

```
IF, Rel 1=1
```

Der sechste Befehl 'IF' entscheidet nun, ob das 1. Relais nun eingeschaltet (REL 1=1) oder

```
ELSE, Rel 1=0
```

ausgeschaltet (REL 1=0) wird.

1.2 Wertebereich, Zahlen und Zeichenketten

In den Summenstationen anfallende numerische Daten können einen recht großen Bereich umfassen. Wichtiger jedoch als die Ausdehnung des Zahlenbereiches ist die Genauigkeit, hier ausgedrückt in der Anzahl der signifikanten Dezimalstellen. 15 signifikante Stellen stehen zur Verfügung, der im Interpreter darstellbare Zahlenbereich umfasst 27 Vorkommastellen und 9 Nachkommastellen.

Hinweis

Sind 15 Stellen nicht mehr ausreichend, um eine Zahl zu repräsentieren, wird intern auf Exponentialnotation gewechselt (64-Bit-Fließkommazahl).

Diese Energie kann z.B. noch ohne Genauigkeitseinbuße verarbeitet werden:

1.234.567.890,12345 kWh

Alle Rechenoperation, die der Interpreter zur Verfügung stellt, arbeiten mit der genannten Genauigkeit.

Wir wollen diesen Datentyp REAL nennen, auch wenn der Vergleich mit Realen Zahlen etwas hinkt. Jedoch merken wir uns: Der Interpreter kennt nur diesen einen Datentyp für Zahlen, Integerwerte sind für ihn eine Untermenge von REAL (es gibt eine Ausnahme: die Aufzählung, z.B: 1..4+7).

Merke:
Der Parameter-Stack nimmt ausschließlich Elemente des Typs REAL auf.

Folgende mathematische Funktionen stehen zur Verfügung:

Grundrechenarten	**+ – * / MOD**
Boole´sche Verknüpfungen	**& | ^**
Vergleiche	**< <= == != >= >**
Umwandlungen	**INT INTR FRAC ABS**
Quadratwurzel	**SQRT**
trigonometrische Funktionen (Basis ist das Bogenmaß)	**SIN COS ASIN ACOS**
Exponentialfunktionen	**EXP LOG ****

Zeichenketten

Neben den Zahlen gibt es noch die Zeichenketten, auch “Strings“ genannt. Zeichenketten können Buchstaben, Zahlen und Sonderzeichen beliebig aneinandergereiht enthalten. Der Name eines Kanals ist eine Zeichenkette, eine Zuweisung sieht folgendermaßen aus:

```
KANAL 4=RAUM5b
```

Auch Programme selbst sind Zeichenketten. Das oben verwendete Befehlsfolgen-Beispiel wird “zum Leben“ erweckt, indem es im Gerät B: als Hintergrundprogamm H 10 ständig den Relaiszustand kontrolliert:

```
<B> H 10= 'A:Pmom – 1..5, G5:Pmom – 8+17, +,125,>,IF,Rel 1=1, ELSE,Rel 1=0'
```

Zeichenketten können also auch Leerzeichen oder Sonderzeichen wie Kommas oder dergleichen enthalten. Daher muss eine Zeichenkette in Hochkommas oder in Anführungszeichen eingeschlossen werden,

falls Leerzeichen oder sonstige syntaktisch erhebliche Sonderzeichen in der Zeichenkette auftauchen können.

Beispiel: Die Ausgabefunktion ! druckt die als Parameter übergebene Zeichenkette:

```
! "der 'Druck'!"      : der 'Druck'!
```

Merke:
Die verwendeten Zeichenketten-Begrenzer dürfen innerhalb der Zeichenkette nicht vorkommen! Wenn Sie Hochkommas als Begrenzer verwenden, dürfen innerhalb der Zeichenkette Anführungszeichen, aber keine Hochkommas verwenden (und umgekehrt).

Für Zeichenketten gibt es keinen Stack, aber ein Puffer hält immer die zuletzt verwendete Zeichenkette fest. Damit ist es z.B. möglich, ein bestehendes Programm in ein anderes zu kopieren.

```
A:P! 1,b:P 11=$
```

Das Programm P1 im Gerät A: wird aufgelistet und in das Programm P11 im Gerät B: kopiert. Das $-Symbol dient als Anweisung, den Zeichenkettenpuffer zu benutzen.

1.3 Argumente, Extensions, Zuweisungen und Fehlermeldungen

Jeder Befehl kann, soweit das überhaupt sinnvoll ist, mit bis zu 3 Argumenten aufgerufen werden. Argumente werden in ECL auch als Parameter eines ECL-Befehls bezeichnet.
Der Zuweisungsoperator '=' im Anschluss ermöglicht weiterhin die Angabe von zusätzlichen Argumenten für eine Zuweisungsoperation. Darüber hinaus gibt es noch die Befehls-Extension (Befehlserweiterung). Mit dieser Extension lässt sich das Verhalten des Befehls steuern.

Der **Argument-Typ** ist abhängig vom Befehl, auch sind mehrere Typen pro Argument möglich. Folgende Typen sind definiert:

REAL	: 12 / 27.3 / –36.3E–2
AUFZÄHLUNG	: 2..7+V1..V7 / * / ** / #
.	: holt ein REAL-Element vom Parameter-Stack
Zeichenkette	: "Ein 'Beispiel' ..." / Kanal-5
$	: verwendet den Inhalt des Zeichenkettenpuffers

Für die Aufzählung gibt es folgende Notationen:

2..7	: Kanal 2 bis Kanal 7
2+7	: Kanal 2 und Kanal 7
V1..V3	: virtuelle Kanäle V1(== Kanal 25) bis V3(== Kanal 27)
2..7+V1	: Kanäle 2 bis 7 und V1
1..8+17+20..V3	: Kanäle 1bis 8 und Kanal 17 und Kanäle 20 bis V3
*	: alle EIN-geschalteten Kanäle (siehe EIN/AUS-Funktion)
**	: alle möglichen Kanäle
#	: alle Kanäle, die für die Messdatenliste formatiert wurden
##	: alle möglichen Kanäle, die für die Messdatenliste formatiert wurden

Die **Extension** beeinflusst das Verhalten der meisten Befehle. Ist es sinnvoll, können Extensions beliebig kombiniert werden. Detaillierte Angaben → ECL-Befehlsreferenz

–	: Ausgabe unterbinden (falls vorhanden)
– –	: Ausgabe umleiten in die Ablage
+	: Ausgabe direkt anschließen, ohne „neue Zeile“
.	: Ausgabe für Datenbanken, Trenner ';', Abschluss <CR><LF>
..	: wie . jedoch mit Trenner ';' zwischen Ausgabeblöcken
...	: wie .. jedoch trennen mehrerer Zeilen statt mit <CR><LF> mit ';'
#	: Ausgabe nur der Zahl, ohne Zusatzinformation Abschluss <CR><LF>
##	: Ausgabe nur der Zahl, ohne Zusatzinformation Abschluss ';'
%	: der 1. Parameter formatiert die Ausgabe (siehe PRINTFORMAT)
&	: Die Kennung z.B.A1:) wird zusätzlich am Anfang der Zeile ausgegeben
&&	: Die Kennung als Zahl (z.B. 2:) wird zusätzlich am Anfang der Zeile ausgegeben
* α	: Befehls-Modifikationen, z.B. Impuls- statt Energie-Ausgabe (siehe EGES)
_	: Harmonisierte (zurückschreibbare) Ausgabe von Energie-Befehlen
l	: zusätzliche Ausgabeformat-Option (siehe EGES)
/	: Ausgabe mit Zeitangabe „bis“
//	: Zeitangabe „von“ ... „bis“
^	: Ausgabe mit Zeitangabe „bis“ als Sekundenzahl ab 1.1.1990
^^	: Ausgabe mit Zeitangabe „von“ ... „bis“ als Sekundenzahl ab 1.1.1990
$	: zusammen mit . oder # erfolgt Ausgabe von Namen in „ ($$: auch Zeit)
!	: erzwingt Ausgabe (Beispiel: P! 3 listet Programm P3)

Beispiel anhand der Gesamtenergie Eges von Kanal 2 (Kanalname=Heizraum):

Eges 2	: EGes (02: Heizraum) = 21.31 kWh
Eges& 2	: A:EGes (02: Heizraum) = 21.31 kWh
Eges. 2	: EGes ;2; Heizraum;21.30527;kWh
Eges# 2	: 21.30527
Eges/ 2	: 15.08.92 23:10:11 : EGes (02) = 21.31 kWh
Eges#// 2	: 10.08.92;14:00:04;15.08.92;23:11:21;21.30527
Eges/## 1..4	: 15.08.92;23:11:21;0;21.30527;0;0
Eges^## 1..4	: 82768281;0;21.30527;0;0
EMON 1 2	: EMon 01 2 = 500.00 kWh, [rückschreibbar]
EMON* 1 2	: EMon* (01: Raum 501) = 50000.00 [Impulsanzahl]

Der Zuweisungsoperator weist einen Befehl an, statt der Ausgabe eine Zuweisung durchzuführen:

```
Eges 1=123.23 $
```

Hier wird dem Register der Gesamtenergie des Kanals 1 der Wert 123,23 zugewiesen.

Beispiele

Wir fassen die letzten Punkte anhand eines Beispiels zusammen. Dazu nehmen wir an, dass wir mit dem Gerät A: kommunizieren, B:Kanal 17 “Kanal17“ heißt und EGesT1 2 den Wert 222,22 kWh hat:

```
<A>b:Kanal – 17, c:Kanal V1..V4+V8=$, !“Name <<“ $ “>>, Wert = “, EgesT1#+ 2, d:Eges 5..8=.
```

- Name des 17. Kanals von B: in Zeichenkettenpuffer, keine Ausgabe
- Zuweisung per '$' (Zeichenkettenpuffer) der Kanalnamen V1 bis V4 und V8
- Ausgabe: Name <<Kanal17>>, Wert = 222.22
- EgesT1 von Kanal 2 auf den Parameterstack pushen
- Zuweisen des obersten Parameterstack-Elementes (=EgesT1 2) der EGes Kanäle 5 bis 8 in D:

Hinweis

Wie wir in diesem Beispiel sehen, spielt bei Befehlsnamen die Groß- und Kleinschreibung keine Rolle.

Fehlermeldungen

ECL liefert Klartext-Fehlermeldungen, die bei der Fehlersuche helfen. Sobald ein Fehler auftritt, wird die Programmabarbeitung abgebrochen und eine Fehlermeldung ausgegeben.
Fehler in Hintergrundprogrammen werden nur auf Wunsch ausgegeben. Der dazu bestimmte Befehl heißt “ERR“ und gibt eine Liste der Fehlermeldungen aller Hintergrundprogramme aus.
Stations- und Kanalfehler können mit den Befehlen ERRSTAT, ERRKAN abgefragt und auch beliebig maskiert werden.
Bitte vergessen Sie nicht, dass ECL eine interpretierende Programmiersprache ist. Programmbefehle werden erst dann ausgewertet (interpretiert), wenn sie auch an der Reihe sind. Ist z.B. in der Befehlsfolge zwischen IF ... ELSE ein Fehler, kann die zugehörige Fehlermeldung überhaupt erst dann erfolgen, wenn dieser Programmteil ausgeführt wird, die IF-Bedingung also erfüllt (==1) war.

1.4 Abbrechen von Programmen

Die Abarbeitung einer Befehlsfolge durch den ECL-Interpreter lässt sich mit folgender Tastenkombination abbrechen: <STRG> + X
Hintergrundprogramme lassen sich auf diese Weise nicht unterbrechen. Hier muss der Befehl HBREAK eingegeben werden. Dadurch wird das gerade laufende Hintergrundprogramm abgebrochen und die Abarbeitung für 16 Sekunden unterbrochen. Danach wird wieder mit H 0 begonnen.

1.5 Der System-Prompt und das Hilfe-System

Nach Drücken der <EINGABETASTE> ↵ meldet sich die Station mit ihrem Prompt:

<A>

Der Prompt informiert uns darüber, dass wir gerade mit der Summenstation mit der Kennung A: kommunizieren. Nach dem Prompt kann der Befehl oder die Befehlsfolge eingegeben werden. Maximal 128 Zeichen pro Zeile sind möglich. Die <EINGABETASTE> schließt die Eingabe ab und beginnt die Bearbeitung. Sobald der Befehl abgearbeitet ist, wird wieder der Prompt ausgegeben.

Einloggen

Durch die Vernetzung im ECS-LAN Verbund ist es möglich, sich in jedes beliebige Gerät im Netz “einzuloggen“. Der Interpreter verhält sich dann so, als ob das Terminal direkt an der RS-232 Schnittstelle des entsprechenden Gerät angeschlossen wäre. Nur anhand des Prompts können Sie feststellen, mit welcher Summenstation Sie gerade kommunizieren.

Zum Einloggen beispielsweise in das Gerät B1 geben Sie folgendes ein:

B1: : ↵

Ist das Gerät B1 verfügbar, so erscheint jetzt ein neuer Prompt: <B1>. Ab jetzt wird direkt mit dem Gerät B1 kommuniziert, d.h. alle Befehle ohne Kennungsangabe gelten für das Gerät B1.

Liste der möglichen Befehle, das Hilfesystem

Eine Liste der verfügbaren ECL - Interpreter Befehle erhalten Sie durch Eingabe des Befehls:

HILFE ↵ oder **?** ↵

Jetzt werden alle Befehle nach Funktionsgruppen gegliedert aufgelistet. Auch werden Stichworte zu allgemeinen Themen aufgeführt. Weitere ausführliche Hilfe zu den gelisteten Befehlen und Stichworten erhalten Sie durch Eingabe des Befehls:

HILFE <suchbegriff> ↵ oder **?** <suchbegriff> ↵

(Bitte beachten Sie das notwendige Leerzeichen (space) zwischen HILFE oder ? und <suchbegriff>)
Beispiel:
Sie möchten allgemeine Informationen zum Gebrauch von Parametern erhalten. Es müssen nicht alle Zeichen eines Suchbegriffs eingegeben werden (solange sie eindeutig sind):

? Para

Eine komplette Ausgabe aller Hilfetexte erhalten sie durch Eingabe von:

? Buch

Mit der PC-Parametrierungssoftware ECSoft 2 kann dann die komplette Ausgabe in eine Datei oder auf einen Drucker umgeleitet werden.

Hinweis

Das Hilfe-System stellt Informationen zu allen Befehlen des ECL-Interpreters zur Verfügung. Diese Informationen sind stets auf dem Stand der aktuellen Betriebssoftware

1.6 Übersicht der ECL-Befehlsgruppen

Stack-Operationen:
+ – * / & | ^ ~ && || ^^ ~~ SHR SHL < <= > >= == != DUP DROP SWAP PICK STKS PRINT !

Grundrechenarten und Boole'sche Vergleiche:
+ – * / & | ^ < <= > >= == !=

Konditionale Programmverzweigungen und Schleifen:
ALL ALS NEXTA FORI I NEXTI DO DOWHILE EXIT RETURN PAUSE
IF IFF ELSE ENDIF

Mathematische Funktionen und Zahlenmanipulationen:
SQRT SIN COS ASIN ACOS DEG RAD EXP LOG LOG10 ** ABS FRAC FIX INT INTR MAX MIN MOD

Gesamtenergien, Kosten und Momentanleistung:
Eges EgesT1 EgesT1T2 EgesT2 KostT1 KostT1T2 KostT2

Energie im Intervall, pro Tag, Monat und Jahr, Maxima:
EInt ETag EMon EJahr Emax EmTag EmMon EmJahr

Leistungen:
PInt PTag PMon PJahr Pmax PmTag PmMon PmJahr Pmom

Bildung von virtuellen Kanälen, Zeit- und Kalenderfunktionen:
VSUM VIRT TAG WTAG MON JAHR HH MM SS VON BIS DAUER
ZEIT DATUM DVSUM DVIRT DELTA

Intervall-Messdatenliste:
Einstellung des Intervalls, Formatierung, Index und Löschen der Liste, Kanallöschen.
INTERVALL INTERVALLQUELLE SYNC FORMAT INDEX LOESCHLISTE LOESCHKANAL

Tarif mit Tarifparametern:
TARIF TARIFQUELLE TEinh TFIX KOSFAK1 KOSTFAK2

Stations-Parameter:
Stations- und Gruppenname, Fehlererkennung:
GRUPPE MENUAPP MENUAPPN PEGEL RS232 STATION STATUS SYSDC SYSRESET SYSSN ERR
ERRKAN ERRKANLIST ERRNR ERRSTAT LBERR LERR

Kanal-Parameter:
Kanalname, Zählerkonst., Urat/Irat, KFix, Einheiten, EIN/AUS PFaktor ...
KANAL ZKONST URAT IRAT EEINH PEINH FLANKE PULS EINAUS KFIX PFAKTOR KMODE
LNAME ANAUSEL STARTSTOP KFAKTOR

Eingangsabfrage, Relais steuern, weitere Werkzeuge, Liste der Hilfsenergieunterbrechungen:
RELAIS RELAISMODE RELAISNAME DISPLAY PASSWORT PAUSE POWERFAIL POWERON TASTE

H- und P-Programme, Druckbefehle:
HBREAK H HLIST Q QLIST P PLIST LPSEARCH ERR LERR ERRLIST REM

Directory der ECS-LAN Teilnehmer, weitere Werkzeuge:
DIR DIRN DIRS INDIR AUFZ FINDER KENN MELD REM SETKENN VER SetLanR SetLanL

LON:
LONANA LONFAKTOR LONID LONKANAL LONMAXKANAL LONOFFSET LONP LONSTOP LONTYPE LONUSERS LONVER LONZW LONStatTiming LONPollDelay SetLON

Analogwerte:
ANA ANAFAKTOR ANAFIX ANAMAX ANAMAXR ANAMAXRN ANAMIN ANAMMCLR ANAMODE ANAN ANAOFFSET ANAR ANARESO ANARS ANASSEL ANAUSEL

Variablen:
A ALIST B BLIST

Zeitbefehle:
ZEIT DATUM TM TMD HTD LASTUPD VON BIS DAUER SOWI TAGBEG MONBEG

Sonstige:
AUFZ DELIMITER CHAIN DEVKEY DISPLAY FINDER LOGIN LOGOUT MELD PASSWORT TASTE TX1 TX2 VER WHOAMI

1.7 Werkzeugkasten

Anhand einiger nützlicher Beispielprogramme können Sie sich mit dem ECL-Interpreter vertraut machen. Bitte beachten Sie, dass Hintergrundprogramme zyklisch abgearbeitet werden und sich daher die Rechenzeit der einzelnen H-Programme auf die gesamte Zykluszeit auswirkt!

Hallo!
Alle Stationen zeigen im Display “Hallo !“, solange Pmom (1) > 30kW ist. Ein Beispiel für den praktischen Einsatz von Hintergrundprogrammen:

```
H 10=’pmom –  1,30,>,if,all,meld “Hallo !“  2’
```

Stellen der Uhrzeit, Zeit und Datum einstellen und des Datums aller Stationen im Netz
Mit dieser Befehlszeile werden alle Uhren im ECS-LAN Verbund gestellt:

```
all, zeit=12h34.56; datum=16.08.93
```

Synchronisieren, Zeit-Synchronisation aller Uhren im Netz
Jeden Tag um 0h00:15 werden alle Uhren von der Station A: (Beispiel) aus synchronisiert. Ein 'x' in der Zeit- oder Datumsangabe wird durch den aktuellen Wert der Station, auf der das Programm ausgeführt wird, ersetzt. Der ALL- Schleifenbefehl mit der Extension '–' bewirkt das Ausführen der ALL-Schleife für alle Stationen außer der “eigenen“ Station (hier A).

```
<A> H 10= ’if 0h0.15, ALL –, Zeit=x:x:x, Datum=x.x.x’
```

Tarif-Umschaltung
T1 (NT) gilt von 21h bis 6h, ansonsten gilt T2 (HT). Die Tarifquelle muss auf “Programm“ gesetzt sein!

```
<A> H 11= ’hh,6,>=,hh,21,<,&,if, Tarif=2, else, Tarif=1’
```

Tarif-Synchronisation im Netz
Systemweite Tarif-Synchronisation von Station A: (Beispiel) aus. Die Tarifquelle muss bei allen Stationen auf “Programm“ gesetzt sein!

```
<A> H 12= ’ tarif –,all –,dup,tarif=. ’
```

Alternativ: Der Update des gültigen Tarifes soll nicht dauernd, sondern nur bei einem Tarifwechsel stattfinden (bei IFF wird die Befehlsfolge zwischen IFF und ELSE bzw. ELSE und ENDIF stets nur einmal - nach Wechsel der Bedingung - durchgeführt):

```
<A> H 12= ’ tarif –,1,–,iff,all –,tarif=2,nexta,else,all –,tarif=1 ’
```

Intervall-Synchronisation im Netz

In Station A: (Beispiel) wird der externe Synchronimpuls über Kanal 24 eingespeist (A:Intervallquelle=24). Diese Station übernimmt dann die Intervall-Synchronisation der anderen Stationen im ECS-LAN.
Die Intervallquelle der Slave-Stationen muss auf “Programm“ gesetzt sein.

```
<A> H 13= ' Sync,iff,all –,sync= '
```

Ausdruck des Druckprogramm H29 jeden abend um 19h30

Ausgedruckt werden soll:

- die am Tag verbrauchte Energie aller eingeschalteten Zählkanäle (systemweit, mit Angabe des Datums als Überschrift „ Aktueller Tagesverbrauch am 23.03.99 19:30:00 “)

```
<A> H 29= 'if 19h30, P 29'
<A> P 29= '! – – "\r\n Akt. Tagesverbr. am %/// DD\r\n", TX2 $, all etag&– – *, TX2 $, na‘
```

Hinweis

Mit dem Com2-Mode „ECL+HP“ kann die Ausgabe der Hintergrundprogramme auf die Com2 zur Weiterverarbeitung umgeleitet werden!

Kopieren von P- und H-Programmen

```
<A> H 14= 'if 1.x.x 12h, h19'
<A> H 19= 'emon% “Verbrauch im %/dM 19%/dj“ 1 1,!!,all,emon& *,na,!!'
```

P 10 kopiert alle P-Programme, P 11 alle H- Programme zur Station B:

```
P 10= '! “Kopieren aller P-Programme nach B:“,0,31,fori,i,p – .,i,B:p=$'
P 11= '! “Kopieren aller H-Programme nach B:“,0,31,fori,i,h – .,i,B:h=$'
```

Betriebsdauererfassung

Wenn der Verbraucher eingeschaltet ist, liegt 24V am Eingang 4 an, ansonsten 0V.
Die Betriebsdauer lässt sich in Eges von Kanal 3 ablesen (in Sekunden), Eges von Kanal 4 gibt an, wie oft der Verbraucher eingeschaltet wurde. Zur Initialisierung der Zählung muss P 18 aufgerufen werden, die Auswertung erfolgt mit H 6:

```
P 18= '! “Vorbereiten Betriebsdauermess.“,zkonst 4=1,kanal 4=Schalten,p 19'
P 19= 'kanal 3=BtrDauer, eeinh 3=Sek,eeinh 4=mal,kfix 3..4=0,eges 3..4=0'
H 6='in – 4,if,zeit –,dup,a 6,–,eges – 3,+,eges 3=., else,zeit –, endif, a6=.'
```

ZählerTorsteuerung

Zähler 1 zählt nur während Eingang 8 auf logisch '1' ist. Mit dem Befehl STARTSTOP lässt sich der betreffende Zählkanal steuern.

```
H 7='in – 8,iff,startstop 1=1,else,startstop 1=0'
```

Aktivieren eines Relais in Abhängigkeit von PMOM

Relais 1 der Station B: wird aktiviert, sobald die Momentanleistung Pmom des virtuellen Kanals V2 der Station A: 55 kW überschreitet.
Dieses Hintergrundprogramm läuft in der Station B und überprüft Pmom in der Station A.

```
<B> H 10= 'A:Pmom – V2, 55, >, IF, Rel 1=1, ELSE, Rel 1=0'
```

Überprüfung der ECS-LAN Teilnehmeranzahl

Wenn die Anzahl der ECS-LAN Teilnehmer vom Soll (hier 4) abweicht, wird eine Warnmeldung auf allen LCD-Anzeigen der Stationen ausgegeben, sowie Relais 4 der Station X1 eingeschaltet.
Dieses Hintergrundprogramm läuft in der Station A. Die genaue Anzahl der Teilnehmer muss bekannt sein und entsprechend in dem Programm eingebettet werden.

```
<A> H 18= 'Bus –,4,!=,dup,X1:Rel  4=.,IF, ALL,meld "BUS Inkonsistenz"  2'
```

Sommer-/Winterzeit Umstellung

Für jeden Umschalttermin wird in einer ausgewählten Station (z.B. diejenige, die auch die systemweite Zeitsynchronisation vornimmt) ein H-Programm benötigt.
In den Monaten März und Oktober wird jeweils am letzten Sonntag um 2h / 3h die Umschaltung durchgeführt.

```
<A> H18= 'Rem Sommer/Winter, SOWI, IF, ZEIT –, +, ZEIT=.'
```

Für alle Stationen im Netz gilt:

```
<A> H 18= 'Rem Sommer/Winter, SOWI, IF, ZEIT –, +, ZEIT=. , ALL –, ZEIT= x:x:x'
```

In diesem Fall dürfen in den anderen Stationen im Netz keine H-Programme zur Umstellung laufen!

