ERKLÄRUNG MAKROKOMMANDOS

Unterstehend finden Sie eine Liste mit möglichen Makrokommandos und ihre Bedeutung.
In der neue Version 6.45 wurden die Kommandos überarbeitet, sie sind jetzt in englischer Sprache und werden in voller Länge geschrieben.
Kommandos und Paramter werden durch (und) getrennt, Paramter werden durch , getrennt, evtl mit Leerzeichen.

Bsp.: lok(3, 10) lässt die Lok mit Folgenummer 3 mit Fahrstufe 10 fahren.

Die Parameter sind alle 16 bit, die Gültigkeit hängt jedoch vom Parametertyp ab.
Die makros werden in Klartext in einer Datei gespeichert.

alt: WSL <folgenummer weiche> <stellung>
neu: switch(folgenummer, stellung)
Stellt eine Weiche oder ein Signal. Ob die Weiche tatsächlich gestellt wird hängt ab:
- ob die Weiche verriegelt ist und
- vom Stellungsparameter.

<stellung>:
G für Weiche gerade oder Signal frei,
R für Weiche abbiegend (rund) oder Signal halt,
L für Weiche link oder Signal langsam,
D für Weiche abbiegend (2e Stellung).
Bei diesen Werten wird die Weiche nicht gestellt wenn sie verriegelt is.

GA für Weiche gerade oder Signal frei,
RA für Weiche abbiegend (rund) oder Signal halt,
LA für Weiche link oder Signal langsam,
DA für Weiche abbiegend (2e Stellung).
Bei diesen Werten wird die Weiche auch gestellt wenn sie verriegelt is.

GB für Weiche gerade oder Signal frei,
RB für Weiche abbiegend (rund) oder Signal halt,
LB für Weiche link oder Signal langsam,
DB für Weiche abbiegend (2e Stellung).
Bei diesen Werten wird die Weiche nicht gestellt wenn sie verriegelt is, wenn sie gestellt wird, wird sie nachher verriegelt.

GAB für Weiche gerade oder Signal frei,
RAB für Weiche abbiegend (rund) oder Signal halt,
LAB für Weiche link oder Signal langsam,
DAB für Weiche abbiegend (2e Stellung).
Bei diesen Werten wird die Weiche auch gestellt wenn sie verriegelt is, wenn sie gestellt wird, wird sie nachher verriegelt.

alt: VRY verriegelung freigeben.
neu: free verriegelung freigeben

Beispiel
switch(5, r) Stelle Weiche 5 abbiegend.
switch(12, l) Stelle Signal 12 langsam (Hp2).
switch(3, gab) Stelle Weiche 3, auch wenn sie verriegelt ist, und verriegele sie nachher.
switch(3, free) Die Verriegelung von Weiche 3 freigeben.

alt: LOK <folgenummer lokomotive> <geschwindichkeit>
neu: lok(folgenummer, fahrstufe)
Lässt eine Lokomotive mit bestimmter Geschwindigkeit fahren.

<geschwindigkeit>: Kommando für den Interface:
0 - 127: geschwindigkeit. Wenn die Lok eine Masse hat wird die Geschwindigkeit langsam erreicht.
Für die Funktionen wird das Kommando FCN benutzt, Richtungsänderung geht mit Kommando KER.

Beispiel
LOK 003 008 lässt Lok 3 fahren mit geschwindigkeit 8

alt: KER <folgenummer lokomotive>
neu: dir (folgenummer)
    Ändert die Fahrtrichtung einer Lokomotive.

alt: FCN <folgenummer lokomotive> <funktionen>
neu: function(folgenummer, funktionen)
    Setzt die Funktionen einer Lokomotive. 'funktionen' ist die Summe der
    nachstehenden Werten:
     1 = funktion 0 (licht) einschalten
     2 = funktion 1 einschalten
     4 = funktion 2 einschalten
     8 = funktion 3 einschalten
    16 = funktion 4 einschalten
    .... usw
    32768 = funktion 15 einschalten

alt: LLS <folgenummer lokomotive>
neu: lokspeedslow(folgenummer)
    Lässt eine Lok weiterfahren mit ihre 'langsame' (30kmh) geschwindigkeit.

alt: LHS <folgenummer lokomotive>
neu: lokspeedsfast(folgenummer)
    Lässt eine Lok weiterfahren mit ihre 'schnelle' (120kmh) geschwindigkeit.

alt: STR <folgenummer fahrstrasse> <AAN/UIT>
neu: route(folgenummer, on/off)
    Stellt die Weichen einer Fahrstrasse, und legt diese fest ('AAN') oder
    gibt eine Fahrstrasse frei ('UIT').