Überbrückung eines fehlenden Synchron-Impulses

Bleibt der Synchron-Impulslänger als 10 s über der eingestellten Intervalldauer aus, wird ein "künstliches" Intervall erzeugt. Dient eine Station als "Intervall-Synchronisations-Master", so genügt es, wenn dieses Programm nur dort installiert wird.

```
<A> H 14= 'rem SYNC-UEBERBRUECKUNG, sync/, intervall –, –,10,>,iff, sync+='
```

Impulserzeugung aus der Energie eines virtuellen Kanals

Auf Relais 1 wird pro 10 kWh (Teilungsfaktor 1/10) der Energie des virtuellen Kanals V1 ein Impuls ausgegeben. Benötigt wird ein Hintergrundprogramm (H 0) sowie ein P-Programm (P 0) und eine Variable (A 0). P 0 wird von H 0 aufgerufen, da der Speicherplatz nicht ausreicht, alle Befehle in H 0 zu implementieren. Sobald H 0 einprogrammiert wird, wird die Variable A 0 initialisiert. Ab diesem Moment beginnt die Impulsausgabe, die Pulsdauer sowie die Pulspause sind einstellbar (siehe Markierung in
P 0). 'PAUSE 0' bewirkt eine Pulsdauer/pause von ca. 80 ms, ansonsten kann die Pulsdauer/pause in Schritten von 200 ms eingestellt werden.

Beispiel für 400 ms Pulsdauer/pause: 'PAUSE 400'
Ändert sich V1 durch Rückstellen oder Zuweisungen der physikalischen Kanäle, so wird bei Erhöhung des Wertes versucht, so viele Impulse zu generieren, dass die Bilanz wieder stimmt (müssen mehr als 50 Impulse ausgegeben werden, wird auf den Bilanzausgleich verzichtet), bei Verringerung des Wertes beginnt die Impulsausgabe automatisch neu ab dem verringerten Wert.
In H 0 ist der Teilungsfaktor der Impulsausgabe markiert (Impulsanzahl=Energie/Teilungsfaktor).

```
H 0= '1,iff,eges –  v1,10,/,int,a=.,endif,eges –  v1,10,/,int,dup,a,–,dup,dup,p,a=.'
P 0= '0,>,swap,51,<,&,if,2,*,1,fori,i,2,mod,rel  1=.,PAUSE  0,nexti,else,drop'
```

Abfrage der Messdatenliste

Sämtliche Pint Eintragungen (bis Pint-1) in der Messdatenliste ab dem 17.08.92 18h45 des Kanals 1 der Station A: sollen ausgegeben werden. Die Zeit und Datum "von" und "bis" wird ebenfalls mit ausgegeben:

```
<A> index 17.8.92 18h45, pint// 1 . *
```

Ausgabe:

```
17.08.92 18:30:00 –– 17.08.92 18:45:00 : Pint–215 (01) = 1.23 kW
17.08.92 18:45:00 –– 17.08.92 19:00:00 : Pint–214 (01) = 1.80 kW
17.08.92 19:00:00 –– 17.08.92 19:15:00 : Pint–213 (01) = 1.12 kW
17.08.92 19:15:00 –– 17.08.92 19:30:00 : Pint–212 (01) = 2.10 kW
17.08.92 19:30:00 –– 17.08.92 19:45:00 : Pint–211 (01) = 2.05 kW
17.08.92 19:45:00 –– 17.08.92 20:00:00 : Pint–210 (01) = 2.07 kW
...
```

Sämtliche Eint im Datenbankformat (bis Eint – 0) mit Zeit und Datum "bis" werden ausgegeben. Diese Befehlsfolge wird P 2 zugewiesen:

```
<A>P 2='Eint/## # * **
<A>p 2
```

Ausgabe:

```
16.08.92;17:45:00;1;0.5;0.75;0.99
16.08.92;18:00:00;1.01;0.1;0.76;0.80
16.08.92;18:15:00;0.99;0.48;0.75;1.02
16.08.92;18:30:00;0.89;0.5;0.76;0.99
16.08.92;18:45:00;1;0.52;0.77;1
16.08.92;19:00:00;1.01;0.51;0.75;0.98
...
```

Erstellen einer Datenbank im ASCII-Format;
Spalten = Kanäle

Es soll eine Datenbank im ASCII-Format (Trennzeichen ;) erstellt werden, die folgende Auswahl von Messdaten aller am ECS-LAN angeschlossener Summenstationen enthält:
Gesamtenergien EGES, EGEST1, EGEST2, sowie die Momentanleistung PMOM.
Spaltenbeschreibung: Station, Funktion, Wert-Kanal-1,..., Wert-Kanal-V8
Die erste Zeile enthält die Spaltenüberschriften.

Station	Funktion	1	2	3	...	32
A	Kanal	Heizraum	Motor015	Kanal-3	...	GesKost8
A	Eges	12.7	6.956	0	...	147.9734
A	EGEST1	12.7	6.956	0	...	147.9734
A	EGEST2	0	0	0	...	0
A	Pmom	0	0	0	...	0.37
...						
C1	Kanal	Motor001	Motor002	Motor003	...	GesMot01
C1	Eges	0	17.22	158	...	1379.5554
C1	EGEST1	55.3	0.12	0	...	147.9734
C1	EGEST2	0	0.93	0	...	192.11
C1	Pmom	0.54	1.17	0	...	5.557
...						

Die ASCII-Datenbank sieht folgendermaßen aus:
Station;Funktion;1;2;3; ... ;32
A;Kanal; Heizraum; Motor015; Kanal-3; ... ;GesKost8
A;Eges;12.7;6.956;0; ... ;147.9734
A;EGEST1;12.7;6.956;0; ... ;147.9734
A;EGEST2;0;0;0; ... ;0
A;Pmom;0.37;0;0; ... ;0.37
...
C1;Kanal;Motor001;Motor002;Motor003; ... ;GesMot01
C1;Eges;0;17.22;158; ... ;1379.5554
C1;EGEST1;55.3;0.12;0; ... 147.9734
C1;EGEST2;0;0.93;0; ... ;192.11
C1;Pmom;0.54;1.17;0; ... ;5.557
...
Aufruf von Programm P 10 der am PC per RS-232 angeschlossenen Station erstellt die gewünschte Ausgabe, P 11 bis P 13 sind Hilfsprogramme von P 10.
Dieses Programm P 10 (zusammen mit den Hilfsprogrammen) kann nur auf der am PC angeschlossenen Station aufgerufen werden (eingeloggt in deren Kennung): P 10

Hinweis

Eine Kennung der Form AA: spricht stets die Station an, die mit dem PC verbunden ist.

```
AA:P 10=’! “Station;Funktion;“,aufz##+  **,aqll,AA:p  12,AA:p  13’
AA:P 11=’kenn.,!+ “;Kanal;“, kanal##+  **’
AA:P 12=’kenn.,!+ “;Eges;“, eges##+  **, kenn.,!+“;EgesT1;“,egesT1##+  **’
AA:P 13=’kenn.,!+ “;EgesT2;“,egesT2##+  **,kenn.,!+ “;Pmom; “,PMOM##+  **’
```

Mit ECSoft 2 ist es möglich, diese Ausgabe direkt in eine Datei umzuleiten, metasprachliche Kommandos bei der Skript-Ausführung erlauben eine automatisierte Betriebsweise von ECSoft.

Erstellen einer Datenbank im ASCII-Format;
Spalten = Funktionen

Es soll eine Datenbank im ASCII-Format (Trennzeichen ;) erstellt werden, die folgende Auswahl von Messdaten aller am ECS-LAN angeschlossener Summenstationen enthält:
Gesamtenergien EGES, EGEST1, EGEST2, sowie die Momentanleistung PMOM.
Spaltenbeschreibung: Station, Kanalnr, Kanal, EGES, EGEST1, EGEST2, PMOM

Die erste Zeile enthält die Spaltenüberschriften.

Station	Kanalnr	Kanal	Eges	EgesT1	EgesT2	Pmom
A	1	Heizraum	12.7	12.7	0	0.37
A	2	Motor015	6.956	6.956	0	0
A	3	Kanal-3	0	0	0	0
A	...					
A	32	GesKost8	147.9734	147.9734	0	0.37
...						
C1	1	Motor001	0	55.3	0	0.54
C1	2	Motor002	17.22	0.12	0.93	1.17
C1	3	Motor003	158	0	0	0
C1	...					
C1	32	GesMot01	1379.5554	147.9734	192.11	5.557
...						

Die ASCII-Datenbank sieht folgendermaßen aus:
Station; Kanalnr;Kanal;Eges;EgesT1;EgesT2;Pmom
A;1;Heizraum;12.7;12.7;0;0.37
A;2;Motor015;6.956;6.956;0;0
A;3;Kanal-3;0;0;0;0
A;...
A;32;GesKost8;147.9734;147.9734;0;0.37
...
C1;1;Motor001;0;55.3;0;0.54
C1;2;Motor002;17.22;0.12;0.93;1.17
C1;3;Motor003;158;0;0;0
C1;...
C1;32;GesMot01;1379.5554;147.9734;192.11;5.557
...
Aufruf von Programm P 15 der am PC per RS-232 angeschlossenen Station erstellt die gewünschte Ausgabe, P 16 ... P 18 sind Hilfsprogramme von P 15.
Dieses Programm P 15 (zusammen mit den Hilfsprogrammen) kann nur auf der am PC angeschlossenen Station aufgerufen werden (eingeloggt in deren Kennung): P 15
Hinweis: Eine Kennung der Form AA: spricht stets die Station an, die mit dem PC verbunden ist.

```
AA:P 15='! "Station;Kanalnr;Kanal;Rges;EgesT1;EgesT2;Pmom",AA:p 16'
AA:P 16='all,fori  **,i,AA:p 17,AA:p 18,nexti,nexta'
AA:P 17='dup,kanal.&  .,!+ ";",dup,eges+#  .,!+ ";",drop,dup,egest1+#  .'
AA:P 18='!+ ";",drop,dup,egest2+#  .,!+ ";",drop,pmom+#  .,drop'
```

Mit ECSoft 2 ist es möglich, diese Ausgabe direkt in eine Datei umzuleiten, metasprachliche Kommandos bei der Skript-Ausführung erlauben eine automatisierte Betriebsweise von ECSoft.

2 ECL-Befehlsreferenz

2.1 Allgemeine Informationen

Online-Befehlsverzeichnis

Allgemeine Informationen zur Verwendung der ECL-Interpreter-Befehle:

Allgemeine Informationen

Abbrechen einer laufenden Ausgabe: **^X** (STRG und X gleichzeitig)
Abbrechen der Ausgabe von Hintergrundprogrammen:

- Eingaben sind trotz der Ausgabe möglich
- Abbruchbefehl: siehe HBREAK (16 s Pausieren der H-Programme)
- Das Steuerzeichen ^B schaltet das Vermischen der H-Programmausgabe für 10 s aus (H-Programm-Ausgaben in dieser Zeit werden ignoriert).

Abrufen spezieller Hilfstexte

- Soweit eindeutig, müssen nicht alle Buchstaben des Suchbegriffes eingegeben zu werden.
- Für einige Befehle gibt es Kurzformen (Angabe in Klammern); die Befehlssuche funktioniert auch für die Befehls-Kurzformen.
- Komplette Ausgabe aller Hilfstexte: **? BUCH**

Kompatibilität

- Die implementierte ECL-Version ist aufwärtskompatibel zum ECL der U1600/10/15-Stationen.
  Neue Befehle oder Befehlsformen, die bei den U1600/10/15-Stationen nicht verfügbar sind, sind mit (U1600: n.v.) gekennzeichnet:
  (U1600: n.v.) : nicht verfügbar
  (U1600: b.v.) : beschränkt verfügbar, d.h. nur bei direktem COM-Zugang über U1600/10/15 NICHT verfügbar.

ECL-SYNTAX, metasprachliche Begriffsdefinitionen:

ECL-SYNTAX

<abcd> :=	: Begriffsdefinition
[ ]	: optionale Angaben
<ab> | <cd>	: Alternative
{ . . }	: Liste
[. .]^ oder **{. .}^**	: Wiederholung
__	: Leerzeichen

```
<befehlsfolge>:= <kennung><befehl> [ , | ; <befehlsfolge>]
<befehl>      := <text><ext> [__<par>[__<par>[__ . . ]]]
                   [=[<par>[__<par>[__..]]]]
<kennung>     := {A AA A1 . . A9 AN B B1 . . B9 C . . Z4 ZZ} : | :: [__]
<ext>         := [{ ! + – # . * / ^ $ _ ? | @ }]^
<par>         := <real> | <zeichenkette> | <aufzählung> | . | $
<real>        := [–]<integer>[ E <integer>]
<integer>     := [–]{0 . . 9}^
<zeichenkette>:= [“ | ’] <text> [__<text>]^ [’ | “]
<text>        := {a . . z A . . Z 0 . . 9 _ – +}^
<aufzählung>  := { * ** # ## <kanal> } [ . . | + | – | ^ [ <kanal>]]^
<kanal>       := <integer> | {V1 . . V8}
```

Extension

Die Extension (Ext) <ext> beeinflusst das Verhalten der Befehle
(Beispiele siehe Extension-Beispiele)
Normalerweise gelten folgende Regeln:

–	:	Ausgabe unterbinden (falls vorhanden)
– –	:	Ausgabe umleiten in Ablage (Befehl muss Ext % kennen)
+	:	Ausgabe direkt anschließen, ohne „neue Zeile“ zu Beginn
!	:	erzwingt Ausgabe (Beispiel: P! 3 listet Programm P3)
%	:	der 1. Parameter formatiert die Ausgabe (s. PRINTFORMAT)
&	:	die Kennung wird zusätzlich am Zeilenanfang ausgegeben (s. KENN)
*** @**	:	Befehls-Modifikation, z.B.: Impuls- statt Energie-Ausgabe (s. EGES)
_	:	Harmonisierte (zurückschreibbare) Ausgabe von Energie-Befehlen
l	:	zusätzliche Ausgabeformat-Option (s. EGES)
.	:	Ausgabe für Datenbanken mit Trenner ';' und Abschluss >CR><LF>
. .	:	wie . jedoch mit ';' zwischen Ausgabeblöcken
. . .	:	wie . . jedoch Trennen mehrerer Zeilen statt mit <CR><LF> mit ';'
#	:	Ausgabe nur des Hauptwertes. ## und ### ist analog zu . . und . . .
/	:	Ausgabe mit Zeitangabe (weitere Infos: siehe VON oder BIS)
^	:	Ausgabe mit Zeitangabe als Sekundenzahl ab 1.1.1990

Beispiele zur Verwendung der Extension

EGES	1+V2	:	EGes (01 : Raum501) = 874,01 kWh EGes (V2 : Raum777) = 12,74 kWh
EGES.	1+V2	:	EGes;1;Raum501;874,0124;kWh EGes;26;Raum777;12,739;kWh
EGES..	1+V2	:	EGes;1;Raum501;874,0124;kWh;EGes; 26;Raum777;12,739;kWh
EGES#	1+V2	:	874,0124 12,739
EGES##	1+25+V2	:	874,0124;100;12,739
EGES_	1 2	:	EMon 01 2 = 500,00 kWh [rückschreibbar]
EGES*	1 2	:	EMon* (01:Raum501) = 50000,00 [Impulsanzahl]
EGES//	1 2	:	01.09.92 00:00:00– –01.10.92 00:00:00 EMon–2 (01: ...)
EGES/##	1..4 2	:	01.10.92;00:00:00;500;1,1234;7555; 0,0001
EGES^##	1..4 2	:	86745600;500;1,1234;7555;0,0001
INTERVALL&		:	A:INTERVALL = 15 Minuten

Befehlsparameter

Befehle nehmen folgende PARAMETER an (Beispiele):
Aufzählungen: [Suchen nach Namen siehe FINDER]

<aufzählung>		Bereich 1 ... 64 oder (wenn möglich) 0 ... 63
2..9+15+17	:	2 bis 9 und 15 und 17
5+V1..V4	:	5 und V1(=25) bis V4(=28)
2..16–5–8+24	:	2 bis 16 und 24, ohne 5 und 8
*	:	alle EINgeschalteten Kan. (siehe EINAUS). Falls Bezug fehlt: '**'
#	:	alle formatierten Kanäle (siehe FORMAT)
**	:	1 bis 32
*–3..6	:	alle '*', jedoch ohne 3 bis 6
*+7^+10..12	:	Komplement von (alle '*' und 7) plus 10 bis 12
.	:	ein Element vom Stack, Nachkommastellen entfernt
..	:	eine Aufzählungszahl vom Stack, siehe AUFZ (nur U1600)
i	:	aktuelle i, j oder k-Zählervariable (U1600: n.v.)

Allgemeine Zahlenangaben

–12.34E3	:	<real>
8	:	<integer>
0x12ab	:	<hexadezimal> (32 Bit)
0b100101	:	<binär> (32 Bit)
.	:	from Stack
i	:	aktuelle i_Zählervariable, j+k analog (U1600: n.v.)
t	:	Echtzeit-Sekundenzahl mit 1/1000s Anteil (U1600: n.v.)
t_	:	Betriebsstundenzähler-Sekundenzahl " " (U1600: n.v.) bitte beachten Sie den Stations-Bezug von Zeitabfragen, siehe ZEIT.

Allgemeine Zeichenketten (siehe STRINGS)

Hallo	:	<string>
“Hallo 'Welt'“	:	<string> mit "" begrenzt, da Leerzeichen enthalten sind
$	:	aus String-Ablage (Clipboard)

FINDER

FINDER : systemweites Suchen von Kanal-, Relais- oder Stationsnamen

- Statt <aufzählung> lässt sich bei kanalbezogenen Funktionen der Name des gesuchten Kanals angeben (auch für Relaisnamen bei REL... und Stationsnamen bei KENN). Der Suchbegriff wird im gesamten ECS-LAN-Verbund gesucht, angefangen bei der Prompt-Station.
- Der Suchbegriff muss mit einem Buchstaben beginnen, ansonsten muss ein '$' vorangestellt werden.
- „name*“ sucht das erste Auftreten von „name...“, Groß-/Kleinschreibung ist unwichtig. „name**“ gibt alle passenden Kanäle von der ersten möglichen (!) Station aus.
- <suchbegriff>& sucht nur auf aktueller Station.
- <suchbegriff>@@unterdrückt '... nicht gefunden' Meldungen.

FINDER (FI)	:	Funktion zum Ermitteln von Stationsdetails zu einem Suchbegriff
Aufruf	:	FINDER <suchbegriff>>
Stack	:	– >>> <kanalnr> <kennungszahl> <1: gefunden/0: nicht gefunden>
Ext	:	* (Suche nach Relaisnamen) @@(Suche nach Stationsnamen)

Parameter-Stack

Parameter-Stack

Der Parameter-Stack ist ein Kellerspeicher, der zur Übergabe von Zahlen-Parametern zwischen Befehlen fungiert. Der Stack lässt sich mit bestimmten Befehlen manipulieren (DUP DROP PICK SWAP...). Diese Befehle entnehmen dem Stack Werte (pop) und fügen auch wieder welche hinzu (push).

Stacktiefe	:	63
Datentyp	:	<real> : 64 Bit Fließkommazahl (15 signifikante Dezimalstellen)

Das Stack-Verhalten ist elementarer Teil der Befehlsbeschreibung. Allgemein gilt: Beim Schreiben (Verwenden des '=' Zeichens) werden auf dem Stack NIE WERTE ABGELEGT (Push).

Hinweis
Der Stack und die Zwischenablage gelten nur während der Abarbeitung einer Zelle (Verschachtelungen mit P-Programm möglich). Sobald der Promt zurückkehrt, wird der Stack gelöscht. Dadurch wird eine konsistente Programmierumgebung ermöglicht.

Zwischenablage

Für Zeichenketten (Strings) existiert ein Zwischenspeicher ("Ablage" oder "Clipboard"), der beim Lesen stets das letzte Zeichenketten-Ergebnis beinhaltet. So legt z.B. der Befehl STATION beim Lesen den Stationsnamen in der Zwischenablage ab.
Bezugnahme auf die Zwischenablage in Parametern: '$'

Variablen

Zur dauerhaften Speicherung von <real> Zahlen stehen
64 Register A0 ... A63 sowie 64 Register B0 ... B63 zur Verfügung (siehe A, B).

STRINGS (= Zeichenketten)

Einfügen von Sonder- / Kontrollzeichen in Strings mit '\'Prefix:]

\#	: “	\“	:	“
\'	: ’	\’	:	’
****	: \			
\b	: 0x08 Backspace			
\l	: 0x12 ^L			
\n	: 0x0A (LF)			
\r	: 0x0D (CR)			
\t	: 0x09 (TAB)			
\nnn	: nnn= '3-stellige Dezimalzahl' entsprechend dem Zeichencode			
\000	: nichts einfügen	(U1600: n.v.)		
\-	: restliche Zeichen nicht mehr umwandeln (% ... , \ ...) (U1600: n.v.)			

– nicht genannte Zeichen: \z → z

– Die Umformung der Metacodes in die entsprechenden Zeichen findet bei der ZUWEISUNG eines Strings NICHT statt (Ausnahme: \“ und \'):
p='! “Buchstabe \#\065\#“' , p! : P 0='! “Buchstabe \#\065\#“'
p : Buchstabe “A“

KENNUNG

Jede Summenstation besitzt eine eindeutige KENNUNG
Es gibt 255 Kennungen (A, A1... A9, B, B1... B9... Z4) sowie
3 Sonderkennungen.

AA	: Kennung der mit der RS-232 Schnittstelle verbundenen Station
ZZ	: Kennung der durch den Interpreter-Prompt verbundenen Station
AN	: holt Kennung als Zahl vom Stack (A==1... Z4==255)

A1:<befehl> oder A1:	: vollständige Kennung-Kontext-Umschaltung
A1: <befehl> oder A1:, ...	: schaltet Kontext nur für aktuelle Zeile um
A1:<befehl>	: Kennung gilt nur für diesen Befehl

Beispiel für die Spezialkennung ZZ

Eingeloggt in eine entfernte Station wird ein P-Programm gestartet.
P enthält eine ALL-Schleife, in der ein weiteres P-Programm, dass sich auf der selben logischen Station befindet, aufgerufen wird. Da innerhalb der ALL-Schleife der Bezug auf eine feste Station nicht vorhanden ist, bietet ZZ: die Möglichkeit, diesen festen Bezug wieder herzustellen, ohne explizit die Station nennen zu müssen!
<C> .. ALL, .. , P 15, ...ruft P 15 von ALLEN Stationen auf
<C> .. ALL, .. , ZZ:P 15, ...ruft P 15 nur von Station C: auf

RS-232 Schnittstellen-Protokoll

RS-232 Schnittstellen-Protokoll

^M	(RETURN)	13d	:	sendet die Eingabezeile ab, nach der Antwort kommt der System-Prompt (z.B.: <A1>) [kein Prompt: ^W^M]
^J	(CTRL-RETURN)	10d	:	wie RETURN, nur statt dem Prompt kommt ^Z (SUB, 26d); (geeignet für Host-Verbindung und zusammen mit ^V)
^X	(CAN)	24d	:	Abbruch der Ausgabe, Löschen aller Puffer und Flags
^Y	(EM)	25d	:	Zeile löschen, keinerlei Ausgabe
ESC		27d	:	Zeile löschen, Cursor geht in neue Bildschirmzeile

Protokoll-Flags

^B	(STX)	02d	:	unterdrückt Eingabe-Echos für laufende Zeile
^A	(SOH)	01d	:	Vormerken für kodierte Fehlerausgabe: ^A nnn
^A^A			:	wie ^A, jedoch statt nnn-Fehlernummer: ^A <ErrText>
^V	(SYN)	22d	:	Löschen der internen Prüfsumme und Vormerken für Ausgabe mit „Prüfsumme nach SUB“
^V^V			:	Nach ^M oder ^J muss Prüfsumme folgen, Antwort: ACK/NAK

– Protokoll-Flags ^A, ^V und ^B gelten stets nur für den nächsten Befehl.
– ^V und ^B unterbinden Mischung der H-Programmausgaben für 10 s + Befehlsdauer.
– Prüfsumme: Addition auf 16Bit INT, feste 4-stellige Darstellung in HEX.