    Beispiel
    route(5, on)   Stellt die Weichen der Fahrstrasse 5 und legt die Fahrstrasse fest.
    route(5, off)   Gibt die Fahrstrasse wieder frei.

alt: PMA <meldekontakt>
neu: contact(kontaktnummer, on)
    Wartet (pause) mit weiterlauf der Makro bis das Meldekontakt eingeschaltet wird.
    Danach läuft der Makro weiter.

    Beispiel
    Das Beispiel zeigt einen Makro der Lok 2 langsam fahren lässt und nach vorbeifahren
    von Meldekontakt 5 Lok 2 halten lässt.
    01 lok(2, 4)
    02 contact(5, on)
    03 lok(2, 0)

alt: PMU <meldekontakt>
neu: contact(kontaktnummer, off)
    Wartet (pause) mit weiterlauf der Makro bis das Meldekontakt ausgeschaltet wird.
    Danach läuft der Makro weiter.

alt: PSE <sekunden>
neu: pause(sekunden)
    Wartet (pause) mit weiterlauf der Makro bis die aufgegebene Zeit abgelaufen ist.
    Danach läuft der Makro weiter.

    Beispiel
    Dieser Makro lässt die Lok 2 anhalten (am Bahnhof ?) und nach 15 sekunden
    weiterfahren.
    01 lok(2, 0)
    02 pause(15))
    03 lok(2, 8)

alt: JMP <meldekontakt> <makrozeile>
neu: jump(kontakt, makrozeile)
    Lässt der Makro mit einer anderen Zeile weiterlaufen wenn das meldekontakt
    eingeschaltet ist.

    Beispiel
    jump(5, 7)   lässt der Makro witerlaufen mit Zeile 7 wenn Meldekontakt

5 eingeschaltet ist, sonst läuft er weiter mit Zeile 2.

alt: GTO <makrozeile>
neu: goto(makrozeile)
Lässt der Makro mit einer anderen Zeile weiterlaufen.

Beispiel
goto(5) lässt der Makro witerlaufen mit Zeile 5.

alt: BEL <zeit>
neu: bel(zeit)
Lässt während der aufgegebene Zeit einen 'beep' aus den Lautsprechrer des Pc hören. <zeit> ist in 1/20-stel sekunden.

Beispiel
BEL(20) lässt eine Sekunde einen 'beep' hören.

alt: STP
neu: stop
Nothalt

alt: IFW <folgenummer weiche> <stellung> <zeile>
neu: ifswitch(folgenummer, stellung, makrozeile)
Testet die Stellung einer Weiche. Wenn die Stellung den zweiten Parameter gleicht geht der Makro weiter mit Zeile <zeile>. Stellung kann die Werte G, R oder L haben.

Beispiel
In diesem Beispiel wird Signal 3 auf halt gestellt wenn Weiche 2 abbiegend ist, sonst wird Signal 4 auf halt gestellt
01 ifw(2, r, 4)
02 switch(4, r)
03 end
04 switch(3, r)

alt: IFS <folgenummer fahrstrasse> <stellung> <zeile>
neu: ifroute(folgenummer fahrstrasse, stellung, zeile)
Testet die Stellung einer Fahrstrasse. Wenn die Stellung den zweiten Parameter gleicht geht der Makro weiter mit Zeile <zeile>. Stellung kann die Werte on (ein) oder off (aus) haben.