Geräte-Status

Geräte-Status

Mit dem ECL-Befehl **STATUS (Kurzform STAT)** wird eine Übersicht über wesentliche Stations-Parameter ausgegeben. Hier eine typische Ausgabe:

```
Station        : A4:RaumG48  [Labor]
ECS-U1601      : Software V2.47  (22.03.02)
Intervall      : 1 m  (Zeit)
Format (0)     : 64 Kanäle, 3966 Einträge  (2,8 Tage), 193 benutzt
Tarif          : T1  (Programm)
Relais         : R1:p  R2:p  R3:p  R4:p  R5:p  R6:p
24 V-Ausgang   : OK
Lithium-Bat.   : OK
Status Relais  : 1  (OK), gekoppelt
Max. L-Pegel   : 1  (0:Lo....3:Hi)
COM1           : 9600 Baud, Parität: Off, Protokoll: Xon/Xoff, ECL
COM2           : 115200 Baud, Parität: Off, Protokoll: Xon/Xoff
LON            : Teilnehmer: 1
BUS-L          : 375 kBaud  (4D), Teilnehmer-L: 10(1), gesamt: 12
BUS-R          : 375 kBaud  (4D), Teilnehmer-R: 2(1)
```

ABS FRAC INT INTR MIN MAX MOD FIX FIXE

Allgemeine Zahlenmanipulationen

				Stack:	
ABS	: <wert>		>>>		Betragsfunktion (<wert>)
FRAC	: <wert>		>>>		gebrochener Teil (<wert>)
INT	: <wert>		>>>		Ganzzahlteil (<wert>)
MIN	: a	b	>>>		die kleinere Zahl von a und b
MAX	: a	b	>>>		die größere Zahl von a und b
MOD	: a	b	>>>		(a modulo b)
FIX	:	n	>>>	–	Festkomma-Darstellung, n = Anzahl der Nachkommastellen
FIXE	:	n	>>>	–	Exponential-Darstellung, n = Anzahl der Nachkommastellen. (U1600: n.v.)
Fixpunkt-Einstellung	: FIX	FIXE			

- FIX oder FIXE gelten für den Rest der Zeile.
- FIX n mit n >= 7 schaltet auf Fließkomma-Darstellung mit maximal n Nachkommastellen um. Default: n=7. (U1600: Fließkomma ab n>=9, Default: n=9)
- FIX akzeptiert auch negative Argumente: die Zahl wird auf den Betrag der angegebenen Stellenanzahl gerundet und statt in Festkomma- in Fließkomma-Darstellung angezeigt (U1600: n.v.).
- FIX / FIXE erlauben die Angabe eines Parameters (Stack bleibt unverändert): FIX <n> == <n>, FIX
- Die Exponential-Darstellung mit FIXE hat stets feste Breite (abhängig von der Anzahl der Nachkommastellen), mit der Extension '+' wird auch das Vorzeichen immer mit + oder – angegeben.

Beispiel für FIX

12.345, FIX 0, ! → 12 12.345, FIX 5, ! → 12.34500
12.345, 2, FIX, ! → 12.35 12.345, 9, FIX, ! → 12.345

Beispiel für FIXE

12.345, FIXE 2, ! → 1.23E+01 12.345, FIXE+ 3, ! → +1.235E+01
–12.345, FIXE 3, ! →-1.235E+01

ALL NEXTA

ALL NEXTA (NA) : Programm-Schleife „alle Kennungen“

Diese Schleifenbefehle ermöglichen das Ansprechen aller Stationen mit einem Befehl. Es gibt keine Laufvariable, die aktuelle Kennung für die Zeile wird hochgezählt (nur Kennungen von Stationen im ECS-LAN).

Aufruf	: ALL [<vonKennungsnr> [<bisKennungsnr>]] NEXTA	
Stack	: – >>> –	– >>> –
Ext	: –	: Auslassen der Station mit aktueller Kennung
Aufruf	: ALL [<auswahl>]	U1600: n.v.)

– Mit <auswahl> : { A | B | U | R | | P | * } wird die Geräteliste auf eine bestimmte Geräteklasse eingeschränkt (siehe DIR).
– Beim Aufruf von lokalen P-Programmen innerhalb einer ALL-Schleife bitte das Beispiel für Spezialkennung ZZ: beachten (siehe KENNUNG).
– Zur Demonstration des inneren Ablaufes von ALL... siehe Beispiel bei INDIR.

Beispiele

ALL, eges 1, nexta Ausgabe: Eges Kanal-1 aller Geräte
ALL-, eges 1, nexta Ausgabe: Eges Kanal-1 aller Geräte, außer Promt-Gerät

Arithmetische Operatoren

+ – * / ** & && | || ^ ^^ XOR < <= > >= == != !

+	:	a	b	>>>	(a + b)
–	:	a	b	>>>	(a – b)
*****	:	a	b	>>>	(a * b)
/	:	a	b	>>>	(a / b)
******	:	a	b	>>>	(a hoch b)
&	:	a	b	>>>	(a logisch-UND b)
&&	:	a	b	>>>	bitweises UND (32 Bit)
|	:	a	b	>>>	(a logisch-ODER b)
||	:	a	b	>>>	bitweises ODER (32 Bit)
^	:		a	>>>	(NOT a)
^^	:	a	b	>>>	bitweises Compliment (32 Bit)
XOR	:	a	b	>>>	(a XOR b) bitweises Exclusiv-ODER (32 Bit)
SHL	::	a	n	>>>	(Shift-32bit-Left n*) SHL <n> :: a >>> (Sh... n*)
SHR	::	a	n	>>>	(Shift-32bit-Right n*) SHR <n> :: a >>> (Sh... n*)
<	:	a	b	>>>	(Vergleich a < b) ja = 1, nein = 0
<=	:	a	b	>>>	(Vergleich a ≤ b) ja = 1, nein = 0
>	:	a	b	>>>	(Vergleich a > b) ja = 1, nein = 0
>=	:	a	b	>>>	(Vergleich a ≥ b) ja = 1, nein = 0
==	:	a	b	>>>	(a gleich b)
!=	:	a	b	>>>	(a ungleich b)
!	:	Stack-Ausgabefunktion nimmt eine Zahl vom Stack und gibt diese aus. Weitere Möglichkeiten des !-Befehls: siehe PRINT			

Bit-Shifts und Umwandlungen Binär / BCD

SHL	:: a n >>>	(Shift-32bit-Left n*)
SHL <n>	:: a >>>	(Sh... n*)
SHR	:: a n >>>	(Shift-32-bit-Right n*)
SHR <n>	:: a >>>	(Sh... n*)
BIN2BCD	:: bin >>> bcd	Binär → BCD; Beispiel: 1234 >>> 0x1234
BCD2BIN	:: bcd >>> bin	BCD → Binär; Beispiel: 0x1234 >>> 1234

Hinweis
Mit "::" gekennzeichnete Befehle sind bei U1600/10/15 nur beschränkt verfügbar (U1600: b.v.).

Bei logischen Vergleichen gilt:
FALSCH: gleich 0.0, WAHR: ungleich 0.0

A/B-Register

A0 . . A63 : Register A0 ... A63 für <real> Zahlen (U1600: A0 ... A9)
B0 . . B63 : Register B0 ... B63 für <real> Zahlen (U1600: n. v.)

Aufruf	A <aufzählung> [=<neuerWert>]	
Stack	: – >>> Inhalt (Ai)	Abfrage (Bei Aufzählung wird Summe gebildet)
	wert >>> –	Zuweisung
Ext	: + – . # ! ++ – – %	

- A ohne <aufzählung> == A0
- A1 . . A19 entspricht A 1 . . A 19, A5! == A! 5
- Inkremetieren (+1) : A++ <aufzählung>
- Dekrementieren (–1) : A–– <aufzählung>
- Addieren von <wert> zum Register :
  A++ <aufzählung> = <wert>
- Subtrahieren von <wert> vom Register :
  A–– <aufzählung> = <wert>
- <neuerWert> == {tlz} weist die aktuelle Sekunden-Zeitzahl (mit 1/100 s) zu.

ALIST oder ALIST <aufzählung> : Listet A-Register (entspricht A! *)

Analog-Eingänge

– Aus einer analogen Leistung wird mit AnaMODE = 2 die Energie ermittelt.

– Zum Zählen von Impulsen wird AnaMODE = 3 verwendet. Hier kann mit INPUT der Zustand des Eingangs abgefragt werden {0|1}. Der Befehl PEGEL bestimmt die Schaltschwelle: 0 = 10%, 1 = 25% (Vorgabe), 2 = 50%, 3 = 70% von Fullrage. PULSDAUER wird zur Bestimmung des Eingangszustandes verwendet.

– STARTSTOP beeinflusst die Energie-Berechnung bei AnaMODE = 2 oder 3.

– Mit AnaMODSEL wird die Eingangs-Charakteristik ausgewählt (das Gerät muss zusätzlich hardwareseitig umgestellt werden)

0 : –10 ... 0 ... +10 V
1 : –20 ... 0 ... +20 mA
2 : –5 ... 0 ... +5 mA
3 : S0
4 : 3 ... 20 mA (20mA Bereich)

– U1601 stellt 12 Eingänge auf Kanal 1 ... 12 bereit.

– U1615 stellt max. 7 Eingänge auf Kanal 1 ... 7 bereit.

Analog-Ausgänge:

– Der Befehl AnaRESO ist ohne Bedeutung, (bei U1615 auch AnaMAX und AnaMIN). AnaMODE 2, 3 ist nicht zulässig.

– U1601 stellt zwei Ausgänge auf Kanal 13+14 bereit.

– U1615 stellt max. 7 Ausgänge (nur unipolar) auf Kanal 1 ... 7 bereit.

– Mit AnaMODSEL wird die Ausgangs-Charakteristik ausgewählt. Für U1601 gilt (das Gerät muss ggf. hardwareseitig umgestellt werden):

0 : –10 ... 0 ... +10 V
1 : –20 ... 0 ... +20 mA
4 : 4 ... 20 mA (20mA Bereich)

– Für U1615 gilt:

0 : 0 ... +20 mA
1 : 4 ... 20 mA (20mA Bereich)

– Mit dem Befehl AnaINT (mit n > 0) wird eine sekundengenaue Schleppzeiger-Mittelwertbildung beim Analog-Ausgang aktiviert. Der Ausgang und ANA (lesend verwendet) liefern stets den Mittelwert der zugewiesenen ANA-Werte der letzten n Sekunden. AnaINT ist bei U1601-Versionen ab dem 08.11.99 verfügbar.

Aufruf : AnaINT <kanal> [= <wert>] Mittelwertbildungs intervall in Sekunden (*)
0 : Standard-Momentanwert
1...60 : Schleppzeiger-Intervall

Relais-Ausgänge (nur U1615)

– Nur die Befehle AnaModID und AnaRelMap sind sinnvoll.
– U1615 stellt max. 7 Relais-Ausgänge (Arbeitskontakt) auf Kanal 1 ... 7 bereit.

Allgemein

– Ext '?' (ANA? 1) unterdrückt Fehlermeldung „Funktion nicht verfügbar" auf Stationen, die keine Analogverarbeitung bieten.
– Mit (*) gekennzeichnete Befehle sind bei U1615 nicht verfügbar

Aufruf : ANA <kanal>	Analoger Eingang
Aufruf : ANA <kanal> =<wert>	Analoger Ausgang
Aufruf : AnaR <kanal> [=<wert>]	Analog-E/A, Roh-Wert entsprechend AnaModSel (*)
Aufruf : AnaN <kanal> [=<wert>]	Analog-E/A, –1 ... 0 ... +1 normierter Wert (*)
Aufruf : AnaRS <kanal> [=<wert>]	Analog-E/A, Roh-Wert entsprechend AnaMod-Sel, Vorzeichenbereich (siehe AnaSSEL) jedoch nicht eingeschränkt.(*)
Aufruf : AnaMAX <kanal> [=<wert>]	Maximum (mit Zeit-stempel: ANAMAX/ <kanal>)
Aufruf : AnaMIN <kanal> [=<wert>]	Minimum (Zeitstempel neu: ANAMIN/ <k> = <w>)
Aufruf : AnaMAXR, AnaMINR	Maximum/Minimum, Wertebereich wie AnaR (*)
Aufruf : AnaMAXN, AnaMINN	Maximum/Minimum, Wertebereich wie AnaN (*)
Aufruf : AnaMMCLR <kanal> = 0	Löschen des Maximums und Minimums (*)
Aufruf : AnaFACTOR <kanal> [=<wert>]	

Aufruf : AnaOFFSET <kanal> [=<wert>]
ANA = AnaN * AnaFACTOR + AnaOFFSET
AnaOFFSET = Skalenanfang
AnaFACTOR = Skalenende - Skalenanfang

Beispiel
Die Messwerte 0...20 mA sollen abgebildet werden auf 200 °C... 300 °C
AnaOFFSET = 200
AnaFACTOR = 300 - 200 = 100

Aufruf : AnaUSEL <kanal> [=<wert>] Ana-Einheit, <wert>:
0 = keine
1 = EEinh
2 = PEinh

Aufruf : AnaFIX <kanal> [=<wert>] Fixpunkt des Analogwertes (U1600: n.v.)
0 : 0.
1 : 0.0
2 : 0.00
3 : 0.000
9 : Fließkomma
(Fließkomma bei <wert> : 4 ... 9)

Aufruf : AnaSSEL <kanal> [=<wert>] +/–Bereich, <wert>:
0 = +/–, 1 = +, 2 = –

Aufruf : AnaRESO <kanal> [=<wert>] Auflösung in Messpunkten.
Vorgabe: 2000

Aufruf : AnaModID <kanal> Modul-Typ
(nur Abfrage)

Aufruf : AnaModSN <kanal> Modul-Seriennummer
(nur Abfrage)

Aufruf : AnaModDC <kanal> Modul-Datumcode
(nur Abfrage)

Aufruf : AnaCAL [*]
<kanal> <m> [=<wert>] Modul-Kalibrierung

Aufruf : AnaModSel <kanal> [=<wert>] Interne Funktion:
Auswahl der Modul E/A-Option

Aufruf : AnaRelMap
<modul> [=<relais>] Umlenken von Relais-Modulen auf Relais-Nummern

<relais>
0 : Identität
1 ... 7 : <modul> wird <relais> zugeordnet

Aufruf : AnaT

Test auf Ana-Aktivität.
(nur U1615)
Stack : – >>> {0 | 1}
1 : Anna aktiv,
0 : nicht.
Keine Ausgabe

Aufruf : AnaINT
<kanal> [= <wert>]

Mittelwertbildungsintervall in Sekunden (*)
0 : Standard -Momentanwert
1... 60: Schleppzeiger-Intervall

Aufruf : AnaMODE <kanal> [=<mode>]

<mode>	EGES	PMOM	ANA
0 OFF :	*	*	–
1 ANA :	*	*	ana
2 P →E:	eges(ana)	ana	ana
3 COUN:	eges(count)	pmom(count)	ana/count
4 LON :	LON	LON	ana
5 LonA:	*	*	ana
6 L-PE:	LON	LON	ana
7 LonI:	*	*	ana
8 LonR:	*	*	ana

(* : unberührte Basisfunktion)

AUFZ : gibt Liste der mit <aufzählung> angegebenen Kanalnummern aus **AUFZ**

Aufruf : AUFZ <aufzählung>
Stack : – >>> Anzahl der Elemente
Ext : + – # # %

– Bereich von <aufzählung> 1 ... 64 oder 0 ... 63
– Beispiele für <aufzählung> siehe PARAMETER

Spezifische Aufzählungen

<aufz>

*AA	: alle Analog-Ausgangsmodule	(AnaMODID == 2)
*AE	: alle Analog-Eingangsmodule	(AnaMODID == 1)
*EN	: alle ENergie-Kanäle	(KMODE == 2..4)
*EV	: alle möglichen EventApplikations	(EVENTAPP)
*ERR	: alle Kanäle mit Fehler	(ERRKAN <> 0)
*ERIS	: alle Kanäle InService	(ERRKAN-24)
*LA	: alle LON-Analog-Eingangskanäle	(KMODE == LonAna)
*LI	: alle LON-Binäre-Eingangskanäle	(KMODE == LonInp)
*LO	: alle LON-Energiezähler-Kanäle	(KMODE == LOn)
*LR	: alle LON-Relais-Kanäle	(KMODE == LonRel)

AUFZ@

AUFZ@ : legt Auffzählungs-Zahl auf den Stack (ein Bit pro Kanal → 64 Bit)

Aufruf : AUFZ@ <aufzählung>
Ausgabe : nein
Ext : @
Stack : – >>> <aufzZahl_33_64> <aufzZahl_01_32>

– Befehle, die eine Aufzählung akzeptieren, können per spezieller Stack-Referenz ".." die beiden Stackwerte <aufzZahl_33_64> und <aufzZahl_01_32> vom Stack holen und auswerten.
Beispiel: AUFZ@ 1..4, EGES ..
– das 1. Bit (LSB) entspricht Kanal 1, das 32. Bit entspricht Kanal 32 in <aufzZahl_01_32>, Bits 33-64 sind in <aufzZahl_33_64>.
– U1600/10/15 kennen nur Aufzählungen mit 32 Bits (0..31 oder 1..32); es wird nur ein Element über den Stack ausgetauscht.

Befehl:	Stack U1600/10/15:	Stack U1601/2/3/...:
AUFZ@ :	– >>> <aufzZahl_01_32>	– >>> <aufzZahl_33_64> <aufzZahl_01_32>
EGES .. :	<aufzZahl_01_32> >>> –	<aufzZahl_33_64> <aufzZahl_01_32> >>> –

– Bereich von <aufzählung> : 1..64
– Befehle, die Aufzählungen ab Null erwarten (PLIST, ALIST...), interpretieren die <aufzählungszahl> um einen Kanal verschoben. Die Ext. '@' oder die explizite Aufzählung ab Null erzwingen richtige Bitpositionen:
AUFZ@@ 2..4, PLIST .. → listet P-Prog. 2..4
AUFZ@ 2..4, PLIST .. → listet P-Prog. 1..3
AUFZ@@ 0..4, PLIST .. → listet P-Prog. 0..4
AUFZ@ 0..4, PLIST .. → listet P-Prog. 0..4

BUS, BUSL, BUSR

BUS BUSL BUSR : ECS-LAN Status

Ausgabe der Anzahl der Teilnehmer am ECS-LAN: Gesamtanzahl, Anzahl BUS-Links (davon direkte Nachbarn), Anzahl BUS-Rechts (...)

Aufruf : BUS
Stack : – >>> <anzahl_Bus_Teilnehmer>
BUS: Gesamtanzahl
BUSL: Anzahl links, oder –1 falls BUS-L-Fehler
BUSR: Anzahl rechts, oder –1 falls BUS-R-Fehler
Ext : + – . #

Beispiele

BUS : Busteilnehmer gesamt = 8, BL = 3(1), BR = 4(4)
BUS. : 8; 3; 1; 4; 4; 0; 0 [letzten beiden Werte: L-; R-Fehler (1:Fehler)]
BUS# : 8
BUSL# : 3

Aneinanderfügen von Clipboard-Einträgen.

Aufruf	: CHAIN	Beginn der Verkettung
Ausgabe	: nein	Ende der Verkettung
Stack	: – >>> –	
Ext	: $	

– CHAIN ist verfügbar in Firmware-Versionen ab dem 16.05.1999.

– So einfach die Ausgabe eines kanal-orientierten Befehls mit der Ext. "##" von "mehrzeilig" (Werte mit <CR><LF> getrennt) in einzeilig (Werte mit Semikolon getrennt) umschaltbar ist, so aufwendig ist die gleiche Ausgabe-Form, wenn die Daten von mehreren Befehlen kommen (siehe Beispiel bei DELI). Mit CHAIN wird der Ausgabefunktion mitgeteilt, dass der Ausgabe-Kopf jedes Befehls mit Doppel-Ext. (z.B. "##", ".." oder "%%") nur beim ersten Befehl nach CHAIN der <recordtrenner> ist, ansonsten der <feldtrenner>. Die Ext.'+' wirkt nur beim ersten Befehl nach CHAIN.
Beispiel: CHAIN,KANAL## 1,ZKONST## 1+3,URAT## 1,!## Test -->
Kanal- 1;100;640;1;Test

– Nach CHAIN werden alle mit Ext. '--' erzeugten Clipboard-Ausgaben aneinander gehängt (Gesamtlänge 128 Zeichen). Ohne CHAIN wird die Clipboard-Ausgabe pro Befehl neu initialisiert. Die Clipboard-Referenz '$' bezieht sich auf das Clipboard vor Aufruf von CHAIN. Bevor also das verkettete Clipboard ausgegeben werden kann, muss mit CHAIN- die Verkettung beendet werden.
Beispiel:

```
!-- "eins",!-- "zwei",!$                    -->zwei
CHAIN,!-- "eins",!-- "zwei",CHAIN-,!$     -->eins
                                             zwei
```

– Die Ext. '+' funktioniert, wie oben beschrieben, nur beim ersten Befehl nach CHAIN. Um lediglich Strings zusammen zu führen, wird CHAIN$ verwendet. CHAIN$ modifiziert die Ausgabekopfverarbeitung nicht.

```
CHAIN,!-- "eins",!--+ "zwei",CHAIN-,!$   -->eins
                                             zwei
CHAIN$,!-- "eins",!--+ "zwei",CHAIN-,!$  -->eins
                                             zwei
```

DATUMFORMAT

DATUMFORMAT (DATUMFOR) : Wählt das Datumdformat für alle Datumsausgaben:

Aufruf : DATUMFORMAT [=<dformat>]
Ausgabe : ja
Ext : + – # . $ %

– Befehle mit Datumsausgabe: Übergehen des aktuellen Formats mit Ext ~
– Mögliche Werte für <dformat>:
  tt.mm.jj (dd.mm.yy) → 31.12.93: ~
  mm/tt/jj (mm/dd/yy) → 12/31/93: ~~
  mm–tt–jj (mm–dd–yy)→ 12–31–93: ~~~
– Nur die ersten 2–3 Zeichen müssen angegeben werden {tt mm/ mm–}.
– Ext 'I'pusht beim Lesen den aktuellen Format-Index auf den Stak.
  DATUMFOR I : – >>> <format index> (U1600: n.v.)
– LISTDATUMFORMAT erzeugt eine Liste aller verfügbaren Formate (U1600: n.v.)

DELIMITER

DELIMITER (DELI) (DL) : Einstellen der Delimiter für den Rest der Befehlszeile

Aufruf : DELI [[<feldtrenner>] [_ _<recordtrenner>]]
Ausgabe nein
Stack : – >>> –
Ext : *

– DELI ohne Parameter setzt auf Defaultwerte:
  <feldtrenner> = ';' <recordtrenner> = "\r\n" [<CR><LF>]
– Beide Delimiter können bis zu 8 ASCII-Zeichen enthalten.
– Auch leere Delimiter sind möglich: DELI \000 \000 (U1600: b.v.)
– Zur Wahrung einer einheitlichen Programmumgebung gelten die neuen Delimiter nur für den Rest der Befehlszeile, danach gelten wieder die Defaultwerte.
– Ext * kehrt Parameter-Reihenfolge um: DELI* <recordtrenner>[__<feldtrenner>]
– Wird nur ein Parameter genannt, bleibt der andere unverändert.
– Bei Ext. '.' gilt: <recordtrenner> = <feldtrenner> (U1600: b.v.)
– Bei Ext. '#' gilt: <feldtrenner> = <recordtrenner> (ab 02/2002)
– Ext. '+' unterdrückt den nächsten <recordtrenner> der Ausgabe (U1600: b.v.) Bitte beachten Sie, dass Umleitungen in das Clipboard ohne Ausgabe (mit Ext. '--') den ersten <recordtrenner> meistens enthalten, wodurch die gewünschte Unterdrückung in der Ausgabe nicht (mehr) stattfinden kann. Abhilfe: Umleitung mit Anhängen (Ext. '--+')

Beispiele

- Statt der normalen ';' Trenner soll für die dBASE-Anweisung APPEND FROM ... DELIMITED mit ',' getrennt werden. Strings sollen mit "" eingefaßt werden (Ext. '$'):
  DELI ","; EGES.$ 1 → "EGes",1,"Kanal-1",127.34,"kWh"
- Mehrere verschiedene Werte sollen mit Semikolon getrennt ausgegeben werden. Hierzu wird <recordtrenner> = <feldtrenner> gesetzt und das erste Semikolon unterdrückt (Merke: Ausgaben beginnen stets mit <recordtrenner>):
  !!,DELI.+, KANAL# 1,ZKONST# 1+3,URAT# 1,! Test → Kanal-1;100;640;1;Test

DELTA : Einmalige Addition eines Energiequants zu einem Kanal. Die Funktionsweise entspricht einer einmaligen Anwendung von DVIRT mit einem bestimmten Energiebetrag

Aufruf : DELTA <aufzählung> [<faktor>] = <wert>
Stack : – >>> –
Ext : |

DELTA| ... : nur positive Quanten werden beachtet
DELTA| | ... : nur negative Quanten werden beachtet

– STARTSTOP ermöglicht (STSP == 1) oder ignoriert (STSP == 0) die Zuweisung.
– Der <wert> kann mit einem optionalen <faktor> bewertet werden (U1600: n.v.).

DevKEY : Abfragen des Geräte-Schlüssels (Device Key) **DevKEY**
Eingabe des Öffnungs-Codes

Aufruf	: DEVKEY		DEVKEY = <openCode>
Funktion	: Abfrage des Schlüssels		Öffnen (siehe unten)
Ausgabe	: ja	nein	nein
Stack	: – >>> <systemKey>		– >>> –

– Mit DevKEY können geschützte Geräte-Einstellungen (z.B. Passwörter) initialisiert werden (gelöscht) werden.
Vorgehensweise:
Abfrage des aktuellen Geräte-Schlüssels: DEVKEY
Bitte teilen Sie diesen Schlüssel Ihrem Geräte-Lieferanten mit und verlangen einen Öffnungs-Code (zu genau diesem Schlüssel) für eine bestimmte Aufgabe (z.B. Löschen aller ECL- und Panel-Passwörter).
Eingabe des Öffnungs-Code: DEVKEY = <openCode>
– **Wichtig:** DEVKEY arbeitet systemweit, Sie müssen also auf die richtige Adressierung des DEVKEY-Befehls achten. Sobald die Öffnungs-Code-Zuweisung korrekt erfolgte, ist der Geräte-Schlüssel und damit auch der Öffnungs-Code NICHT MEHR GÜLTIG! Ist ein weiterer Öffnungs-Code notwendig, so muss ein NEUER Schlüssel mit DEVKEY abgefragt werden, der gesamte Vorgang wiederholt sich. Solange ein Schlüssel gültig ist, kann er beliebig oft abgefragt werden.

DIR, DIRN, DIRS

DIR DIRN DIRS : Verzeichnis aller ECS-LAN Teilnehmer

– DIR mit ESC-LAN Info: siehe DIRS

– Die Liste der Teilnehmer lässt sich durch Angabe einer Auswahl einschränken. (1600 n.v.)

<auswahl>	:	{ A | B | U | R | P | * }
DIR	:	nur KENNUNG aller Teilnehmer (A:)
DIRN	:	KENNUNG und STATIONSNAME aller Teilnehmer (A: U1601)
DIRN_	:	KENNUNG und STATIONSNAME der angewählten Stationen
Stack	:	– >>> –
Ext	:	+ – # . %
DIRS	:	wie DIR, aber mit Info je Kennung. L = Links, R = Rechts, + = direkter Nachbar, * = „Ich“
Stack	:	– >>> <anzahl_BUS_Teilnehmer>
Ext	:	+ – # .
*	:	alle Stationen (kann weggelassen werden)
U	:	alle U16xx Stationen (auch AB oder BA möglich)
A	:	alle U1600/10/15 Stationen
B	:	U1601 Stationen

DISPLAY

DISPLAY (DD) : Ausgabe der Anzeige

Aufruf	:	DISPLAY [<tastenkürzel ...>] <tastenkürzel ...> : siehe **TASTE**
Stack	:	– >>> –
Ext	:	+ – # |
Ausgabe	:	U1600: 3 Zeilen: 1.+2. LCD-Zeile + 1 Zeile mit Status der 8 LEDs [LAN/L LAN/R R1 R2 R3 R4 STATUS STAT24V] U1601: 17 Zeilen: 1.-16. LCD-Zeile + 1 Zeile mit Status der 4 LEDs und Status der 6 Relais [STATUS LAN/L LAN/R LON R1 R2 R3 R4 R5 R6]

– Der Cursor im Display wird durch Vorstellen eines '&' kodiert.

– Codierung in der LED/RELAIS-Informationszeile:
'-' : AUS,
'*' : EIN,

– Ext # : Ausgabe ohne " “-Anführungszeichen

– Ext | : zeigt nur die LED-Zeile (U1600: b.v.)

– Ext | | : zeigt nur die Text-Zeilen (U1600: b.v.)

– Dieser Befehl gilt auch für U1602 und U1603, obwohl diese nicht über eine reale Anzeige verfügen.

DUP (DU) DROP (DR) SWAP (SW) PICK : Stack-Manipulationen

DUP, DROP, SWAP, PICK

DUP	: n1 n2	>>> n1 n2 n2
DROP	: n1 n2 n3	>>> n1 n2
SWAP	: n1 n2	>>> n2 n1

DUP <n> : führt DUP n-mal durch
DROP <n> : führt DROP n-mal durch
PICK <i> : kopiert i-tes Stackelement an 1. Position.
'PICK 1' == 'DUP'

dVSUM dVIRT : Beschreibungs-Funktionen zur Bildung von virtuellen Kanälen (im Hintergrundprogramm) mit „differenzieller Summenbildung".

DVSUM, DVIRT

dVSUM : summiert Kanäle der Aufzählung in Zwischenregister
Aufruf : dVSUM <aufzählung> [<faktor>]
Stack : – >>> –

dVIRT : weist die Zwischenregister-Diff.-Summen den virtuellen Kanälen zu
Aufruf : dVIRT <aufzählung> [<faktor>] =
Stack : – >>> –
Ext : + * |

dVIRT+ ... : die berechnete Eneregie wird der vom Zielkanal gemessenen Energie hinzu addiert. Ohne Ext '+' hat die gemessene Energie keine Auswirkung.

dVIRT* ... : normalerweise werden nach DVIRT die Zwischenregister gelöscht. Mit der Ext '*' bleiben sie jedoch unberührt.

– dVIRT kann für alle Kanäle angewendet werden, jedoch pro Kanal nur ein einziges Mal. DVIRT darf nur in H-Programmen ausgeführt werden und kann NICHT mit IF ...THEN ... ELSE beeinflußt werden.
– Bei Angabe eines <faktor>s wird die Teil-Energie/Leistung mit diesem multipliziert. Achtung, dieser <faktor> muss statisch sein, darf also nicht vom Programmablauf verändert werden.
– Ext. 'I' ist nur bei DVIRT möglich.
Bei Verwendung dieser Ext. werden nur positive ('I') oder nur negative ("II") Quanten und Leistungen beachtet. Einsatzbeispiel: Ein Basiskanal wird aufgespalten in einen Bezugs- und einen Abgabekanal. Einschränkungen: die Funktion arbeitet nur dann erwartungsgemäß, wenn genau ein Summenkanal verwendet wird; eine Verwendung mit mehreren Summenkanälen führt u.U. zur permanenten Addition/Subtraktion von kleinsten Bilanzunterschieden. In Kombination mit Ext. '+' werden nur die mit DVSUM/DVIRT gebildeten Energie-Quanten und Leistungen eingeschränkt, nicht aber die vom Zielkanal direkt gemessene Energie.
– Die Momentanleistung PMOM des gebildeten Kanals entspricht der Summe der Momentanleistungen der Basiskanäle. Wenn die DVIRT-

Bildung für länger als 30 s unterbrochen wird, so wird PMOM auf Null oder den eff. Messwert gesetzt. Kürzere Unterbrechungen führen zu KEINEM DATENVERLUST an Energie-Quanten!

– Die Summen im virtuellen Kanal sind vollkommen unabhängig von den Basiskanälen (lose Kopplung). Gebildete Summen können direkt gelöscht werden.
– STARTSTOP kann deshalb unabhängig von den Basiskanälen verwendet werden.
– Einmalige Addition eines Energiequants zu einem Kanal: siehe DELTA
– Effizienz: Jeder DVSUM Befehl verwendet ein ESC-LAN-Telegramm, unabhängig von der <basis _aufzählung>. Der DVIRT Befehl jedoch verwendet pro Kanal der <ziel_aufzählung> ein ESC-LAN-Telegramm.

Beispiele

– Der Kanal 26 von Station D: bildet eine Kostenstelle aus den Kanälen 1 ... 5+8 der Station B: (bewertet mit 0.7) und dem Kanal 4 der Station C: (bewertet mit 0.3):
H 1 = 'B:DVSUM 1 .. 5+8 0.7, C:VSUM 4 0.3. D:VIRT 26='
– Der Kanal 10 entspricht der Bilanz der Kanäle 1 ... 8 und dem Gesamtsummen-Kanal 9 (Summe 1 ... 8 abzüglich Kanal 9):
H 2 = 'DVSUM 1 .. 8, DVSUM 9 –1, DVIRT 10=´

EEINH, PEINH, AEINH, TEINH

EEINH PEINH AEINH TEINH: Einheiten der Energie E, Leistung P, Analog-E/A und des Tarifs (max. 4 Zeichen)

Aufruf	:	EEINH <aufzählung> [=<zeichenkette>]
Ausgabe	:	ja
Ablage	:	<einheit>
Stack	:	– >>> –
Ext	:	+ – . # $ %

– Zeichenvorrat siehe KANAL.
– Damit die Analog-Einheit angewendet werden kann, muss ANAUSEL <kan> = 3 gesetzt werden.
– AEINH bei U1600/10/15 nicht verfügbar, Antwort ist ein Leerstring.

EGES, EGEST1, KOSTT1, EGEST2, KOSTT2, EGEST1T2, KOSTT1T2, PMOM

Eges EgesT1 KOSTT1 EgesT2 KOSTT2 EgesT1T2 KOSTT1T2 PMOM:

Eges	:	Gesamt-Energie
EgesT1	:	Gesamt-Energie Tarif 1
KOSTT1	:	Kosten Tarif 1
EgesT2	:	Gesamt-Energie Tarif 2
KOSTT2	:	Kosten Tarif 2
EgesT1T2	:	Gesamt-Energie Summe Tarif 1+2
KOSTT1T2	:	Kosten Tarif 1+2
PMOM	:	momentane Leistung (P)ower
Aufruf	:	EGES <aufzählung> [=<neuerWert>]
Stack	:	– >>> <wert> (Bei Aufzählungen ist <wert> = Summe(Einzelwerte))

Ext : + – . # / * _ | $ %

– Statt der Energie kann mit der Ext: * auch die (berechnete) Impulsanzahl ausgegeben werden.
  Beispiel: 'EGES* 1'
  Zur Berechnung wird Zkonst, Urat und Irat des entsprechenden Kanals verwendet (auch bei virt. Kanälen).
– Ext | gibt nur <wert> und <einheit> aus:
  Eges | 1→ 123.34 kWh
– <neuerWert> == 0 → Zeitinfo wird zusätzlich gelöscht (VON = 0, BIS = 0)

Messung der Momentanleistung PMOM

– Wenn der Eingang als Zähler parametriert ist, so wird die Momentan-Leistung aus dem Abstand der letzten beiden Impulse ermittelt. Pulsabstände von > 130 s bewirken ein Löschen von Pmom. Wenn die Zeit seit dem letzten Impuls größer als der letzte Pulsabstand ist, so wird diese als Bezug zur Pmom-Messung verwendet.
  Der PFAKTOR geht in die Berechnung der Momentanleistung mit ein.

Die von PMOM mit Ext '/'oder '//'gelieferte Zeitinformation ist wie folgt zu interpretieren:

PMOM//	: VON_Zeit	:	Zeitpunkt der letzten Energie-Änderung, entspricht Zeitpunkt des letzten Zählimpulses.
PMOM/	: BIS–Zeit	:	aktuelle Zeit

– Die VON-Zeitinformation (Auflösung 1s) wird auf Basis von LASTUPD (siehe unten) bebildet. Wird eine Auflösung < 1s benötigt, muss LASTUPD angewendet werden. Zeitspannen größer 20 Tage liefern VON–ZEIT=0 (01.01.1990 00:00:00)
– U1600/10/15: PMOM liefert keine brauchbare Zeitinformation (VON=0, BIS=0)

LASTUPD (LUPD)	: Zeitdauer [S] seit letztem Kanal-Update (U1600: n.v.)	
Aufruf	: LASTUPD <aufzählung>	
Ausgabe	: ja	
Stack	: – >>> <wert>	(Bei Aufzählungen ist <wert>=Summe (Einzelwerte))
Ext	: ! + – . # %	

– Mit LASTUPD kann die Zeitspanne in Sekunden (Auflistung 1ms) vom letzten Kanal-Update (Energie änderte sich) bis jetzt ermittelt werden.
– Die maximale Zeitspanne beträgt 20 Tage. Für längere (beliebig lange) Zeitspannen gilt: <wert> = 1728000 s == 20 Tage.
– Bei Kanälen, die über keine Zeitspannenberechnung verfügen, ist <wert> = 0.

EINT, PINT

EINT PINT : Energie und Leistung im Intervall

Aufruf	:	EINT <aufzählungKanal> [<startindex>] [<wieviele>] [=<neuerWert>]
Stack	:	– >>> <wert> (Bei Aufzählungen: <wert> = Summe(Einzelwerte))
Ext	:	+ – . # / * _ | $ % @

Die Ausgabe erfolgt ab <startindex> in zeitlich aufsteigender Reihenfolge
<startindex> ohne Angabe → <startindex> = 0
<startindex>=='*' → <startindex> = zeitlich erster Eintrag
<wieviele> ohne Angabe → <wieviele> = 1
<wieviele>=='**' : Alle Einträge ab <startindex> (mit Eint–0)
<wieviele>=='*' : Alle Einträge ab <startindex> (OHNE Eint–0)
<wieviele> größer <startindex> : "" " "

– Suchen eines bestimmten Index: siehe INDEX
– Die Datenliste muss für eine Kanalauswahl formatiert (#-Aufzählung) werden. Formatieren und Formatinfo-Abfrage erfolgen mit dem Befehl FORMAT.
– Die Einträge in der Datenliste (außer dem aktuellen Wert) sind zu Lasten der Genauigkeit komprimiert (siehe FORMAT).
– zur schnellen Datenübertragung eignet sich ausschließlich: EINT##

Leistungsoptimierung

– PINT@ <aufzählung> gibt den gemittelten Leistungswert des gerade laufenden Intervalls mit höherer Genauigkeit aus (innerhalb der ersten Sekunden), da der Zeitbezug auch Millisekundenanteile berücksichtigt. Die normale Leistungsberechnung (PINT <aufzählung> arbeitet stets mit ganzen Sekunden.
Für eine Leistungsoptimierung ist es wichtig, daß am Anfang des Intervalls die Daten sich relativ schnell einschwingen und keine großen Überschwinger aufweisen. Bei einer solchen Anwendung sollten Sie PINT@ ... verwenden. Auf U1600/10/15-Stationen darf PINT@ erst ab Version 1.63m (auch bei Zugriff über das ECS-LAN) verwendet werden.

ERR, LERR, LBERR, ERRNR, ERRSTAT, ERRSTATLIST, ERRKAN, ERRKANLIST

ERR LERR LBERR ERRNR ERRSTAT ERRSTATLIST ERRKAN ERRKANLIST

ECL-Interpreter-Fehler

ERR	:	Ausgabe des Fehlerzustandes der Hintergrundprogramme
Aufruf	:	ERR
Ausgabe	:	ja
Stack	:	– >>> –
Ext	:	_ (unterdrückt die "Keine Fehler in ..." Meldung)

LERR	:	Last Error, Fehler-Nummer von letzter Ausführung des aktuellen H-Programms.
Aufruf	:	LEER
Ausgabe	:	nein
Stack	:	– >>> <n> <n> : Fehler-Nummer siehe unten oder 0 (kein Fehler)

LBERR : Last Bus Error; wie LEER, es werden jedoch nur ECS-LAN relevante Fehler betrachtet (sonst 0).

ERRNR : Error-Nummer → Beschreibung
Aufruf : ERRNR <aufzählung> ERRNR
Ausgabe : ja
Stack : – >>> – <ErrNr> >>> –
Ablage : Fehlerbeschreibung (ErrNr)
Ext : + – . # $ %

– **ERRLIST** entspricht **ERRNR** (U1600: n.v.)

Err000: O.K.
Err001: Exit
Err002: Allgemeiner Fehler
Err003: Syntax-Fehler
Err004: Fehler: Zu wenig Parameter
Err005: Fehler: Zu viele Parameter
Err006: Fehler: Argumentbereich falsch
Err007: Fehler: Zahl zu groß
Err008: Fehler: Division durch Null
Err009: Fehler: Zu viele Programm-Verschachtelungen
Err010: Fehler: Zu viele IF/ELSE-Verschachtelungen
Err011: Fehler: Zu viele FOR-Verschachtelungen
Err012: Fehler: ALL-Verschachtelungen nicht möglich
Err013: Fehler: Funktion nicht verfügbar
Err014: Fehler: Nur virtuelle Kanäle erlaubt
Err015: Fehler: Keine virtuellen Kanäle erlaubt
Err016: Fehler: außerhalb Indexbereich
Err017: Fehler: Zuweisung nicht möglich
Err018: Fehler: Falsche Zeit/Datum-Angabe
Err019: Fehler: Extension nicht anwendbar
Err020: Fehler: Suchbegriff nicht gefunden
Err021: Interner Fehler
Err022: Fehler: Nur in H-Prog. anwendbar
Err023: Kein Zugriffsrecht
Err024: Fehler: Eingabezeile zu lang
Err025: Fehler: Kennung falsch
Err026: Fehler: Teilnehmer nicht bekannt
Err027: Fehler: Bus Timeout
Err028: Zugriff verweigert
Err029:
Err030:
Err031:

Stations-Fehler

ERRSTAT : Abfrage des aktuellen Stations-Fehlerzustandes
Aufruf : ERRSTAT [<fehler maske aufz>]
Aufruf : ERRSTAT@ [<fehler maske>]
Ausgabe : ja
Stack : – >>> <fehler-word>
Ext : _ (unterdrückt die "Keine Fehler in ..." Meldung)

– Angabe der zu betrachtenten Fehler per Aufzählung:
<fehler maske aufz>,
32-Bit Zahlenangabe (LSB gesetzt: Fehler 1 betrachten ...):
<fehler maske>
– Keine Angabe von [<fehler maske aufz>] oder [<fehler maske>]:
ALLE Fehler
– Bei U1600/10/15 werden nur folgende Fehler erkannt:
8: Interner Batt.-Fehler, 11: Uv Ausfall,
21: LAN/L Fehler, 22: LAN/R Fehler

ERRSTATLIST : Liste aller möglichen Stations-Fehler

Aufruf : ERRSTATLIST <fehler nr aufz> ERRSTATLIST
Ausgabe : ja
Stack : – >>> – <fehler-nr>
Ablage : Fehlerbeschreibung (fehler_nr)
Ext : + – . # $ %

ErrStat-01: Selbsttest-Fehler
ErrStat-02: ROM-Fehler
ErrStat-03: RAM-Fehler
ErrStat-04: EEPROM-A-Fehler
ErrStat-05: EEPROM-B-Fehler
ErrStat-06: Anwender-Fehler-A
ErrStat-07:
ErrStat-08: Interner Batterie-Fehler
ErrStat-09:
ErrStat-10:
ErrStat-11: Uv Ausfall
ErrStat-12:
ErrStat-13:
ErrStat-14: COM1 Kommunikations-Fehler
ErrStat-15: COM2 Kommunikations-Fehler
ErrStat-16: COM3 Kommunikations-Fehler
ErrStat-17: LAN Kommunikations-Fehler
ErrStat-18:
ErrStat-19:
ErrStat-20:
ErrStat-21: LAN/L Fehler

ErrStat-22: LAN/R Fehler
ErrStat-23: LON Fehler
ErrStat-24:
ErrStat-25:
ErrStat-26:
ErrStat-27:
ErrStat-28:
ErrStat-29: Batterie fast verbraucht
ErrStat-30:
ErrStat-31:
ErrStat-32:

Kanal-Fehler

ERRKAN	:	Abfrage des aktuellen Kanal-Fehlerzustandes
Aufruf	:	ERRKAN <kanal_aufz> [<fehler_maske_aufz>]
Aufruf	:	ERRKAN@ <kanal_aufz> [<fehler maske>]
Ausgabe	:	ja
Stack	:	– >>> <fehler-word> (geodert aus allen <fehler word>s)
Ext	:	_ (unterdrückt die "Keine Fehler in ..." Meldung)
		* (Fehlerangaben für alle Kanäle zusammen, nicht aufgetrennt)

– Angabe der zu betrachtenten Fehler per Aufzählung:
<fehler_maske_aufz>,
32-Bit Zahlenangabe (LSB gesetzt: Fehler 1 betrachten ...):
<fehler_maske>
– Keine Angabe von [<fehler_maske_aufz>] oder [<fehler_maske>]:
ALLE Fehler
– Mit "ALL, ERRKAN&_ *" werden alle Kanalfehler systemweit aufgelistet.
Es wird pro Kanal ein Telegramm benötigt, bei m Stationen und 64 eingeschalteten Kanälen sind das also m*64 Telegramme.
– Zur deutlich schnelleren Abfrage aller Kanalfehler kann die ab V2.45 verfügbare gerätespezifische Aufzählung "*ERR" verwendet werden:
ALL, ERRKAN&_ *err
Alle U1601/2/3 Stationen im Netz benötigen V2.45 oder größer, sonst liefert "*ERR" für ältere Geräte eine leere Liste.
– ERRKAN* (Ext.'*') benötigt nur ein Telegramm pro Station, um alle Kanalfehler zur gegebenen Fehlermaske zu zeigen. Eine Kanalzuordnung ist hier aber nicht möglich (Kanalinfo = 0).
– U1600/10/15 liefern stets "keine Fehler".

ERRK-ANLIST	:	Liste aller möglichen Kanal-Fehler
Aufruf	:	ERRKANLIST <kanal_aufz> <fehler_nr_aufz>ERRKANLIST <kanal_aufz>
Ausgabe	:	ja
Stack	:	– >>> – <fehler_nr> >>> –
Ablage	:	Fehlerbeschreibung (fehler_nr)
Ext	:	+ – . # $ %

ErrKan- 1: Kommunikations-Fehler
ErrKan- 2: Unbekanntes Gerät
ErrKan- 3: Selbsttest-Fehler
ErrKan- 4: Kalibrierungs-Fehler
ErrKan- 5: Authentifizierungs-Fehler
ErrKan- 6: Offline
ErrKan- 7:
ErrKan- 8:
ErrKan- 9: Fühlerbruch
ErrKan-10: Phasenausfall
ErrKan-11: Drehfeld-Fehler
ErrKan-12: Überlauf
ErrKan-13:
ErrKan-14:
ErrKan-15:
ErrKan-16: Drahtbruch 4-20mA
ErrKan-17: Oberer Grenzwertalarm 1
ErrKan-18: Unterer Grenzwertalarm 1
ErrKan-19: Oberer Grenzwertalarm 2
ErrKan-20: Unterer Grenzwertalarm 2
ErrKan-21:
ErrKan-22:
ErrKan-23:
ErrKan-24: In Service
ErrKan-25: Parametrierungs-Fehler
ErrKan-26: Kalibrierungs-Aufforderung
ErrKan-27:
ErrKan-28:
ErrKan-29:
ErrKan-30:
ErrKan-31:
ErrKan-32:

ETAG, EMON, EJAHR, EMAX
PTAG, PMON, PJAHR, PMAX

ETAG EMON EJAHR EMAX PTAG PMON PJAHR PMAX

	im Intervall	pro Tag	pro Monat	pro Jahr
Energien	(siehe Eint)	ETag	EMon	EJahr
mittlere Leist.	(siehe Pint)	ETag	EMon	EJahr
Energie-Maxima	Emax (10 St.)	EmTag	EmMon	EmJahr
mittlere Leist.	Pmax (10 St.)	EmTag	EmMon	EmJahr
Listenlänge	variabel	10 + lfd. Tag	12 + lfd. Monat	2 + lfd. Jahr

Aufruf : ETag <aufzählungKanal> [<aufzählungIndex>] [=<neuerWert>]
Stack : – >>> <wert> (Bei Aufzählungen ist <wert> = Summe (Einzelwerte))
Ext : + – . # / * _ | $ %

– <aufzählungIndex> = 0 oder keine Angabe: laufender Zyklus
– <neuerWert> = 0 → Zeitinfo wird zusätzlich gelöscht (VON = 0,BIS = 0)
– EMAX <. .> <index> = 0 → Löschen ab <index> bis <maxIndex> (nur EMAX)
– Variabler Tages- / Monats- / Jahresanfang: TAGBEG MONBEG

EXIT RETURN

EXIT, RETURN

EXIT : vorzeitiges Beenden des laufenden Programmes
Aufruf : EXIT

RETURN : vorzeitiges Beenden nur des laufenden Unterprogrammes
RET : Kurzform von RETURN
Aufruf : RETURN

FDIR : File-Directory anzeigen (ab 12/2001)

FDIR

Aufruf : FDIR // komplette Informationen
FDIR# // nur Dateinamen
FIDR## // komplette Informationen, kompakte Darstellung

FREAD (FR) : Lesen aus Datei (Record-orientiert)

FREAD

Aufruf : FREAD <fname> <index> [<anzahl>]
FREAD <fname> <datumODERzeit> [<zeitODERdatum>] [<anzahl>]
Ausgabe : Ja
Stack : – >>> <startzeit> <index>
Ext : + - # . % & /
. : Binär-Codierung verwenden
| : verwende nächst jüngeren Record zum angegebenen Record.
_ : unterdrücke aktuellen Record
$: Ausgabe mit Checksum (ab 12/2001)
@ : statt <index> wird <zeitzahl> verwendet

– <fname> ist der Datei-Name (z.B. "E.S" für Gesamtenergie-Datei)
– Die Ausgabe erfolgt in Hex-Darstellung, 256 Bytes werden auf 512 Zeichen (Zeichenmenge: 0..9,A..F) abgebildet. Mit Ext.'.' wird die binärcodierte Ausgabeform gewählt.
– Die Ausgabe ist besonders zeitoptimiert, auch der Zugriff über mehrere Stationen im ECS-LAN verlangsamt die Geschwindigkeit nicht sonderlich.

FLIST (FL) : Lesen aus textorientierter Datei (ab 12/2001)

FLIST

Aufruf : FLIST <fname> [<index> [<anzahl>]]

FSIZE

FSIZE (FSxxx) : Datei-Grössen bestimmen

Aufruf	:	FSIZE <fname> <mode> FS<mode> <fname>
Ausgabe	:	Ja
Stack	:	– >>> <wert>
Ext	:	+ – # . % &

<mode>	<wert> = Grösse in	<mode>	<wert> =
B	Bytes	MB	MaxBytes
R	Records	MR	MaxRecords
P	Prozent	RS	RecordSize
S	Sekunden	FI	FirstIndex
T	Tage	LI	LastIndex
FT	FType (file type)	FN	FNum (file number)

– <fname> ist der Datei-Name (z.B. "E.S" für Gesamtenergie-Datei)
– FType (file type):
  0: umlaufende Datei (mit ActRecord)
  1: statische Datei
– FNum (file number) ist die intern verwendete Nummer der Datei.
– Befehls-Kurzform: fsize E.S B == fsb e.s
– Für <mode> kann das oben genannte Kürzel oder die komplette Bezeichnung verwendet werden.
– VON und BIS Zeiten werden entsprechend gesetzt.

FORI NEXTI, FORJ NEXTJ, FORK NEXTK

FORI I NEXTI (NI) FORJ J NEXTJ (NJ) FORK K NEXTK (NK) : Programm-Schleifen

FORI hat Zählvariable I, FORJ hat J und FORK hat K (U1600: n.v.).

Aufruf	:	FORI	FORI <aufzählung>	I	NEXTI
Ausgabe	:	nein	nein	nein	nein
Stack	:	<von> <bis> >>> –	– >>> –	– >>> I	– >>> –

Hinweis
Die Schleife wird mindestens einmal ausgeführt. Am Schleifenende (NEXTI oder Zeilenende) wird I um 1 hochgezählt oder auf das nächste Aufzählungselement gesetzt. Wenn kein Element mehr in der Aufzählung enthalten ist, oder wenn gilt: I > <bis>, so wird die Schleife nicht mehr ausgeführt.
Die Zählvariablen sind stets verfügbar und können auch auf einen bestimmten Wert gesetzt werden (z.B.: I = 15), jedoch kann mit einer Zuweisung von I eine "FORI <aufzählung>" Schleife NICHT

beeinflußt werden.
Jedes Unterprogramm hat seinen eigenen FORI / FORJ / FORK Satz.
Folgen nach dem NEXTI keine Befehle mehr, kann NEXTI weggelassen werden.
<aufzählung> kann den Bereich 0 ... 63 oder 1 ... 64 umfassen.
FORI* : Umdrehen der Abarbeitungs-Reihenfolge.

Beispiele

```
! Test:, 2,5, FORI, i, !+ " " ., nexti   ⇒ Test: 2 3 4 5
! Test:, FORI 2..5, i, !+ " " .          ⇒ Test: 2 3 4 5
! Test:, FORI 2..5, i, !+ " %ai"         ⇒ Test: 2 3 4 5
! Test:, FORI* 2..5, i, !+ " %ai"        ⇒ Test: 5 4 3 2
```

FORMAT : Formatieren der Datenliste (Abruf der Daten mit EINT)

– Die Formatierung teilt den Datenspeicher für eine bestimmte Kanalanzahl von Intervallmessdaten ein, die gesamte Speichertiefe hängt jedoch dynamisch mit der Länge des Synchron-Intervalls ab. Ist der Speicher voll, so wird rotiert: zu Gunsten eines neuen Eintrags wird der älteste Eintrag gelöscht.
– Eine Neuformatierung (mit Löschen) der Datenliste erfolgt nur, wenn der FORMAT-Anweisung eine <aufzählung> zugewiesen wird. Ohne Zuweisung gibt FORMAT die aktuelle Formatierung aus, auf den Stack kommt die Anzahl der speicherbaren Kanäle.
– Jede beliebige Aufzählung (auch mit virtuellen Kanälen) ist möglich
– Die Einträge in der Datenliste (außer dem aktuellen Wert) sind zu Lasten der Genauigkeit in einem 2-Byte-Wert komprimiert. Ab ECSys V1.60 sind verschiedene Kodierungen des Datenbereichs möglich, um den Arbeitsbereich optimal anzupassen (siehe unten).
  Achtung: ältere ECSys Versionen verstehen die neuen Kodierungen nicht!
– Bei der Auslieferung des Gerätes gilt: FORMAT = 1 . . 64 0.

Aufruf	: FORMAT	FORMAT@
Stack	: – >>>	<kanal_anzahl> – >>> <kodierung> <kanal_anzahl>
Ausgabe	: Format-Info (nicht bei . und # ##), sowie die Kanalliste	
Ext	: + – . # ## @	

Aufruf	: FORMAT = <aufzählung>	Formatieren gemäß Kanalangabe und aktueller Kodierung
Stack	: – >>> –	

Aufruf	: FORMAT = <aufzählung> <Kodierung>	Formatieren mit Vorgabe einer Kodierung
Stack	: – >>> –	

Kodierungen des Datenbereiches: (0: Standard. Auflösung in [ ] angegeben)

0 : 0...+/-0.8191[0.0001]...+/-81.91[0.01]....+/-8191[1]...+/-819100[100]
1 : 0....+/-8.191[0.001]....+/-81.91[0.01]...+/-819.1[0.1]...+/-8191[1]
2 : 0....+/-16383[1].....+/-163830[10]
3 : 0.....+32767[1].....+327670[10]
4 : 0..+/-99999999 [8 Dezimalstellen, kleinste Stelle: 1E-6]
lst die Zahl > 99999999, so werden die führenden Stellen abgeschnitten.

1234567890	--> 34567890	Abschneiden der ersten 2 Stellen
12345678.9	--> 12345679	8. Stelle wird 5/4 gerundet
1234567.8	--> 1234567.8	keine Einschränkung
12.345678	--> 12.345678	keine Einschränkung
12.3456789	--> 12.345679	8. Stelle wird 5/4 gerundet
1.23456789	--> 1.234568	nur 6 Nachkommastellen (s.u.).

Hinweise

– Die Kodierungen 0,1,2,3 verwenden 2 Byte pro Eintrag, die Kodierung 4 jedoch verwendet 4 Byte pro Eintrag, hier wird also die Speicherdauer halbiert.

– Kodierung 4 ist erst ab V2.46 verfügbar, Intervalldaten mit Kodierung 4 können von anderen Stationen mit älterer Firmware nicht ausgelesen werden!

– Bei Kodierung 4 ist die kleinste Auflösung 1E-7, bei der schnellen Ausgabe mit Ext.'#' beträgt die Auflösung 1E-6 (die 6. Nachkommastelle wird bei Bedarf 5/4 gerundet).

Testen der Kodierung

Aufruf : DLVAL <w> [<dlcode>] (U1600: b.v.)
Ausgabe : ja
Stack : – >>> <w komprimiert>

HH, MM, SS, TAG, WTAG, MON, JAHR

HH MM SS TAG WTAG MON JAHR :Selektives Abfragen von Uhrzeit und Datum, Ergebnis auf den Stack

HH	: Stunde	(0... 23)
MM	: Minute	(o... 59)
SS	: Sekunde	(0... 59)
TAG	: Tag	(1... 31)
WTAG	: Wochentag	(1: Montag... 7: Sonntag)
MON	: Monat	(1... 12)
JAHR	: Jahr	(90... 99, 0... 20)
JAHR*	: Jahr	(1990... 1999, 2000... 2020)

Aufruf : HH : Basis = Systemzeit
Ausgabe : nein
Stack : – >>> <Stundenzahl>
Ext : ! _

Aufruf : HH . : Basis = Sek. Zahl vom Stack
Ausgabe : nein
Stack : <sekZahl> >>> <Stundenzahl>
Ext : ! _

Bedeutung der Extensions

– Die Extension '!' forciert eine Aufgabe:
  Bei HH!, MM!, SS!, TAG!, JAHR! wird die Zahl dargestellt, bei
  WTAG! : Tagesname
  MON! : Monatsname
– Die Extension '_' verwendet bei Aufruf ohne Argument die Betriebsdauerzähler-Zeit statt der Echtzeit. (U1600: n.v.)
– Ext '–' oder "– –" : Die Zahl oder die Antwort wird in die Zwischenablage geschrieben, keine Ausgabe.

Zeitbezug

– Alle oben aufgeführten Befehle OHNE Argument verwendet, beziehen sich auf die Echtzeit derjenigen Station, auf der der Befehl physikalisch interpretiert wird (als ob stets die Kennung AA: vorangestellt wäre). Siehe hierzu KENNUNG und ZEIT. Wollen Sie dennoch die Zeit einer bestimmten Station als Bezug verwenden, verfahren Sie wie folgt:
  C5:time–,SS! .: gibt die Sekunde (0... 59) der Station C5: aus

Beispiele

– Andere Wochentageszahl-Bedeutung (0: Sonntag... 6: Samstag):
  WTAG,7,MOD, !
– Sekundenanzahl des laufenden Tages:
  TIME–,86400,MOD, !
– SS-Befehl "zu Fuß" geschrieben:
  TIME–,86400,MOD,60,MOD, !
– MM-Befehl "zu Fuß" geschrieben:
  TIME–,86400,MOD,3600,MOD,60,/,INT, !

H-Programme HLIST HBREAK

H-Programme, HLIST, HBREAK

H0 . . H31 : HINTERGRUND-Programme ausführen / programmieren

H0 ... H31 (U1600: H0 .. H19) werden der Reihe nach im Hintergrund ausgeführt. Laufzeit-Fehler können mit ERR angezeigt werden. Ausgaben der Hintergrundprogramme werden immer an COM1 der Station geleitet, auf der das Hintergrundprogramm läuft.
(Ausnahme: Ausgabe-Umleitung bei U1600 mit COM-Modus "COM+MIX" bzw. bei U1601 mit COM-Modus "ECL+HP" auch auf COM2 möglich).

Ext : + – . # $! % ?

Aufruf	: H <aufzählung>	[=<zeichenkette>]
Ausgabe	: nein	(na bei den genannten Ext. außer '-' u. '?')
Ablage	: <programm>	(außer bei Ausführung)
Stack	: - >>> -	(außer bei Ext. '?', s.u.)
Ext.	: + - . # $! % ?	

– H ohne <index> == H 0, H1..H31 == H 1..H 31, H5! == H! 5
– Bei beliebiger Extension erfolgt keine Programm-Ausführung.
– Maximale Verschachtelungstiefe: 10 (U1600: 3) P-Programme, maximale Zeilenlänge: 128

– H 19 ist das "Druckprogramm", aktivierbar mit dem Bedienpanel. H 19 arbeitet nur in dieser Zeit im Hintergrund und ist sonst inaktiv (nur U1600 !).
– H? schiebt Nummer (0..31, -1:Pause) des aktuellen H-Programms auf den Stack. Diese Funktion darf nur innerhalb von H-Programmen ausgeführt werden.
– H?? schiebt auf den Stack: 1= Focus H-Programm, 0= Focus Kommandozeile. Damit lässt sich feststellen, ob ein Programm als H-Programm ausgeführt wird.

H-Programme auflisten

HLIST : alle H-Programme auflisten, entspricht: H! *
HLIST <aufzählung> : entspricht: H! <aufzählung>
HLIST* : alle nicht-leeren H-Programme auflisten (U1600: b.v.)

– Kopieren von H 7 in H 13:
  h- 7,H 13=$
  oder
  h7-,h13=$
– Kopiert alle H's nach Station B:
  fori 0..31, i,h- ., i,B:h .=$

H-Programme unterbrechen

HBREAK : Abbruch und 16s Pausieren der Hintergrundprogramme.
HBREAK+ : wie HBREAK, jedoch Dauer der Pause 32s
HBREAK++ : wie HBREAK, jedoch Dauer der Pause 60s
HBREAK- : Pausieren der H-Programme sofort beenden. (U1600: n.v.)

IF ELSE ENDIF

IF ELSE ENDIF (EIF) : Programm-Verzweigungen

Der IF-Befehl nimmt ein Element vom Stack und rundet es (5/4) auf die nächste Ganzzahl. Ist diese ungleich Null, so wird der Programmteil zwischen IF und ELSE / ENDIF abgearbeitet.
Ist sie gleich Null, gilt der Programmteil zwischen ELSE und ENDIF.

Aufruf	: **IF**	**ELSE**	**ENDIF**
Stack	: <bedingung> >>> –	– >>> –	– >>> –

– Pro Unterprogramm gibt es bis zu 4 Verschachtelungsebenen.
– Folgen nach ELSE oder ENDIF keine Befehle mehr, kann ELSE / ENDIF weggelassen werden.
– Soll der Programmteil zwischen IF und ELSE nur einmal beim Auftreten einer wahren Bedingung durchgeführt werden, benutzen Sie IFF (siehe IFF).
– IF mit Zeitvergleich siehe ZEITVERGLEICH.
– EIF ist die Kurzform für ENDIF

Beispiel für "Bedingung erfüllt"

1, IF, PRINT "Dieser Teil wird ausgeführt", ELSE , PRINT "dieser nicht"

Beispiel für "Bedingung NICHT erfüllt"

0, IF, PRINT 'Nein', ELSE , PRINT "JA", endif; PRINT "Ja/Nein-Fertig"

IFF ... ELSE ENDIF : Programm-Verzweigungsbefehle zur einmaligen Ausführung

IFF

Sollen im Gegensatz zu IF / ELSE/ENDIF die Programmteile zwischen IF ... ELSE bzw. ELSE ... ENDIF nur EINMAL beim Auftreten der entsprechenden Bedingung durchgeführt werden, wird der Befehl IFF verwendet.
Intern wird dazu pro Hintergrundprogramm ein permanentes Flag (IFF) sowie ein flüchtiges Flag (IFF+), das stets nach Power-On initialisiert wird, automatisch verwaltet. Bei der H-Programmierung werden beide Flags initialisiert. In einem H-Programm (mit seinen Unter-P-Programmen) kann also der IFF und der IFF+ Befehl je ein einziges Mal verwendet werden.

Aufruf	:	IFF		ELSE	ENDIF
Stack	:	<bedingung> >>> –		– >>> –	– >>> –
Ext	:		+ :	(IFF+ Flag wird nach Power-On stets neu initialisiert IFF mit Zeitvergleich: siehe ZEITVERGLEICH	

Beispiel

Wenn an Eingang 8 +24 V angelegt werden, soll einmalig Eges der Kanäle 1 ... 4, bei Rückkehr auf 0 V einmalig die Zeit ausgegeben werden.
H10 = 'IN– 8, IFF, Eges 1 . . 4, ELSE, zeit'

INDEX : Berechnung eines Index für die Datenliste

INDEX

Aufruf	:	INDEX *		
Stack	:	– >>>	'Index des zeitlich ersten Eintrags'	
Aufruf	:	INDEX **		
Stack	:	– >>>	'maximale mögliche Anzahl der Einträge'	
Aufruf	:	INDEX <abDatum/abZeit> [<abZeit>]		
Stack	:	– >>>	'Index für Suchzeit'	
Ext	:	+ :	Index für Suchzeit – 1 (vermeidet Überlappungen)	
		// :	statt „bis“ wird „von“ Zeit gesucht	

Beispiel

INDEX 17.03 12:15, eint##/ 1 . . 4 . *

– Für den dem INDEX-Befehl nachfolgenden Befehl (in der Zeile) kann die Gültigkeit der INDEX-Nummer garantiert werden, unabhängig von einer zwischenzeitlichen Intervallgrenze (Gültigkeit mindestens 0,3 s).
– Intervallgrenzen während eines EINT-Befehls bringen die Datenausgabe nicht durcheinander.

INDIR

INDIR : überprüft, ob eine Kennung im Verzeichnis (Dir) vorhanden ist

Aufruf	:	INDIR
Stack	:	<kennungnr> >>> (1: <kennungnr> in DIR / 0: nicht vorhanden
Aufruf	:	INDIR >kennungnr>
Stack	:	– >>> (1: <kennungnr> in DIR / 0: nicht vorhanden
Aufruf	:	INDIR *
Stack	:	– >>> (1: „aktuelle“ Kennung in DIR / 0: nicht vorhanden)

– Ext ‘–‘ ignoriert die Promt-Station wie bei ALL– (U1600: n. v.).

Beispiele

- Sobald Station G3 im ECS-LAN ist, erscheint für 10 s “G3 im Netz“ auf allen LCD-Displays:
  <A> H = ’G3:indir*, iff, all, meld “G3 im Netz“ 10’
- Der ALL-Schleifen-Befehl wird nur zur Demonstration „zu Fuß“ geschrieben:
  ’ALL, ... , NEXTA, ...’ entspricht:
  ’1, 255, fori, i, indir ., if, i, an: , ... , endif, nexti, zz: , ...’

INPUT

INPUT (IN) : Einlesen des Eingangs-Zustands

Aufruf	:	IN <aufzählung>
Stack	:	– >>> (1: 24 V liegen an, 0: 0 V liegen an)
Ext	:	+ – . # %

INTERVALL

INTERVALL (ITV) : Synchron-Intervall

Aufruf	:	INTERVALL [=<dauer>] <dauer>: 10 s ... 999 h
oder	:	INTERVALL [=<zahl>_ _<einheit>] <einheit>: s | m | h
Stack	:	– >>> <dauerInSekunden>
Ext	:	+ – . # | %

Beispiel

INTERVALL = 15 m oder INTERVALL = 35 s

– Bruchteile von Stunden oder Minuten müssen umgerechnet werden: 1 h 30 → 90 m oder 5400 s. Bei der Ausgabe wird stets versucht, die größte Zeiteinheit darzustellen.
– siehe auch INTERVALLQUELLE und SYNC

INTERVALLQUELLE (IQ) TARIFQUELLE (TQ)

INTERVALLQUELLE, TARIFQUELLE

INTERVALLQUELLE (IQ) : Quelle zur Erzeugung des Synchron-Intervalls

TARIFQUELLE (TQ) : Quelle zur Erzeugung des aktuellen Tarifs T1 oder T2

Aufruf : IntervallQuelle [=<quelle>]
Ausgabe : ja
Stack : – >>> <quellenzahl>
Ext : + – . # $ %

	<quelle> ==	<quellenzahl>	
INTERVALL	1 . . 12 (Eingang)	1 . . 12	(U1600: 1 ... 24)
	Z | Zeit	99	
	P | Programm	100	
TARIF	1 . . 12 (Eingang)	1 . . 12	(U1600: 1 ... 24)
	P | Programm	100	

Beispiele

Tarif von Eingang 7: TARIFquelle = 7
Intervall zeitabhängig: IQ = Z

KANAL LANGNAME

KANAL, LANGNAME

KANAL (KAN) : Der Name des Kanals (max. 8 Zeichen)
LANGNAME (LNAME) : Der Langname des Kanals (max. 20 Zeichen)

Aufruf : KANAL <aufzählung> [=<zeichenkette>]
Ausgabe : ja
Ablage : <name>
Stack : – >>> –
Ext : + – . # $ %

Zeichenvorrat

– Es können alle ASCII-Zeichen (außer Steuerzeichen) verwendet werden, mit Ausnahme der folgenden Zeichen: ' , ' ' , ' <leerzeichen>
– ' , ' und ' , ' werden in ' _ ' verwandelt, ein <leerzeichen> wird als String-Ende angesehen. Die Zuweisung KANAL <aufzählung> = " $ " weist nicht etwa das Dollarzeichen dem Kanalnamen zu, sondern den Zwischenablageninhalt.
– Kanalnamen können systemweit gesucht werden, wenn folgende Einschränkung beachtet wird: das 1. Zeichen muss ein Buchstabe sein.
– Damit die Kanalnamen für dBASE-Feldbezeichner benutzt werden können, darf außer Zahlen und Buchstaben nur ' _ ' verwendet werden. Das erste Zeichen muss ein Buchstabe sein.

KENN

KENN : Kennung als Nummer auf den Stack. (Kennung einstellen: SETKENN)

Aufruf : KENN [<kennungsNummer>]
ohne <kennungsNummer>: aktuelle Kennung
Stack : – >>> <kennungsNummer>
A: 1, A1: 2, ..., B: 11, C: 21, ... , Z4: 255
Ext : ! + . # %

– Kennung-zu-Kennungsnummer-Beziehung:
A : 1, A1 : 2, ... B : 11, C : 21, ... Z4 : 255
– Umwandlung einer Kennung → Kennungsnummer:
<kennung>:KENN#
– Umwandlung einer Kennungsnummer → Kennung:
KENN! <kennungNummer>

Beispiel

z4:kenn, ! kenn! 21 kenn. 21 kenn# 21
255 C: C 21

Einfluß der Extension '&' auf allgemeine Befehlsausgaben:

& : Ausgabe der Kennung am Zeilenanfang
[Eges& 1 → A: EGes ...]
&# : <kenn>:<w1>
&#. : <kenn>:<w1>
&. : <kenn>:<w1a>;<w1b>; ...
&& : Ausgabe der Kennung am Zeilenanfang und vor jedem Ausgabeblock.
&&## : <kenn>:<w1>;<kenn>:<w2>; ...
&&##. : <kenn>;<w1>;<kenn>;<w2>; ...
&&.. : <kenn>;<w1a>;<w1b>;
..;<kenn>;<w2a>;<w2b>; ..

KMODE

KMODE : Kanal-Modus

Aufruf : KMODE <aufzählung> [=<kanalmode>]
Ausgabe : ja
Stack : – >>> –
Ext : + – . # $ % ?

<kanalmode>		mögliche Kanäle		
		U1601	U1602	U1603
0 : OFF	AUS	1 ... 64	1 ... 64	1 ... 64
1 : ANA	Analog-E/A (ANA)	1 ... 14	—	1 ... 6+ 13+14
2 : P->E	PMOM=ANA → ENERGIE	1 ... 12	—	1 ... 6
3 : COUN	ANA-IMPULSE→ ENERGIE/PMOM	1 ... 12	—	1 ... 6
4 : LON	LON-Zähler → ENERGIE / PMOM	1 ... 64	1 ... 64	1 ... 64
5 : LonA	LON-Analog-E/A (LONANA)	1 ... 64	1 ... 64	1 ... 64

6 : L-PE	wie LONA plus LONANA → ENERGIE	1 ... 64	1 ... 64	1 ... 64
7 : LonI	LON-Binär-Inputs	1 ... 64	1 ... 64	1 ... 64
8 : LonR	LON-Relais	1 ... 64	1 ... 64	1 ... 64

– KMODE entspricht bei Analog-Kanälen dem Befehl ANAMODE
ANAMODE **: Mode aller Analog-Kanäle (U1601/2/3: 1..14)
KMODE **: Mode aller Kanäle (U1601: 1..64)
– LISTKMODE gibt eine Liste aller verfügbaren Zuordnungen aus.
– KMODE ist bei U1600/10/15 nicht verfügbar (bei U1615 ANAMODE verwenden), die Ext. '?' unterdrückt Fehlermeldung "Syntax-Fehler" oder "Funktion nicht verfügbar".

KOSTFAK1 KOSTFAK2 TFIX

KOSTFAK1, KOSTFAK2, TFIX

KOSTFAK1, KOSTFAK2 : Kostenfaktoren der Tarife 1 + 2
Aufruf : KOSTFAK1 [=<faktor>]
Stack : – >>> <faktor>
Ext : + – . # %

TFIX : Fixpunkt Kosten
Aufruf : TFIX [=<fix>]
Stack : – >>> <fix>
Ext : + – . #

LOESCHKANAL (ERAKAN) LOESCHLISTE (ERALIS)

LOESCHKANAL, LOESCHLISTE

LOESCHKANAL (ERAKAN) : Löschen aller Messdaten eines Kanals (außer den Werten in der Messdatenliste)
Aufruf : LOESCHKANAL =<kanalaufzählung>

LOESCHLISTE (ERALIS) : Löschen der Intervall-Messdatenliste ab Index.
Aufruf : LOESCHLISTE = <abindex> : Löschen einschließlich <abindex> ... Ende
LOESCHLISTE = * : gesamte Liste

LON-spezifische Befehle

LON-spezifische Befehle : (beginnen stets mit Lon...)

LON-MESSDATEN:

LonZW <k>	: Zählerwert des LON-Zählers [LonCR]
LonE <k>	: Energie des LON-Kanals, entspricht LonZW
LonP <k>	: Momentan-Leistung des LON-Kanals
LonANA <k>	: optionaler Analog-Wert des LON-Kanals

LON-PARAMETER

LonKANal <k> [=<w>]	: Unterkanal-Auswahl des LON-Kanals [LonCHANel]
LonFAKTOR <k> [=<w>]	: optionaler Bewertungs-Faktor [LonFACTOR]
LonOFFSET <k> [=<w>]	: optionaler Bewertungs-Offset
LonSTOP <k> [=<w>]	: Aktivität des LON-Kanals; <w>: 1=stop, 0=run
LonPOLLDELay [=<w>]	: Polling-Delay [ms]; <w> : 0...32767 (Default=0)
LonSTATTIMing [=<w>]	: Stations-Timingcode; <w> : 0...15 (Default=9)
LonSUBNODE [=<s_n>]	: Lesen und Schreiben der SUBNET- und NODE-Adresse in der Form <s_n> : SxxxNyyy xxx : SUBNET-Adresse (1...255) yyy : NODE-Adresse (1...127) Beispiel: LonSUBNODE = S22N003

LON-RELAIS und INPUTS

LonRel <k> [<bit>] [= <w>]	: Ansprache von LON-Relais
LonInp <k> [<bit>] [= <w>]	: Ansprache von LON-Inputs (binär)

LON-ALLGEMEIN (nur Abfrage)

LonVER	: Version des U1601 LON-EPROMs
LonUSERS	: Anzahl der aktiven LON-User
LonUSE <k>	: 1=LON-Kanal aktiv (run), 0=nicht aktiv
LonUSE* <k>	: 1=LON-Kanal aktiv (run oder stop), 0=nicht aktiv
LonTYPe <k>	: Gerätetyp des LON-Kanals
LonMAXKANal <k>	: Anzahl der verfügbaren Unterkanäle; wenn LonUSE==0, wird 16 geliefert. [LonMAXCHANnel]
LonSHOWZW <k>	: 1=LonZW sinnvoll+anzeigen, 0=LonZW nicht sinnvoll+anzeigen. [LonSHOWCR]
LonNEU <k>	: Neuinstallation des LON-Kanals
LonRESET = 0	: Komplette LON-Neuinstallation

Hinweise

- Alle LON-Befehle beginnen mit Lon..., gefolgt von max. 9 Buchstaben. Die GROSS-geschriebenen Buchstaben müssen verwendet werden, die kleinen Buchstaben AM ENDE des Befehls können weggelassen werden.
- Typisch englische Befehle werden in [eckigen Klammern] genannt, englische und deutsche Befehle können beliebig gemischt werden, unabhängig von der eingestellten Dialogsprache.
- Bei einem Kommunikationsfehler mit dem LON-Knoten werden die oben genannten LON-MESSDATEN nach dem 6. erfolglosen Kommunikationsversuch auf Null gesetzt.
- Bei der Ansprache von Relais oder Inputs kann bitweise (bei Angabe des <bit> Parameters) oder auf das ganze Wort zugegriffen werden.
  <bit> : Bitnummer mit 1 = LSBit, 32 = MSBit
  Beispiel: Relais 1 ... 4 werden mit einem Befehl gesetzt, später das 2. Relais ausgeschaltet (Lon-Relaismodul: Kanal 36).
  LonREL 36 = 0b 1111oder LonR 36 = 15
  LonREL 36 2 = 0

Syntax-Beschreibung einiger typischer Lon-Befehle

LonID : LON-NeuronID des LON-Knotens abfragen/eingeben

Aufruf : LONID <kanal> [=<lon_id>]
Ausgabe : ja
Ablage : <lon_id>
Stack : – >>> –
Ext. : + – # . % ?

– <lon_id> : 12-stelliger HEX-Zahlenstring
Beispiel : LONID 1 = 01002a201F00

LonSTOP : Aktivität des LON-Kanals (run oder stopped)

Aufruf : LONSTOP <kanal> [=<wert>]
Ausgabe : ja
Stack : – >>> <wert> <wert> : 0=run, 1=stopped
Ext. : + – # . % ?

LonZW : Zählerwert des LON-Zählers

Aufruf : LONZW <kanal>
Ausgabe : ja
Stack : – >>> <wert>
Ext. : + – # . % ?

MELD MELD2

MELD MELD2

MELD : Anzeige einer Meldung im LCD-Display für eine bestimmte Zeit.
Durch Angabe eines Parameters kann eine Zeitdauer angegeben werden, wie lange die Meldung nach Wegfall der Meldungsursache noch angezeigt werden soll; dadurch können auch kurzzeitige Störungsmeldungen sicher erfasst werden. Abbruch der Meldungsanzeige beim Drücken einer Panel-Taste.

Aufruf : MELD <zeichenkette> [<Dauer-in-Sek>] (max. 60 s, Default = 5 s)

Stack : – >>> –

Beispiel

MELD "! Motor defekt !" →„! Motor defekt !"
**********"
"

MELD2 : Anzeige einer ZWEIzeiligen Meldung im LCD-Display für eine bestimmte Zeit.
Abbruch der Meldungsanzeige, sobald eine Panel-Taste gedrückt wird.

Aufruf : MELD2 <zeichenkette> [<zeichenkette> <Dauer-in-Sek>]

Stack : – >>> –

Beispiel

2,1,MELD2 "%!. Zeile" "%!. Zeile" → „1. Zeile "
„2. Zeile "

– <Dauer-in-Sek> == 99: unbegrenzt warten.
– wird bei MELD2 ein leerer String "" angegeben, bleibt die entsprechende Display-Zeile unberührt (außer bei Extension *).
– Ext – kopiert Meldung in die Ablage.

MENUAPP, MENUAPPN

MENUAPP MENUAPPN : Menü-Anwendungen definieren

Mit MENUAPPN wird die Bezeichnung einer Menü-Anwendung festgelegt, mit MENUAPP wird ein P-Programm einer Menü-Anwendung zugeordnet. Menü-Applikationen werden von der Hintergrundprogramm-Task zwischen H-Programmen ausgeführt."

MENUAPP

Aufruf	:	MENUAPP <menuapp aufz> [= <p prog nummer>]
Ausgabe	:	ja
Stack	:	– >>> –

MENUAPPN

Aufruf	:	MENUAPPN <menuapp aufz> [= <app bezeichnung>]
Ausgabe	:	ja
Ablage	:	<app bezeichnung>
Stack	:	– >>> –

– Es stehen 3 mal 5 Anwendungen zur Verfügung.
Bereich von <menuapp aufz>:
Anwendungen (1) : 1 . . 5
Anwendungen (2) : 6 . . 10
Anwendungen (3) : 11 . . 15

– <p prog nummer> ist die Nummer des zugeordneten P-Programms (0 . . 31); soll keine Aktion ausgeführt werden, muss –1 zugewiesen werden.

– <app bezeichnung> kann max. 10 Zeichen lang sein, Leerzeichen, ',' und ';' dürfen nicht verwendet werden.

– U1600/10/15: MENUAPP / MENUAPPN sind hier nicht verfügbar. Das Hintergrundprogramm H 19 wird zur sogenannten Druckausgabe verwendet.

MENUEDIT

MENUEDIT : Editiermöglichkeit in Menü-Anwendungen (U1600: n.v.)

Aufruf :	MENUEDIT <mode> <n> = <titel> [<unterwert> [<oberwert>]]		
Ausgabe :	Nein	<mode>	: Variable A | a | B | b
Stack :	– >>> –	<n>	: Index der Var. A oder B
Ext. :	@	<titel>	: Titel-String (max.16 Z.)
		<unterwert>	: Zahlen-Unterwert
		<oberwert>	: Zahlen-Oberwert

– Ext. '@' ermöglicht die Ablehnung von Nullwerten.

– Das Editieren kann nur während offener Anwendungsmenüs ausgeführt werden, daher muss stets 100 zur P-Programmnummer für MenuApp addiert werden.

– Beispiel: In A 14 ist ein Wert für einen Grenzwertalarm 1, in A 15 der Wert für Alarm 2 enthalten. Mit F4 und F5 im 1. Anwendungsmenü kann nun A14 und A15 unter Beachtung von bestimmten Unter/Oberwerten editiert werden.

MenuApp 04 = 114
MenuApp 05 = 115
MenuAppN 04 = 'Alarm 1'
MenuAppN 05 = 'Alarm 2'
P 14 = 'MenuEdit a 14 = "Alarm 1 [1..40]" 1 40'
P 15 = 'MenuEdit a 15 = "Alarm 2 [1..50]" 1 50'

MERKMALE (FEATURES): Abfrage aller Merkmale

MERKMALE

Aufruf : MERKMAL <name> [= <wert> [<freigabe>]]
MERKMALE Abfrage aller gesetzten Merkmale (<wert> != 0)
MERKMALE * Abfrage aller verfügbaren Merkmale

Ausgabe : Ja
Ablage : <name> MERKMALE : <letzter_name>
Stack : – >>> <wert> MERKMALE : - >>> <summe_aller_werte>
Ext. : + – & $ # .

– Die Verfügbarkeit von Merkmalen ist vom Gerätetyp abhängig. Jedes Merkmal hat einen bestimmten <wert>ebereich. Ist ein Merkmal deaktiviert, so ist dessen <wert> stets gleich 0. Im Normalfall sind die Merkmale beim Erwerb des Gerätes entsprechend den Kundenanforderungen gesetzt. Dennoch können einige Merkmale kundenseitig modifziert (<wert> ungleich 0) werden, manche auch komplett eingestellt werden (<wert> : 0,1,2,..).
– Ist ein <freigabe>-Code erforderlich, so kann es bei mehrfacher Falscheingabe zu Zeitsperren kommen.

MONBEG : Variabler Monats- / Jahresanfang.

MONBEG

Jeder Monat kann an einem beliebigen Tag, jedes Jahr an einem beliebigen Monat anfangen. Setzen des variablen Modes, sobald eine Zuweisung erfolgt. Der Anfang eines Tages kann mit TAGBEG festgelegt werden.

Aufruf : MONBEG <MonatsAufzählung> [<Jahr>]
[=<StartTag>]
Stack : – >>> <StartTag>
<Jahr> : 90 ... 99, 0 ... 30
oder 1990 ... 2030
Ext : – . # &

– Die optionale Angabe von <bisJahr> erlaubt die Verarbeitung von mehreren Jahren gleichzeitig (U1600: n.v.).

MONBEG@ = 0 : zurück zum Normalzustand (Beginn stets am Ersten).
MONBEG@@ = 0 : zurück zum Normalzustand, Rücksetzen der Einträge.
MONBEG@ = 1 : variabler Mode mit den ehemaligen Einträgen.
MONBEG@ : – >>> <zustand> <zustand> : 0 = Normal, 1 = variabler Mode

Beispiele

(wird <Jahr> weggelassen, gilt das aktuelle Jahr)
MONBEG 1..12 = 17: Monate 1 ... 12 beginnen am 17. des Monats
MONBEG 0 = 10: das Jahr beginnt im Oktober (hier: 17.10.)
MONBEG 0 1995 = 2: 1995 beginnt im Februar

NF4I, NF4M, NF8I, NF8M, NFSTD : Ausgabe von REAL-Zahlen im 32 oder 64 Bit IEEE-HEX-Format

Umschaltung auf den gewählten Ausgabemodus erfolgt für den Rest der Zeile.

NF4I	:	Ausgabe von REAL-Zahlen (4-Byte FLOAT) im 32 Bit IEEE-HE X-Format INTEL-Byte Ordering, 8 HEX-Zeichen.
NF4M	:	Ausgabe von REAL-Zahlen (4-Byte FLOAT) im 32 Bit IEEE-HE X-Format MOTOROLA-Byte Ordering, 8 HEX-Zeichen.
NF8I	:	Ausgabe von REAL-Zahlen (8-Byte DOUBLE) im 64 Bit IEEE-H EX-Format INTEL-Byte Ordering, 16 HEX-Zeichen.
NF8M	:	Ausgabe von REAL-Zahlen (8-Byte DOUBLE) im 64 Bit IEEE-H EX-Format MOTOROLA-Byte Ordering, 16 HEX-Zeichen.
NFSTD	:	Auf Standard-Ausgabe zurückstellen.

Von der globalen Ausgabe-Änderung für den Rest der Zeile sind betroffen: REAL-Ausgaben mit !, <befehl>% "%w", Hauptwert-Ausgaben mit Ext '#'

– Ohne Umschaltung lassen sich die verschiedenen IEEE-HEX-Formatausgaben mittels freier Formatierung wie folgt erzeugen:

<befehl>% "%4_ _ _w"	:	INTEL FLOAT-HEX
<befehl>% "%8_ _ _w"	:	INTEL DOUBLE-HEX
<befehl>% "%4_ _w"	:	MOTOROLA FLOAT-HEX
<befehl>% "%8_ _w"	:	MOTOROLA DOUBLE-HEX

Eingabe von REAL-Zahlen im 32 oder 64 Bit IEEE-HEX-Format:

		Stack:
NF4I <float_hex_intel>	:	– >>> <real>
NF4M <float_hex_motorola>	:	– >>> <real>
NF8I <double_hex_intel>	:	– >>> <real>
NF8M <double_hex_motorola>	:	– >>> <real>

– Der NF.. Befehl dient zur Ein- und Ausgabe von Zahlen im 32- oder 64-Bit IEEE-Format (die Station arbeitet intern mit diesen Zahlenformaten). Dargestellt werden die 4 oder 8 Byte als 8- oder 16-stellige HEX-Zahlen (2 Zeichen 0 . . 9+a . . f pro Byte), d.h. 32-Bit float Zahlen belegen 8 Zeichen (4 Byte, nf4i oder nf4m) und 64-Bit double-Zahlen belegen 16 Zeichen (8 Byte, nf8i oder nf8m).

– Ausgaben mit NF. . sind deutlich schneller als Standard-Ausgaben mit mehreren Nachkommastellen.

– Das MOTOROLA-Byte-Ordering (nf4m oder nf8m) fängt links mit dem MSBit an, rechts steht das LSBit:

INTEL:	B7 . . B0	B15 . . B8	B23 . . B16	B31 . . B24
MOTOROLA:	B31 . . B24	B23 . . B16	B15 . . B8	B7 . . B0

Beispiel

nf4i,	12345678,	!	→	4E613C4B	(INTEL)
nf4i	4E613C4B,	!	→	12345678	
nf4m,	12345678,	!	→	4B3C614E	(MOTOROLA)
nf4m	4B3C614E,	!	→	12345678	

PASSWORT : Sichern des Zugriffs per RS-232 auf die Station / ECS-LAN

Ein Master-Nutzer (Nutzer 1) und vier weitere Nutzer (2 ... 5) können verschiedene Passwörter (Zahlen im Bereich 1 ... 999999999) erhalten. Der Master-Nutzer kann das Zugriffsrecht der einzelnen Nutzer individuell einstellen, die Nutzer können ihr Passwort - soweit das alte bekannt ist - verändern.
Das Zugriffsrecht, das beim Master-Nutzer eingetragen wird, gilt, wenn kein Nutzer eingeloggt (Nutzer 0) oder das Timeout abgelaufen ist. Der Master hat stets alle Rechte (= 5). Anfangs sind alle Passwörter = 0, nur der Master kann 0-Passwörter ändern. Zum Löschen aller Passwörter: Master-Passwort = 0.

Achtung!
Falsche Eingaben bewirken u.U. Zeitsperren!

<nutzer>	: 1=Master=Nutzer–1, 2 ... 5=Nutzer–2 ... 5, 0=aktueller Nutzer	
<passwort>	: 1 .. 999999999,	0:Löschen
<pw_alt>	: altes Passwort oder Master-Passwort als Berechtigung	
<pw_neu>	: neues Passwort (muss wiederholt werden)	
<rechte>	: Zugangsrechte, siehe unten	
<timeout>	: in Minuten	0=kein Timeout
<frei>	: 1=befreit	0=nicht befreit
<com_i>	: COM-Zugang:	1=COM–1, 2=COM–2, 0=aktueller Zugang

LOGIN : Einloggen unter einem Passwort
Aufruf : LOGIN <nutzer> <passwort>
Ausgabe : ja
Stack : – >>> –

LOGOUT : Ausloggen
Aufruf : LOGOUT
Ausgabe : nein
Stack : – >>> –

WHOAMI : Abfragen der Nutzer-Nummer und der Zugriffsrechte (Who am I)
Aufruf : WHOAMI
Ausgabe : ja
Stack : – >>> –

Passwort : Einstellen der Gerätepasswörter und Zugriffsberechtigungen

Aufruf : PASSWORT <nutzer> <pw_alt> = <pw_neu> <pw_neu> <rechte> <timeout>
PASSWORT <nutzer> <pw_alt> = <pw_neu> <pw_neu>
PASSWORT* <nutzer> <pw_alt> = <rechte> <timeout>

Stack : – >>> –

Aufruf : PASSWORT → Ausgabe des aktuellen Nutzers
PASSWORT <nutzer> → Ausgabe der Rechte von <nutzer>
PASSWORT * → Ausgabe der Rechte von <nutzer> 1 .. 5

Stack : – >>> <timeout> <rechte> <nutzer>

PWLRELEASE: Befreien bestimmter ECL COM-Zugänge (U1600: n.v.)

(PWLREL) vom Passwort-Schutz (PasswordLockRelease)

Aufruf : PWLRELEASE <com_i> <master_pw> = <frei>

Aufruf : PWLRELEASE <com_i>

Ausgabe : ja

Stack : – >>> <frei>

– Wird ' * ' zur Listenausgabe verwendet, beziehen sich die Stack-Werte auf das erste angezeigte Element. Eine Summenbildung ist hier nicht sinnvoll.

Mit Ext ' | ' pusht PASSWORT bei Ausgabe zusätzlich <com_i> auf den Stack. <com_i> ist nur bei aktuellem Nutzer definiert (1:COM–1, 2:COM–2, ansonsten wird Null gepusht. (U1600: n.v.)

PASSWORT |
Stack: – <com_i> <timeout> <rechte> <nutzer>

Hinweis

Systemweite Passwort-Verwaltung.
Der PASSWORT-Befehl arbeitet systemweit, d.h. von einer Station können alle Zugangsberechtigungen verwaltet werden. Die Zugangssperre jedoch wirkt sich nur auf den LOKALEN RS-232 Zugang aus. Wenn sich ein Nutzer mit LOGIN anmeldet, so sind damit seine Zugriffsrechte für die lokale Station und für alle anderen Stationen im ECL-LAN festgelegt (siehe unten). Beim Anmelden gelten die Passwörter, Rechte und Timeout-Zeiten der lokalen Station.
Bei der Vergabe von Passwörtern st also Vorsicht geboten, daß der Adresskontext auf die lokale Station (Spezialkennung AA:) verweist, wenn der PASSWORT-Befehl ausgeführt wird.
Ein Voranstellen der Spezialkennung AA:PASSWORT ... ist also ein wirksamer Schutz vor Seiteneffekten.

Freigeben einzelner ECL-Zugänge

– U1600/10/15 haben nur einen ECL-Zugang (COM-1). Bei Stationen mit mehreren ECL-Zugängen (COM-1, COM-2) werden zwar die selben Passwörter und Zugangsrechte / Timeoutzeiten für alle Zugänge verwendet, der Zustand ist jedoch für jeden Zugang vollständig entkoppelt. So kann auf COM-1 Nutzer-2 eingeloggt sein, auf COM-2 Nutzer-5. Wird also der aktuelle Nutzer abgefragt, ist die Angabe des Zugangs notwendig, ohne Angabe gilt der aktuelle Zugang (wie WHOAMI, jedoch mit Stack-Ausgabe).

PASSWORT 0 <com_i>	→	Ausgabe des aktuellen Nutzer auf <com_i>
PASSWORT 0 2	→	Ausgabe des aktuellen Nutzer auf COM–2
PASSWORT 0 *	→	Ausgabe des aktuellen Nutzer auf COM–1+2

Bedienpanel-Passwörter

– Mit PASSWORT können auch die Panel-Passwörter verändert werden. Hierzu werden die folgenden <nutzer>-Nummern verwendet. (U1600: n.v.):)

101 : Nutzer–1 (Master)
102 .. 105 : Nutzer–2 ... Nutzer–5

Zugriffsrechte und Timeout-Zeiten werden hier nicht verwendet. Wie auch bei den ECL_Passwörtern gilt: Der Master (Nutzer–1) kann alle Panel-Passwörter ändern, die anderen Nutzer nur ihr eigenes Panel-Passwort. Gelöscht wird ein Panel-Passwort durch die Angabe 0 oder 111111. Wird das Master-Panel-Passwort gelöscht, wird zwar der Zugangsschutz aufgehoben, die anderen Panel-Passwörter bleiben jedoch erhalten.

Hinweis
Panel-Passwörter sind nicht identisch mit ECL-Passwörtern. U1600/10/15-Stationen kennen nur ein Panel-Passwort, dieses wird mit der <nutzer>-Nummer 101 oder 99 angesprochen.

Beispiel (Reihenfolge beachten!)

Master gibt ein:

PASSWORT 1 0=123 123 0 5 : Master-Pw = 123,
Timeout = 5 m, 0-Nutzer-Rechte = 0
PASSWORT 2 123=222 222 3 10: Nutzer-2-Pw = 222,
Timeout = 10 m, Rechte = 3
PASSWORT* 2 123=2 5 : Änderung der Rechte = 2
und Timeout = 5 m

Nutzer ändert Passwort:

PASSWORT 2 222=2121 2121 : Änderung des Passworts

Nutzer loggt sich ein:

LOGIN 2 2121

Master löscht alle Passwörter: PASSWORT 1 123=0 0

Zugriffsrecht	Lokal		ECS-LAN		Notation
	Lesen	Schreiben	Lesen	Schreiben	
0	–	–	–	–	[– – L: – –]
1	ja	–	–	–	[r– L: – –]
2	ja	–	ja	–	[r– L: r –]
3	ja	ja	–	–	[r w L: – –]
4	ja	ja	ja	–	[r w L: r –]
5	ja	ja	ja	ja	[r w L: r w]

PAUSE

PAUSE (PP) : Pause in Sekunden

Der Programmablauf wird für n Sekunden unterbrochen, auch Bruchteile von Sekunden können angegeben werden. Die effektive Wartezeit ist stets ein Vielfaches von 100 ms.

Aufruf	: PAUSE	oder PAUSE <wert>
Stack	: <wert> >>> –	– >>> –

Hinweis
Maximale Pausendauer = 20 s Zahlen für n > 20 werden als Angabe in Millisekunden angesehen.
Beispiel: 'Pause 2.2' entspricht 'Pause 2200', der Programmablauf wird für 2,2 Sekunden unterbrochen.

PEGEL

PEGEL : Einstellen der Empfindlichkeit der Zählereingänge.

Schaltschwelle: 0=10%, 1=25%(Default), 2=50%, 3=70% vom Fullrange..

Aufruf	: PEGEL [=<wert>]		
Ausgabe	: ja		
Stack	:	– >>> <wert>	(nur beim Lesen)
Ext	:	+ – %	<wert> : 0 (Lo) ... 3 (Hi), Vorgabe: 1

– Die Einstellung gilt für alle Zähleingänge zusammen.
– Die Pegelerkennung arbeitet mit einer Genauigkeit von 5% und einer Hysterese von 1% von Fullrange.
– Für U1600-Stationen gelten andere Zuordnungen:
Der maximale H-Eingangspegel (ohne Hysterese) kann ungefähr im Bereich von 3 Volt (Lo) und 5,5 Volt (Hi) eingestellt werden (typische Angaben).

PFAKTOR

PFAKTOR : Faktor zur Berechnung der Leistung aus der Energie pro Zeitspanne

Mit diesem Faktor kann der Zeitbezug der Leistungsberechnung angepaßt werden.

– Normalerweise wird der Stundenbezug benutzt (kWh zu kW):
Pfaktor = 3600

– Soll ein Sekundenbezug verwendet werden (Ws zu W) : Pfaktor = 1
Formel zur Berechnung der Leistung P aus Energie E und Zeitspanne dt: P = E * Pfaktor/dt

Aufruf	: PFAKTOR <aufzählung> [=<wert>]	
Stack	: – >>> <wert>	(beim Lesen)
Ext	: + – # . %	

POWERFAIL (PWR) : Liste der Hilfsenergie-Unterbrechungen (max. 32 Einträge) **POWERFAIL**

Aufruf : POWERFAIL <aufzählung> [=0]
Ausgabe : ja
Stack : – >>> –
Ext : + – # . / * %
| @ (siehe unten)

– Liste aller Unterbrechungen : PWR *
(beginnend mit dem zuerst aufgetretenen Power Fail)
– umgekehrte Ausgabereihenfolge mit Ext ' * ' : PWR* *
– Löschen z.B. ab Index 7 : PWR 7=0
– VON und BIS der zuletzt gelisteten Unterbrechung werden stets (auch ohne Extension " / ") gesetzt.
– Dauer der zuletzt gelisteten Unterbrechung [s] : PWR–,DAU, !
– Formatierung : %w entspricht der jeweiligen Unterbrechungsdauer [s], %e der Einheit " s " (U1600: n.v.)

POWERFAIL I : wie POWERFAIL, Gesamtausfalldauer → Stack U1600: n.v.) **POWERFAIL I**

Aufruf : POWERFAIL I <aufzählung> [=0]
Ausgabe : ja
Stack : – >>> <summe_der_ausfallzeiten>
Ext : + – # . / * %

POWERFAIL@ : gibt Intervall <PowerON> . . <jetzt> an [siehe auch POWERON] **POWERFAIL@ (PWR@)**

Aufruf : POWERFAIL@
Ausgabe : ja
Stack : – >>> –
Ext : + – # . / | %

– Ermittlung der Einschaltdauer [s] seit PowerOn : PWR@– ,DAU, !

POWERON (PWR0) :Betriebsdauer seit dem letzten PowerOn oder Reset **POWERON**

Aufruf : POWERON
Stack : – >>> <dauer_in_Sekunden>
Ext : + – # . / $ %

– VON und BIS werden stets (auch ohne Ext /) gesetzt.

PRINT

! (PRINT) : Ausgabe-Befehl

!	:	Stack-Ausgabefunktion nimmt eine Zahl vom Stack und gibt diese aus
! ...	:	Ausgabefunktion ! [<par> [__<par> [__<par>]]] Beispiel: fix 3,5,! "Wert = " . " Kg" → Wert = 5.000 Kg
!$	:	Ausgabe der Ablage
!?	:	Stringvergleich (Argument ↔ Ablage) Stack hier: – >>> {1 | 0}: Gleichheit = 1, Verschieden = 0
!_	:	Ausgabe einer Linie von 78 Unterstrichen '_'
!!	:	Ausgabefunktion „Eine Leerzeile“

– die auszugebenden Zeichenketten können Formatanweisungen ähnlich der printf() Funktion in der Sprache 'C' enthalten, siehe PRINTFORMAT.

– Es gilt: Zeichenketten <par1> ... <par3> werden entsprechend interpretiert und nahtlos aneinander gefügt. Zuweisungs-Zeichenketten <zpar1> ... <zpar3> werden unverändert übernommen und ebenfalls nahtlos aneinander gefügt.
Beispiel:
! \065 \066 \067=\068 "F e r t i g" → ABC\068F e r t i g

– weitere Beispiele siehe PRINT-Beispiele.

– mit Ext * wird die gesamte Ausgabe zusätzlich in die Ablage (max. 128 Zeichen) kopiert.

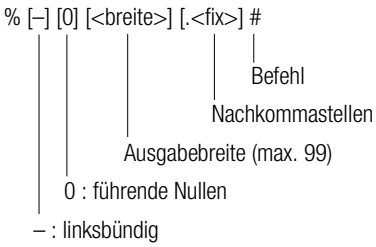
– Ext – [Ausgabe unterdrücken] leitet gesamte Ausgabe in die Ablage um.

– Ext ? vergleicht „case-INsensitiv“, Ext ?? „case-Sensitiv“.

PRINT-Formatierung

PRINT-Formatierung

Formatierung der Ausgabe:(ähnlich Formatbefehlen bei printf() in 'C'



Befehl	:	**Ausgabe**
%!	:	Zahl (vom Stack holen)
%x %X	:	Zahl in HEX (vom Stack holen)
%$ %s	:	Ablage
%<b>c<Z>	:	<b>-mal Zeichen <Z>
%<b>C	:	<b>-mal Zeichen (vom Stack)
%%	:	%

Weitere Optionen siehe PRINTMODI; Beispiele siehe PRINT-Beispiele.
Einige Befehle mit Ext % aufgerufen, formatieren die Ausgabe entsprechend des Format-Strings (1. Parameter).

Dann bedeutet:

%g	**%G**	: Geräte-Kennung, %g: Buchstaben-Kennung 'A1', %G: Kennungszahl
%f		: Funktions-Benennung (bei EGes also 'EGes')
%k		: Kanal-Nummer
%v	**%V**	: Kanal-Nummer/Code, fest formatiert: (00), 01 ... 24, V1 ... V8
%i		: Index (numerisch), bei ETag–3 also 3
%w		: Hauptwert (numerisch) des Befehls (bei EGes also die Energie)
%e		: Einheiten-String des Befehls
%n		: Namen-String des Befehls (Kanalname)

PRINT-Modifikationen

PRINT-Modifikationen

Stack-Ausgaben / -Manipulationen bei der Ausgabe-Formatierung:

%!		: n >>> –; drucke n
%#		: n >>> n; drucke n
%<	**%n<**	: DROP [<n = breite>]
%>	**%n>**	: DUP [<n = breite>]
%n^		: PICK
%~		: SWAP
%a0	**%a63**	: Inhalt (A n) : möglich für Variablen A(%a00) und B(%b00)
%&ann		: A nn push+drucken
%&&ann		: A nn nur push
%ai		: Inhalt (I) : möglich für I(%ai), J(%aj), K(%ak)

String-Ausgaben:

%$	**%s**	: Ausgabe der Zwischenablage
%p00	**%p19**	: Inhalt (P i)
%h00	**%h19**	: Inhalt (H i)

– %@p %@h %@s %@$: String wird als Formatstring verwendet (1 Verschachtelung)

– %$ mit Fixpunktangabe (z.B.: %.3$) : n Zeichen vom Stringanfang weglassen.

Kennungsangabe bei %p %h %a b% :

```
%p<kenn>:<num>     ! %pc2:17  (Inhalt von C2:P 17) oder
%p:<kenn>:<num>    ! %p:d:5   (Inhalt von D:P 5);
```

die 2. Möglichkeit ist bei folgender Konstellation wichtig:

```
! "%ai3:15"          → <inhalt von i>3:15
! "%a:i3:15"         → <inhalt von I3:A15>
```

Zeitausgaben

%z . .	:	Zeitzahl vom Stack, Stack bleibt (n >>> n)
%_z . .	:	letzter Zeitpunkt innerhalb der Formatierung (U1600: n.v.)
%/z . .	:	VON Zeitpunkt
%//z . .	:	BIS Zeitpunkt
%///z . .	:	Uhrzeit
%& z	:	push+drucken
%&& . . z	:	nur push
%zz %ZZ	:	Zeit$ / Zeit-Datum$
%zh	:	Stunde (0 . . 23)
%zm	:	Minute (0 . . 59)
%zs	:	Sekunden (0 . . 59)
%dd %DD	:	Datum$ / Datum-Zeit$
%dt %dT	:	Tag (Zahl / String)
%dw	:	Wochentag (1:Mo: . . . 7:So)
%dW	:	Wochentag (String)
%dm %dM	:	Monat (Zahl / String)
%dj	:	Jahr (90 . . 99, 0 . . 78)
%dJ	:	Jahr (1990 . . 2078)

– %DM, %DW erzwingen großgeschriebenen Namen. %dM, %%dW belassen Schreibweise.

– %:ZZ (':'bei Strings) modifiziert bei Zeitzahl = 0 die Ausgabe: (U1600: b.v.)

– 0,! %ZZ → "00:00:00 01.01.90"

– 0,! %:ZZ → "--:--:-- --.--.--"

– nur U1600: %p beläßt Inhalt von p, %P wandelt Inhalt um (siehe STRINGS)

– nur U1600: %$ beläßt den Ablage-String, %s wandelt \-Codes um (siehe STRINGS)

PRINT-Beispiele

- Ausgabe von Stackdaten
  12, 34, ! "n1 = %04!, n2 = %.3!"→ n1 = 0034, n2 = 12.000
  1,2,3,4, ! "%3^%07.3! %>%!,%!"→ 002.000 4,4
- Formatierung eines Ausgabe-Befehls (Kanal V1 = Heizraum)
  Eges% "%g:%f von %4n = %w %e" V1 →
  A:EGes von Raum = 1234.12 kWh
  Eges% "%G:%f von %–4n = %.0w %e" V1→
  1:EGes von Heiz = 1234 kWh
- Zeit-Ausgabe
  ! "Heute ist %///&dT, der %dt. %dM %dJ" →
  Heute ist Montag, der 22. März 1999
- HEX-Ausgabe
  %x : 43981,! 'In Hex: %05x' → In Hex: 0abcd
  %X : 1997,! 'In HEX: %X' → In HEX: 7CD

P0 . . P31 : Programme P0 ... P31 ausführen / programmieren
Q0 . . Q31 : Programme Q0 ... Q31 ausführen / programmieren (U1600: n. v.)
Aufruf : P <aufzählung> [=<zeichenkette>] max. Zeilenlänge: 128
Ext : + – . # $! ? @ Bei Ausgabe stets keine Ausführung

– P ohne <index> == P0, P1 . . P19 == P 1 . . P 19, P5! == P! 5
– Bei beliebiger Extension erfolgt keine Programm-Ausführung.
– Maximale Verschachtelungstiefe = 3 (P[P[P]])
– Beim Lesen: <programm → Zwischenablage
(außer bei Extension "– –" oder Ausführung).
– P@ oder DO $ führen den Inhalt der Ablage als Programm aus.
– P? schiebt Nummer (0 . . 31, –1:keins) des aktuellen P-Programms auf den Stack.
– Der Adress-Kontext eines P-Programmes bestimmt den Ursprung des Programmtextes, NICHT jedoch den Adress-Kontext, unter dem die einzelnen Befehle des Programms ausgeführt werden. Hier gilt der aktuelle Zeilen-Kontext, dieser kann also durchaus von einem P-Programm verändert werden.
Ein Beispiel mit ALL finden Sie bei KENNUNG.
– I, J, K Variablen gelten lokal, d.h. eine Programm-Ebene hat keinen Zugang auf die I, J, K Variablen der darunter liegenden Ebenen.
– Nicht genutzte P-Programme können als String-Speicher verwendet werden.
– Q-Programme Q 0 .. Q 31 sind ab Firmware-Stand April 2001 verfügbar.

• Aufruf eines P-Programmes mit frei definierbarem Namen: siehe REM
• Auflisten aller P's: P! * oder PLIST
• Kopieren P7 in P13: P– 7, P 13=$ oder P7–,P13=$
• Kopiert alle P's nach Station B: fori 0 . . 31, i,P– ., i,B:P .=$

PLIST oder PLIST <aufzählung> : Listet P-Programme (entspricht P! *)
PLIST* : Listet alle nicht-leeren P-Programme (U1600: n.v.)

DO : Ausführen eines Programmstrings (U1600: n.v.)
Aufruf : DO <programm>
Ausgabe : nein
Stack : – >>> –

– Adress-Angabe bei DO hat keine Auswirkung.
– I, J, K Variablen gelten lokal, d.h. eine Programm-Ebene hat keinen Zugang auf die I, J, K Variablen der darunter liegenden Ebenen.

DOWHILE (DOWH)	: Ausführen eines Programmstrings, solange eine Bedingung erfüllt ist (U1600: n.v.)
Aufruf	: DO <programm>
Ausgabe	: nein
Stack	: <bedingung> >>> –

– Eine Zahl wird vom Stack genommen und auf die nächste Ganzzahl gerundet (5 / 4). Wenn diese Zahl ungleich Null ist, wird <programm> ausgeführt. Nach der Ausführung wird wieder eine Zahl vom Stack genommen und der Vorgang wiederholt, bis diese Zahl gleich Null ist.
– Adress-Angabe bei DOWHILE hat keine Auswirkung.
– I, J, K Variablen gelten lokal, d.h. eine Programm-Ebene hat keinen Zugang auf die I, J, K Variablen der darunter liegenden Ebenen.

Beispiel
Runterzählen von 10 auf 1:
10, dup, dowhile '! "noch ,1,–,dup'

RELAIS, SORELAIS, RELAISMODE

RELAIS SORELAIS RELAISMODE

RELAIS (REL)	: Relais-Ausgänge schalten 4 Ausgänge (Wechsler) 1 ... 4 stehen zur Verfügung. '1': aktives Relais, '0': nicht aktives Relais
Aufruf	: REL <aufzählung> [={1 | 0}]
Ausgabe	: ja
Stack	: – >>> {1 | 0} (beim Lesen)
Ext	: + – . # * $ % @

SORELAIS (SOREL)	: SO-Relais (Halbleiter-Relais) Ausgänge schalten (U1600: b.v.)
Aufruf	: SOREL <aufzählung> [={1 | 0}]
Ausgabe	: ja
Stack	: – >>> {1 | 0} (beim Lesen)
Ext	: + – . # * $ % @

RELAISMODE (RELM)	: Betriebsmode der Relais-Ausgänge festlegen: 0: Relais stets AUSgeschaltet 1: Relais stets EINgeschaltet 2: Relais per Programm steuerbar (Default)
Aufruf	: RELM <aufzählung> [=<mode>]
Ausgabe	: ja
Stack	: – >>> <mode> (beim Lesen)
Ext	: + – . # * %

– Ext * unterdrückt stets den optionalen Relaisnamen (siehe RELAISNAME).
– Ext @ bei REL (REL@ <aufzählung>) gibt den tatsächlichen Relaiszustand an, berücksichtigt also Überreitungen mit RELAISMODE.

– Statt <aufzählung> kann auch der Name eines Relais (siehe RELAIS-NAME, FINDER) angegeben werden. SORELAIS verhält sich dann wie RELAIS.
– Die Stationen sind wie folgt bestückt:
  U1600 : 4 Relais-Ausgänge (Wechsler) REL 1 . . 4
  U1601 : 2 Relais-Ausgänge (Wechsler) REL 1 . . 2,
  4 Halbleiter-Relais-Ausgänge (Arbeitskontakt) REL 3 . . 6
  (S1 <– –> REL 3, S2 <– –> REL 4,
  S3 <– –> REL 5, S4 <– –> REL 6)
– U1615 : max. 7 Relais-Ausgänge (Arbeitekontakt) REL 1 . . 7
  (siehe ANARELMAP)
– Mit SOREL lassen sich die Halbleiter-Relais-Ausgänge (sofern vorhanden) direkt steuern. Bei U1601 gilt: SOREL 1 entspricht REL 3, bei Geräten ohne dedizierte SO-Relais gilt: SOREL 1 entspricht REL 1.
  Mode- und Namen-Einstellung müssen mit RELAISMODE / RELAISNAME erfolgen, die Relaisnummer ist dann entsprechend zu korrigieren.
– Im STATUS erscheint:
  'p' : per Programm AUS 'P' : per Programm EIN
  '_' : stets AUS (Mode 0) '+' : stets EIN (Mode 1)
  '–' : AUS '*' '*' : EIN

S0Pxxxx Befehle zur Pulsausgabe auf S0-Relais:(ab 03/2002) **S0Pxxxx**

```
                // <s0rel> : 1..4
S0PCH <s0rel> [=<chan>]// Eges-Kanal als Basis (0=aus, 1..64)
S0PDElta <s0rel> [=<delta>]// deltaE pro Puls (0=aus) entsprechend
                // der Einheit von <chan>
S0PMS <s0rel> [=<pulsdauer_ms>]// Pulsdauer in Millisekunden (0..1000)
                // Werte < 20ms → 20ms
                // minimale Periodendauer = 2 * S0PMS
S0PULSe <s0rel> [<periodendauer_ms>] = <pulsanzahl>
                // Ausgabe von <pulsanzahl> Pulsen mit
                // entsprechender Periodendauer unab-
                // hängig von S0PCH und/oder S0PDElta
                // Werten. S0PMS wird beachtet.
                // <periodendauer_ms> : nicht angegeben
                // oder < 20ms → 2 * S0PMS
```

– Kleine Buchstaben im Befehlsnamen sind optional, S0PULS und S0PULSE sind also gleichwertige Namen für den selben Befehl.
– Intern wird ein Eges-Vergleichsregister geführt, das mit dem zu Testzwecken implementierten Befehl S0PEGES <s0rel> gelesen werden kann. Wird eine Energiedifferenz von umgerechnet mehr als 250 Pulsen erzielt, wird diese Differenz genullt (S0PEGES=EGES) und keine Pulse ausgegeben. Ansonsten wird die Anzahl der auszugebenden Pulse anhand der Energiedifferenz berechnet und das Eges-Vergleichsregister S0PEGES entsprechend angepasst.
  Pulsanzahl = Int(EGES-S0PEGES / S0PDElta)
  Die Vergleichsbildung findet mehrfach pro Sekunden statt.
– **Bitte beachten:**
  S0PDELTA ist reziprok zu ZKONST (Annahme: EEinheit = kWh, URAT = IRAT = 1)
  ZKONST: <zkonst> Impulse pro kWh
  S0PDELTA: <s0pdelta> kWh pro Impuls

– S0PDElta kann auch negativ sein und damit auf negative Differenzen reagieren, um so z.B. nicht den Bezug sondern die Abgabe zu zählen.
– Leistungsinformation in der Pulsausgabe:
  Durch ein spezielles Verfahren kann aus der Pulsausgabe wieder auf die Leistung geschlossen werden. Wenn mehr als ein Puls auszugeben ist, werden also keine "Pulsetrains" ausgegeben, sondern der Pulsabstand wird bei der Differenzbildung aus der aktuellen PMOM berechnet (bei Kanälen mit PMOM = 0 werden dann dennoch Pulsetrains erzeugt). Werden plötzliche Delta-Peaks erfasst, so wird der aktuelle Pulsabstand entsprechend gekürzt, damit schnell auf die neue PMOM-Situation eingegangen werden kann.
  Wenn nach einer Vergleichsbildung ein oder mehrere Pulse auszugeben sind, und noch Pulse aus der letzten Vergleichsbildung ausstehen (insgesamt n Pulse), so werden diese n Pulse genau dann gleichmässig auf 10 s aufgeteilt, wenn gilt: n * Periodendauer > 10 s (Periodendauer berechnet aus höchster PMOM der angesammelten Pulse). Die Genauigkeit der aus dem Pulsabstand zu gewinnenden Leistung hängt von mehreren Faktoren ab: Frequenz der Pulsausgabe, Kanalmode (Zählimpulse, P→E Wandlung), Intervall einer u.U. verketteten DVIRT-Bildung (Zykluszeit der H-Programme), usw. Trotz einer möglichst realitätsnahen Leistungsab- bildung sollte der Leistungswert nur als Richtwert gesehen werden, der Energiewert jedoch ist (nahezu) fehlerfrei (64-Bit double Rechengenauigkeit, Zeitverzögerungen und Impulsverluste durch Hilfsenergieunterbrechungen möglich.
– Default-Werte: S0PCH 1..4 = 0, S0PDE 1..4=0, S0PMS 1..4=0, S0PEGES 1..4=0
– **Beispiel**: Kanal 1 zählt Impulse mit 1 Impuls pro kWh. EGES von Kanal 1 dient als Basis zur Impulsausgabe auf dem S0-Relais 1 (1 kWh pro Impuls). Dieses wird über den 24 V-Ausgang dem Kanal 2 zugeführt. Registerinhalte der hier nicht genannten Register entsprechen den Default-Werten.
  KMODE 1+2 = 3 P→E, ZKONST 1+2 = 1, PULS 1+2 = 10 ms
  S0PCH 1 = 1, S0PDELTA = 1 kWh, S0PMS = 20 ms
  Startzustand (nur zu Testzwecken!) einnehmen:
  StartStop 1+2 = 0, EGES 1+2 = 0, S0PEGES 1 = 0, StartStop 1+2 = 1
  Alle Zählimpulse auf Kanal 1 werden nun über das S0-Relais 1 ausgegeben und von Kanal 2 gezählt (max. Frequenz 25 Hz). Die von Kanal 2 gemessene PMOM entspricht im oben genannten Rahmen der PMOM von Kanal 1.

RELAISNAME

RELAISNAME (RELN) : Name eines Relais

Jedem Relais kann ein Name zur besseren Identifikation zugeordnet werden.

Aufruf	:	RELN <aufzählung> [=<name>]
Stack	:	– >>> –
Ext	:	+ – . # %

– Länge von <name> = 8 Zeichen.
– Der Name ist besonders nützlich bei Verwendung der Suchfunktion (siehe FINDER)
– Löschen eines nicht verwendeten Namens: RELN 2 = ““

Beispiel

REM “Mittelwert Programm“
@ Oberwert-Ermittlung
REM eins zwei drei (je <par> eine Zeichenkette)

– Auskommentieren einer Programmzeile
  Beginnt eine Zeile mit '#', so wird diese Zeile komplett ignoriert. H-Programme können damit recht einfach unwirksam geschaltet werden, ohne den Inhalt löschen zu müssen.
– Aufruf eines P-Programmes mit freidefinierbarem Namen
  Ist der 1. Befehl eines P-Programms eine spezielle Bemerkung "@@ prog_name", so kann dieses Programm einfach durch Eingabe von prog_name statt P n aufgerufen werden, egal wo es im ECS-LAN abgelegt ist.

Beispiel

P 10 = '@@ Hallo, ! "Hallo, wie geht es Ihnen?"'
Aufruf: hallo Ausgabe: Hallo, wie geht es Ihnen?

LPSEARCH : Limited P Search, Einschränkung der systemweiten Programmsuche (ab 12/2001)

Aufruf:

LPSEARCH = 0 Sucht zuerst auf Station mit aktueller Adresse, danach auf allen anderen Stationen im LAN (beginnend von A:..Z4:). DEFAULT.

LPSEARCH = 1 Sucht zuerst auf Station mit aktueller Adresse, danach auf der Prompt-Station (ZZ:), schließlich auf der RS232-Station (AA:).

Ausgabe : ja
Stack : – >>> <lpsearch>
In großen Netzen ist eine Einschränkung der systemweiten Programmsuche mit LPSEARCH = 1 sehr nützlich, denn falsch geschriebene Befehle führen hier zu langen Wartezeiten.

SET... : Einstellen der Schnittstellen **SET...**

Einstellen der COM-, LAN-, LON- Schnittstellen:

SETCOM1 [= <param> [<dly]] // serielle COM-1 Schnittstelle
SETCOM2 [= <param> [<dly]] // serielle COM-2 Schnittstelle
SETLANL [= <param> [<dly]] // ECS-LAN LINKS
SETLANR [= <param> [<dly]] // ECS-LAN RECHTS
SETLON [= <param>] // LON Netzwerk (soweit verfügbar)
SETCOMS // Ausgabe aller Schnittstellen-Einstellungen

Ausgabe : ja
Ablage : Parameter-String
Stack : – >>> –
Ext : & + - # % $?

– die Einstellmöglichkeit per ECL ist erst mit Firmware-Versionen ab dem 16.05.1999 möglich.
– Die Einstellungen müssen mit größter Vorsicht vorgenommen werden, da ein Geräte-Verbund bei falscher Einstellung u.U. irreversibel gestört werden kann. Diese Störungen können u.U. nur noch lokal per Bedienpanel beseitigt werden.
– Damit ECS-LAN und COM Einstellungen ausgeführt werden können, ohne den aktuellen Befehl zu behindern, kann ein Delay <dly> in Sekunden angegeben werden (0.3s .. 25.5s, Werte kleiner 0.3 --> 0.3s). SETLON arbeitet stets ohne Verzögerung (auch wenn <dly> angegeben wird).
– <param> ist der Parameter-String, der keine Leerzeichen enthalten darf. Groß-/Kleinschreibung ist unwichtig. Die komplette Parametrierung oder nur ein Teil kann angegeben werden, die Angaben in Klammern sind alternativ gültig.

SETCOM1, SETCOM2: mode/baudrate/parity/handshake
mode : OFF, ECL, ECL+HP, DCF77(DCF)
baudrate : 110, 150, 300, 1200, 2400, 4800(4k), 9600(9k), 19200(19k), 38400(38k), 57600(57k), 76800(76k), 115200(115k)
parity : P-, EVEN(PE), ODD(PO)
handshake : XON(XOFF), RTS(CTS)

SETLANL, SETLANR: mode/baudrate
mode : 2D(2W), 2D+(2W+), 4D(4W)
baudrate : 15600(15K6), 31200(31K2),62500(62K5), 125000(125K), 375000(375K)

SETLON: mode
mode : OPEN(O)(-), RA50(RT50)(50), RA100(RT100)(100)

Beispiele

- Setzen von COM1 auf Standard-Vorgaben (ECL/9600/P-/XON):
  SETCOM1 = DEFAULT
- Ändern der Baudrate auf 115200 Baud:
  SETCOM1 = 115k
- Ändern der Parity und des Handshakes (PE/RTS)
  SETCOM1 = PE/RTS
- ECS-LAN: Ändern der Baudrate einer Line-to-Line Verbindung (hier seien die Stationen D:(LAN/R) mit E:(LAN/L) verbunden (2-Draht oder 4-Draht). Die Verbindung arbeitet einwandfrei mit 62k5 Baud, die Baudrate soll nun auf 125k Baud erhöht werden. Sie haben Zugriff auf beide Stationen über das ECS-LAN (via D:LAN/L); damit die Verbindung zu E: nicht abreißt, wird die Einstellung um 2 Sekunden verzögert.
  D:SETLANR = 125k 2, E:SETLANL = 125k 2
  Nach Ausführen dieser Zeile werden die Einstellungen nach 2 s gültig, die Befehle konnten in dieser Zeit garantiert beide Stationen erreichen und bestätigt werden. Testen Sie nun die Netz-Performance mit SYSTEST.

setKENNUNG (setKENN) : Einstellen der Kennung

SETKENN

Aufruf : setKENN = {A, A1, . . A9, . . Z, Z1, . . Z4}:

Beispiel:
setKENN = U1:

SOWI : Unterstützung zur Umschaltung auf Sommerzeit (Sommer ↔ Winter)

SOWI

Darf nur in einem einzigen H-Programm verwendet werden.
Eine Umstellung kann nur dann erfolgen, wenn die Station zum Zeitpunkt der Zeitumstellung eingeschaltet ist.
Wenn keine Parameter angegeben werden, wird Monat März und Oktober zur Umschaltung verwendet, jeweils der letzte Sonntag um 2h/3h.
Ein internes Flag verhindert mehrfache Umstellung.

Aufruf : SOWI [<WiSo_monat> [<SoWi_monat>]]
Stack : – >>> <offset> <tu_es>
<offset> : 0, 3600, –3600
<tu_es> : 1 = Wechsel, 0 = nichts

Beispiel
H31 = 'SOWI, IF, ZEIT–, +, ZEIT=.'
oder für alle Stationen:
H31 = 'SOWI, IF, ZEIT–, +, ZEIT=., ALL–, ZEIT = x:x:x'
Im letzten Fall dürfen in den anderen Stationen keine H-Programme zur Zeitumschaltung laufen!
Angabe von Monatsparametern ab V2.46

STATION GRUPPE

STATION, GRUPPE

STATION : Stationsname
GRUPPE : optionaler Gruppenname (nicht über Bedienpanel verfügbar)

Aufruf : STATION [=<name>]
Ext : + – # %

– Beim Lesen: <name> → Zwischenablage (außer bei Extension "– –")
– maximale Länge von <name> = 8 Zeichen.
– Zeichenvorrat siehe KANAL.

STATUS STAT24V STATBAT STATREL STATREL* STATCHECK

STATUS (STAT) : Ausgabe einer Geräte-Statusmeldung
Aufruf : STATUS
Ausgabe : ja
Stack : – >>> –
Ext : + $ ##

STAT24V : Status der 24 V-Ausgangsspannung
Aufruf : STAT24V
Ausgabe : nein
Stack : – >>> {1 | 0} 1: OK, 0: Fehler

STATBAT : Status der Lithium-Speicherstützbatterie
Aufruf : STATBAT
Ausgabe : nein
Stack : – >>> {2 | 1 | 0} 0: Fehler, 1: OK
nur U1601: 2: Batterie fast leer

STATREL : Zustand des Status-Relais
Aufruf : STATREL
Ausgabe : nein
Stack : – >>> {1 | 0} 1: OK (Relais EIN), 0: Fehler (AUS)

STATREL* : Status-Relais für 10s abfallen lassen (AUS)
Aufruf : STATREL* = 0 Status-Relais bleibt 10s ausgeschaltet
Aufruf : STATREL* = 1 Status-Relais wieder aktivieren

STATCHECK : Einstellen / Abfragen der Kopplung Gerätestatus-Relais mit dem Status der 24 V-Ausgangsspannung und der Lithium-Batterie
Aufruf : STATCHECK [=<wert>]
Ausgabe : ja
Stack : – >>> <wert> 0: NICHT gekoppelt, 1: gekoppelt (Default)
Ext : + – %

– Die folgenden Kopplungen sind mit STATCHECK einstellbar:
0 : keine Kopplung, nur system-interne Funktionsbereitschaft (Sys)
1 : Sys + Lithium-Batterie ok (Bat) + 24V-Ausgangsspannung ok (24V)
– Sobald eine der Bedingungen nicht erfüllt ist, fällt das Status-Relais ab und die Status-LED erlischt.
– Auch ohne Kopplung mit STATCHECK kann die Station mit dem Befehl ERRSTAT überprüft werden. Die einzelnen Fehler können entsprechend maskiert werden und per Hintergrund-Programm das Status-Relais und die Status-LED mit dem Befehl STATREL* =0 manipuliert werden.

Aufruf	: SYNC =		das laufende Intervall wird abgeschlossen, Voraussetzung: INTERVALLQUELLE == PROG
Stack	: – >>> –		
Ext	: +		das laufende Intervall wird abgeschlossen, unabhängig von Intervall-quellen-Einstellung!
Aufruf	: SYNC		Abfragen, ob Intervall-grenze erreicht wurde
Ausgabe	: nein		
Stack	: – >>>	<SyncFlag>	<SyncFlag> == 0: sonst <SyncFlag> == 1: für 5 s nach Intervallgrenze (s.u.) <SyncFlag> == 2: Sync-Anforderung läuft gerade (<ls)

Aufruf	: SYNC*	
Ausgabe	: nein	
Stack	: <Anzahl-Intervalle-seit-PowerON>	(max. 255)

Aufruf	: SYNC**	
Ausgabe	: nein	
Stack	: <Gesamte-Anzahl-Intervalle>	(max. 65535)

Aufruf	: SYNC/	
Ausgabe	: nein	
Stack	: <laufendes-Intervalldauer-in-Sekunden>	

– VON und BIS werden stets entsprechend des laufenden Intervalls gesetzt.

– Bei Intervall-Dauer <= 5s gilt:
  <SyncFlag> == 2 : Sync-Anforderung läuft (<ls)
  <SyncFlag> == 1 : sonst

SysRESET : Prozessor-Reset durchführen (ähnlich PowerOn Reset)
Aufruf : sysRESET = 0

SysTEST : Test einiger Systemfunktionen, Ausgabe des Resultats
Aufruf : sysTEST [<Anzahl<] [=0]
Stack : – >>> ESCC2[L] ESCC2[R]
Ext : – . &

– Mit SYSTEST <Anzahl> kann das ECS-LAN überprüft werden. <Anzahl> mal werden 64 Nutzdatenbytes zu Testzwecken mit der angesprochenen Station ausgetauscht (32 Byte hin, 32 Byte zurück). Ist ansonsten kein ECS-LAN Verkehr, entspricht die gemessene Datenrate der Güte der Übertragungsstrecke.

Beispiel
<A> B: systest 100
6400 Bytes werden zwischen Station A: und B: ausgetauscht, der Zeitbedarf und die Datenrate werden anschließend ausgegeben.
WICHTIG: Pro ECS-LAN Segment zwischen Prüfer und Prüfling vermindert sich die Datenrate (Zeitbedarf * n). RICHTWERT (n = 1, 62K5 Bd):
U1600 : 2000 . . 2500 Byte/s
U1610/15 : 2500 . . 3000 Byte/s
U1601 : 3000 . . 3500 Byte/s

SysSN : Geräte-Seriennummer abfragen
SysDC : Datums-Code der Calibrierung abfragen
Aufruf : SYSSN
Stack : – >>> –

SysOPEN : Öffnen bestimmter interner Zugriffs-Sperren:

Aufruf	SYSOPEN	SYSOPEN–	SYSOPEN?
Funktion	Öffnen	Schließen	–
Ausgabe	nein	nein	nein
Stack	– >>> –	– >>> –	– >>> <openlevel>

– <openlevel> : 0 : geschlossen, >=1 : geöffnet
– Zum Calibrieren von Analog-Kanälen muss zuvor mit SYSOPEN das Gerät geöffnet werden, ansonsten erfolgt bei Calibrieren die Fehlermeldung "Zugriff verweigert".
– Die Öffnung bleibt für 4 Minuten nach SYSOPEN bestehen und verlängert sich bei Eingabe von beliebigen Befehlen innerhalb dieser Zeit um weitere 4 Minuten.
– Ist ein Gerät geöffnet, so ist es für alle Instanzen (COM1, COM2, . . .) geöffnet.
SYSOPEN arbeitet ESC-LAN-weit, bei U1600/10/15 nur lokal (AA:Station);
SYSOPEN–, SYSOPEN? sind bei U1600/10/15 nicht verfügbar.

SPRACHE : wählt die Bedienpanel-Dialog Sprache
Aufruf : SPRACHE [=<land>]
Ausgabe : ja
Ablage : ja
Stack : – >>> –
Ext : + – # . $ %

<land> : 1 | D | DEUTSCH | G | GERMANY
: 2 | E | ENGLISCH
: 3 | S | SPANISCH | ES | ESPANGNOLE
: 4 | I | ITALIENISCH
5 | F | FRANZÖSISCH

– Die Verfügbarkeit der Sprachen ist vom Stand der Firmware abhängig.
– Der ECL-Interpreter ist nur zweisprachig (Deutsch / Englisch) ausgelegt. Deutsche oder englische Befehle sind beliebig mischbar, für die Ausgabe-Sprache gilt jedoch: Deutsch bei <land> == 1, Englisch bei <land> >= 2
– Nur der erste Buchstabe von <land> muss angegeben werden, Groß- / Kleinschreibung ist unwichtig.
– Ext 'I' pusht beim Lesen den aktuellen Sprachenindex auf den Stack.
– LISTSPRACHE erzeugt eine Liste aller verfügbaren Sprachen (U1600: n.v.)
– Deutsch wählen: SPRACHE = Deutsch oder SPRACHE = 1
– Englisch wählen: SPRACHE = Englisch oder SPRACHE = 2

TAGBEG : Variabler Tagesanfang.

TAGBEG

Default-Einstellung = 00:00:00
Diese Einstellung wirkt sich auf alle vergangenen und zukünftigen Tagesanfänge aus und darf nicht dynamisch verändert werden!

Aufruf : TAGBEG [=<tagesanfangs_zeit>]
Stack : – >>> <tagesanfangs_zeit>
Ext : – . # &

TARIF : Abfrage oder Einstellen des aktuellen Tarifs

TARIF

Aufruf : TARIF [=<tarif>]
Ausgabe : ja
Stack : – >>> <tarif> (beim Lesen)
Ext : + – * %

<tarif> = {1 | 2}; '1': Tarif 1, '2': Tarif 2
– Mit der Extension * gilt <tarif> = {0 | 1}; 0': Tarif 1, '1': Tarif 2
Beispiele (mit Ext '*'):
1. Zwischen 22h00 und 6h00 gilt Tarif T2, sonst T1:
H 10= 'hh,6,<,hh,22,>, | ,tarif* =.'
2. Am Wochenende (Sa.+So.) soll T2 gelten:
H10 10= 'wtag,6,>=,tarif* =.'

TASTE

TASTE (TT) : Übergabe von Tastendrücken an das Bedien-Panel

Aufruf : TASTE <tastenkürzelkette>
Ausgabe : keine
Stack : – >>> –

Elemente der <tastenkürzelkette>, maximale Länge 20 Elemente:

1 . . 5	: F1 . . F5
+	: Pfeil hoch
–	: Pfeil runter
<	: Pfeil links
>	: Pfeil rechts
m	: Menü
s	: Setup (wie „Menü-Taste 1 s drücken“)
#	: Enter
!	: ESC
q	: ESC-ESC (alle Ebenen zurückfallen). (U1600/10/15 n.v.)
u	: Umschalten (nur U1600, wie Pfeil hoch+runter gleichzeitig drücken)
l	: Löschmenü oder ähnliche Funktion
x	: Setzen des Grundzustandes (Normalanzeige mit Eges, Kanal 1)

Beispiel
TASTE x++++4
x setzt Bedienpanel auf Normalanzeige, 4* ’+’ geht auf Kanal 5, 4 steht für F4, also Wechsel zur Pmom.

LOCKKB, LOCKKBM

LOCKKB, LOCKKBM : Sperren der Tastatur (auch selektiv)

(U1601 ab V2.45, U1600: n.v.)

Die gesamte Tastatur oder nur bestimmte Tasten können auf Timeoutbasis gesperrt, d.h. der Sperrbefehl ist nur eine bestimmte Zeitdauer gültig, danach wird die Sperre sicherheitshalber wieder aufgehoben. Verwendet werden die Befehle vorwiegend in H-Programmen.

LOCKKB : Sperre der kompletten Tastatur

Aufruf : LOCKKB [= <sperr_dauer>] <sperr_dauer>
Ausgabe : ja
Stack : – >>> <verbleibende_sperr_dauer>
0 : Sperre aufheben
1 : 5 s sperren
2...60: Sperrdauer in [s]

– Wird trotz Sperre eine Taste gedrückt, erscheint kurzzeitig die Meldung "TASTATUR GESPERRT" im Display.

LOCKKBM : Selektives Sperren der Tastatur (mit Tastenmaske) für 60 s

Aufruf : LOCKKBM [= <sperrmaske>]
Ausgabe : ja
Stack : – >>> <sperrmaske>

<sperr-maske> :

Taste	Bit	Taste	Bit	Taste	Bit
F1	0	LINKS	8	SETUP	16
F2	1	ENTER	9	LOESCH	17
F3	2	MENU	10	X	18
F4	3	ESC	11	ESCESC	19
F5	4	AUTO	12	–	
AUF	5	HAND	13	–	
AB	6	LR	14	–	
RECHTS	7	<meld>	15	–	

– Bitpositionen: (MSB)Bit-31 .. Bit-0(LSB)
– Bitwert: 1 = Taste gesperrt, 0 = Taste möglich;
Bei <meld> gilt 1 ="TASTATUR GESPERRT"-Meldung erscheint,
0 = keine Meldung.
– Masken können in binärer Schreibweise ("0b" voranstellen) verständlich formuliert werden.

Beispiel
Sperren der Tasten AUF + AB, Meldung bei Verwendung dieser Tasten:
LOCKKBM = 0b1000000001100000 oder
LOCKKBM = 32864

Trigonometrische Funktionen

SQRT SIN COS ASIN ACOS DEG RAD EXP LOG :
Mathematische Funktionen

		Stack:			
SQRT	:	x	>>>	Quadratwurzel (x)	
SIN	:	x	>>>	sin (x)	Basis ist Bogenmaß
COS	:	x	>>>	cos (x)	
ASIN	:	x	>>>	asin (x)	Umkehrfunktion von SIN
ACOS	:	x	>>>	acos (x)	
DEG	:	x	>>>	((x/pi)*180)	Umwandlung Bogenmaß in Grad
RAD	:	x	>>>	((x/180)*pi)	Umwandlung Grad in Bogenmaß
EXP	:	x	>>>	(e hoch x)	
LOG	:	x	>>>	LOGe (x)	
PI	:	–	>>>	pi	3,141592653589793

TX1, TX2

TX1 TX2

TX1	:	Senden einer Zeichenkette an COM1 (bei COM2-MIX an COM2)
Aufruf	:	TX1 <zeichenkette>
TX2	:	Senden einer Zeichenkette an COM2
Aufruf	:	TX2 <zeichenkette>

– Die Zeichenkette kann bis zu 127 Zeichen lang sein.
– Befehlsausgabe senden (verwendet den Zwischenpuffer): EGES– – 1, TX2 $
– Die Zeichenkette kann auch an eine andere Station gesendet werden.

VER

VER : Ausgabe der aktuellen Software-Version

Aufruf	:	VER
Ausgabe	:	ja
Stack	:	– >>> <Versions-Nummer>
Ext	:	+ – . $ # %

LVER

LVER : Ausgabe der aktuellen ECS-LAN-Version zur Feststellung des ECL-Befehlumfangs des adressierten ECS-LAN Teilnehmers (ab 12/2001)

Aufruf	:	LVER VER@ (alternative, stets verfügbare Befehlsform)
Ausgabe	:	ja
Stack	:	– >>> <Lan-Code> <Lan-Versions-Nummer>
Ext	:	+ – . $ # % |

– **Beispiel** für Ext.'|', die lediglich die Stackreihenfolge umkehrt: Es soll Code nur ab LanVersion ≥ 2 ausgeführt werden und es nicht bekannt, ob die Firmware den Befehl LVER überhaupt kennt:
VER@|-,dr,2,>=,if, ... // führt ... nur ab LV 2 aus
VER@|-,dr, ! // Stackausgabe: LV oder 0

VON BIS, DAUER

VON BIS DAUER.

VON BIS	:	Abfragen der Zeitzahl der letzten Ausgabe „mit Zeit“
DAUER (DAU)	:	Dauer der Zeitspanne VON ... BIS in Sekunden
Aufruf	:	VON
Ausgabe	:	nein
Stack	:	– >>> <zeitzahl> <zeitzahl>: Sekunden-zahl ab 1.1.1990

Die beiden Variablen VON und BIS werden gesetzt, sobald ein entsprechender Befehl mit Extension / oder ^ verwendet wird. Mit DAUER lässt sich die Dauer der durch VON ... BIS beschriebenen Zeitspanne in Sekunden ermitteln.

Datum/Zeit <zeitzahl>**Hinweise** zur Verwendung der Extension / und ^

/	: ^	: Ausgabe mit Zeit „bis“
/ /	: ^ ^	: Ausgabe mit Zeit „von ––bis“
/ / /	: ^ ^ ^	: Ausgabe mit Zeit „von“
/ / / /	: ^ ^ ^ ^	: Ausgabe unterdrücken (VON und BIS werden jedoch gesetzt)

Modifizieren der durch / oder / / oder / / / definierten Ausgabe (/ stets vor ^)

/ ^	: Zeit/Datum anstatt Datum/Zeit Ausgabe 31.12.93;17:33:56
/ ^ ^	: Datum/Zeitausgabe, Datum im DBase-Format jjjjmmtt 19931231;17:33:56
/ ^ ^ ^	: Zeit/Datumausgabe, Datum im DBase-Format jjjjmmtt 17:33:56;19931231
/ ^ ^ ^ ^	: Datum/Zeit Sep. ';' → ' ' (geeignet für MS-EXCEL) 31.12.93 17:33:56
/ ^ ^ ^ ^ ^	: Zeit/Datum Sep. ';' → ' ' (geeignet für MS-EXCEL) 17:33:56 31.12.93

ZEIT (TIME) DATUM (DATE) : Systemzeit und Datum stellen oder abrufen **ZEIT, DATUM**

	Stellen	Abfragen	Zeitzahl darstellen
Aufruf	: ZEIT = <zeitstring<	ZEIT	ZEIT .
Ausgabe	: nein	ja : hh:mm:ss	ja : hh:mm:ss
Stack	: – >>> –	– >>> <zeitzahl>	<zeitzahl> >>> –

Ext : + – * . / ^ _ %

– Format des <zeitstring>: 12:36:00 oder 2h15
– Format des <datumstring>: 17.03.92 oder 26.02 [Datumformate siehe DATUMFORMAT]
– <zeitzahl> ist die Sekundenanzahl ab dem 1.1.1990
– ZEIT / / gibt stets Zeit und Datum aus, DATUM / / das Datum und die Zeit. Mit ZEIT / / = 30.11 11h wird Zeit/Datum oder Datum/Zeit zusammen gestellt.
– Zeitvergleiche siehe ZEITVERGLEICHE. Sommerzeit-Umschaltung siehe SOWI.
– ZEIT* pusht (<zeitzahl>.<sekbruchteil>) Funkuhr-Synchronisation siehe DCF77.
– Zeitmessungen: [Befehlspaar TM / TMD (== ZM / ZMD)] 'tm, <Block>, tmd, !' gibt Dauer von <Block> aus, Stack muss passen! 'a = t, <Block>, a, tmd, !' gibt Dauer von <Block> aus, Stack unwichtig. Zeitangabe in Sekunden mit 1/100 s.
– Ext '^' verändert die Reihenfolge Zeit / Datum und / oder wählt ein dBase-kompatibles Datumformat. DATE/^^ → 19980427 [siehe auch VON]
– Ext '_' : statt Echtzeit wird die Betriebsstundenzähler-Zeit verwendet. Siehe ZEITVERGLEICHE.

Hinweis
Bitte beachten Sie, daß nur Zeitabfragen mit dem Befehl Zeit oder Datum die tatsächliche Zeit / Betriebsdauerzähler-Zeit der angesprochenen Stationen liefern. Alle anderen Zeitbefehle beziehen sich auf die Station, auf der der Befehl physikalisch ausgeführt wird. (So, als wenn stets die Kennung AA: dem Befehl vorangestellt würde).

TM / TMD: Zeitmessungen

'tm, <Block> , tmd,!' gibt Dauer von <Block> aus, Stack muss passen!
'a=t, <Block> ,a,tmd,!' gibt Dauer von <Block> aus, Stack unwichtig.
Zeitangabe in Sekunden mit 1/100 s.

Vergleiche mit der Systemzeit

Vergleiche mit der Systemzeit

Aufruf : IF <datum_oder_zeitstring> [<zeitstring>]
(IFF auch möglich)
Stack : – >>> –

– Da die Zeitpunkt-Angabe auch Platzhalter "x" enthalten darf, kann ein ganzer Zeitbereich aufgespannt werden (siehe Beispiel).
– Bei Verwendung von IF <zeitpunkt> in H-Programmen wird die Bedingung während des gültigen Zeitbereichs nur einmal pro Sekunde wahr.
– Wenn die Zykluszeit der H-Programme länger als 1 Sekunde beträgt, sorgt ein bestimmtes Verfahren dafür, dass Zeitpunkte nicht ausgelassen werden.
– Wenn die Systemzeit synchronisiert wird, (z.B. per DCF77-Funkuhr), sind Abweichungen der Zeit (+/–1 ... 3s) unvermeidbar. Ein Vorstellen der Systemzeit wird mit dem oben genannten Verfahren abgedeckt, ein Zurückstellen führt zu keinem erneuten Erkennen des Zeitpunkts, sofern die Abweichung geringer als –3s ist (U1600: n.v.).

Beispiel

h10 = 'IF 17.3 xh10.xx, rel 1 = 1, else, rel 1 = 0'
Am 17.3 wird zu jeder vollen Stunde + 10 Minuten
Relais 1 eine Minute lang aktiviert.

Betriebsdauerzähler-Zeit

– Die Extension '_' bei ZEIT_, TM_, TMD_, SYNC_/, a=t_ usw. verwendet statt der Echtzeit die stets momentan wachsende Betriebsdauerzähler-Zeit (0 bei Master-Reset).
– Wichtig bei Zeitdauermesseungen (z.B. während Sommerzeit / Winterzeit-Umschaltungen).

Bitte beachten Sie den Stations-Bezug von Zeitabfragen, siehe ZEIT!

Zeitzähler

– Zur ausschließlichen Verwendung in H-Programmen gibt es den Befehl HTD:

Aufruf : HTD
Stack : – >>> <ZeitDeltaInSekunden>

Auf den Stack wird die Zeitdifferenz seit dem letzten HTD Aufruf gepusht (für jedes H-Programm separat). Zeiten größer 60 s → 0 s.

– HTD* führt bei Ergebnis == 0 ein EXIT aus:
HTD*, ... ist also identisch mit HTD,DUP,0,==,IF,EXIT,ELSE,...

Beispiel zur Zeitzählung (Sekunden) in den Kanälen 10 ... 15, solange STARTSTOP von jeweiligen Kanal = 1 ist:
H 10 = 'HTD, DELTA 10..15=.'
Oder in Stunden:
H 10 = 'HTD, 3600,/, DELTA 10..15=.'
Kopplung vom Eingang 10 an STARTSTOP 10:
H 11 = 'IN– 10,STSP 10=.'

ZKONST Urat Irat PULS FLANKE EINAUS STARTSTOP (STSP) KFIX KFAKTOR : Kanalspezifische Parameter

ZKONST, URAT, IRAT, PULS, FLANKE, EINAUS, STARTSTOP, KFIX, KFAKTOR

ZKONST : Die Zählerkonstante <real>
Urat : Spannungsübersetzungsverhältnis <real>
Irat : Stromübersetzungsverhältnis <real>
PULS : Pulsdauer in Millisekunden (10 ... 2550 == 0, 01 s ... 2,55 s)
FLANKE : aktive Zählflanke oder Tarifzuordnung (Binäreingang):
1: _ _ – –Wechsel 0 V >>> 24 V (+) oder 24 V → Tarif 2
0: – – _ _ Wechsel 24 V >>> 0 V (–) oder 24 V → Tarif 1
EINAUS : Kanal ein- / ausschalten:
1: EIN 0: AUS
Die Auswahl * bei Aufzählungen des Kanals steuern:
1: START 0: STOP
STARTSTOP : Impulszählung des Kanals steuern:
1: START 0: STOP
KFIX : FIX-Punkt bei Ausgabe (0: 0 1: 0.0 2: 0.00 oder 3: 0.000)
KFAKTOR : Allgemeiner Faktor für Energie und Leistung (U1600: n.v.)

Aufruf : ZKONST <aufzählung> [=<zuweisung>]
Stack : – >>> wert (beim Lesen)
Ext : + – . # %

2.3 Befehls-äquivalenz

Befehlsäquivalenz Deutsch – Englisch

Das ECS Betriebssystem ermöglicht die Eingabe von nahezu allen ECL Kommand in Deutsch oder Englisch, unabhängig von der eingestellten Landessprache. Das Online-Hilfesystem kann ebenso mit Suchbegriffen in beiden Sprachen arbeiten.
Für die folgenden ECL-Kommandos existieren verschiedene Namen in beiden Sprachen:

Deutsch	Englisch
AEINH	AUNIT
AUFZ	ENUM
BIS	TO
DATUM	DATE
DATUMFORMAT	DATEFORMAT
DAUER	DURATIO (DUR)
EEINH	EUNIT
EGES ...	ETOT ...
EINAUS	ONOFF
EMAXTAG, ... JAHR	EMAXDAY, ... YEAR
ERRKAN	ERRCHAN
FLANKE	EDGE
INTERVALL	INTERVAL
INTERVALLQUELLE	INTERVALSOURCE
JAHR	YEAR
KANAL	CHANNEL (CHAN)
KANALFIX	CHANNELFIX (CFIX)
KENN	ID
KOSTFAK1 KOSTFAK2	COSTFAC1 COSTFAC2
LOESCHKANAL	ERACHAN
LOESCHLISTE	ERALIS
LONKANAL	LONCHANNEL
PASSWORT	PASSWORD
PEGEL	LEVEL
PEINH	PUNIT
PFAKTOR	PFACTOR
PMAXTAG, ... JAHR	PMAXDAY, ... YEAR
PULS	PULSE
RELAIS	RELAY
RELAISMODE	RELAYMODE
RELAISNAME	RELAYNAME
SETKENN	SETID
SPRACHE	LANGUAGE
TAG	DAY
TARIF	TARIFF
TARIFQUELLE	TARIFFSOURCE
TASTE	KEY
TEINH	TUNIT
VON	FROM
WTAG	WDAY
ZEIT	TIME
ZKONST	MCONST

4 Produktsupport

Bitte wenden Sie sich im Bedarfsfall an:

GOSSEN METRAWATT GMBH
Hotline Produktsupport
Telefon +49-(0)-911-8602-112
Telefax +49-(0)-911-8602-709
E-Mail support@gmc-instruments.com

5 Schulung

Wir bieten interessante Seminare mit Praktikum zum Thema „ECS, das Energy Control System für Transparenz im Energiebereich“. Bei diesen Seminaren werden die erforderlichen Geräte, zu denen auch das U1601 gehört, konfiguriert aber auch mit verschiedenen Softwarepaketen Auswertungen erzeugt und diskutiert.
Wir senden Ihnen gerne einen Seminarkalender zu.

GOSSEN METRAWATT GMBH
Bereich Schulung
Telefon +49-(0)-911-8602-406
Telefax +49-(0)-911-8602-724
E-Mail training@gmc-instruments.com

Gedruckt in Deutschland • Änderungen vorbehalten

GOSSEN METRAWATT GMBH
Thomas-Mann-Str. 16-20
90471 Nürnberg • Germany
Member of GMC Instruments Group
Telefon +49-(0)-911-8602-0
Telefax +49-(0)-911-8602-669
E-Mail info@gmc-instruments.com
www.gmc-instruments.com
GOSSEN METRAWATT