Beispiel
In diesem Beispiel wird Signal 3 auf halt gestellt wenn Strasse 5 eingelegt ist, sonst wird Signal 4 auf halt gestellt
01 ifroute(5, on, 4)
02 switch(4, r)
03 end
04 switch(3, r)

alt: IFT <blockabschnitt> <makrozeile>
neu: ifblock(blockabschnitt, makrozeile)
Lässt den Makro weiterlaufen mit Zeile <zeile> wenn der Blockabschnitt besetzt ist.

alt: IFB <blockabschnitt> <loknummer> <makrozeile>
neu: iflok(blockabschnitt, loknummer, makrozeile)
Lässt den Makro weiterlaufen mit Zeile <zeile> wenn der Blockabschnitt besetzt ist mit Lokomotive <loknummer>.

alt: CMD <a><b><c>
neu: command(a, b, c)
Sendet die Zahlen a, b en c direkt zur Interface.

alt: REG <regler> <blockabschnitt>, nur für Edits I system.
neu: controller(regler, blockabschnit)
Verbindet Regler <regler> mit der Lok in Blockabschnitt <blockabschnitt>.

alt: PRG <makronummer>
neu: start(makronummer)

    Startet den Makro mit nummer <makronummer>. Nummer muss >= 1 und <= 100 sein.

alt: BVK <folgenummer blockabschnitt> <wert>
neu: block(folgenummer blockabschnitt, wert)
    Meldet einen blockabschnitt besetzt oder frei. Der Blockabschnitt wird im
    Gleisbild eingefarbt. Bei Freimeldung folgt keine Zugverfolgung, die Lok
    'verschwindet' vom Gleisplan.
    wert = on (ein) oder off (aus)

alt: END
neu: end
    Endet den Makro

Variabelen
==========

Das Makromodul des PctWin kennt 26 programmvariabelen. Diese Variabelen können
einen Wert enthalten von 0 bis 255. Eine bestimmte Variabele steht jeden Makro zu
verfügung, wenn in Makro 1 eine Variabele einen Wert bekommt, kann diesen z.B. in
Makro wieder gelesen werden. Jede Variabel wird angedeutet mit einer Buchstabe A
bis Z
Die Kommdos für die Variabelen finden Sie hier unten.

alt: LET <var> <wert>
neu: let(var, wert)
    Gibt eine von den 26 Variabelen einen wert. Als var wird hier eine Buchstabe
    gegeben, A bis Z. Der Wert soll 0 bis 255 sein.

    Beispiel
    01 let(b, 10)   gibt Variabel B den Wert 10

alt: INC <var>
neu: inc(var)
    Erhöht den Wert der Variabele mit 1.

    Beispiel
    01 LET   Q 010   Q hat den Wert 10
    02 INC   Q   Q hat jetzt den Wert 11

alt: DEC <var>
neu: dec(var)
    Zieht 1 von dem Wert der Variabele ab..

    Beispiel
    01 LET   Q 010   Q hat den Wert 10
    02 DEC   Q   Q hat jetzt den Wert 9

alt: IFV <var> <wert> <zeile>
neu: ifvar(var, wert, zeile)
    Testet den Wert der Variabele, wenn dies den 3. Parameter gleicht geht
    der Makro weiter mit Zeile <zeile>.

    Beispiel
    Lässt Lok 2 10 Runden fahren und hält dann die Lok an.
    01 INC   A
    02 IFV   A 010 004
    03 END
    04 LOK 002 000

alt: IFG <var> <waarde> <regel>
neu: ifvarg(var, wert, zeile)
    Testet den Wert der Variabele, wenn dies grösser ist als den 3. Parameter
    geht der Makro weiter mit Zeile <zeile>.

Kommandos für die Zugverfolgung
===============================

Die nachfolgenden Kommandos können nur verwendet werden wenn Zugverfolgung aktiviert ist.

alt: LBV <blockabschnitt> <geschwindigkeit>
neu: blockspeed(blockabschnitt, fahrstufe)
    Gibt die Lok in dem blockabschnitt die gegebene Geschwindigkeit. Die jetztige geschwindigkeit wird behalten und kann mit dem Kommando LBS aufgerufen werden um eine Lok mit der vorigen Geschwindigkeit weiter fahren zu lassen.

alt: LBS <blockabschnitt>
neu: blockspeedprev(blockabschnitt)
    Lässt die Lok die im Blockabschnitt angehalten war mit einem der Kommandos LBV, LBL oder LBH weiterfahren mit der vorherige Geschwindigkeit.

    Beispiel
    Dieses Beispiel lässt jede Lok in blockabschnitt 4 anhalten, 5 Sekunden warten und wieder weiterfahren.
    LBV 004 000   lässt die lok in block 4 anhalten
    PSE 005   warte 5 sekunden
    LBS 004   lässt die lok in block 4 weiterfahren

alt: LBL <blockabschnitt>
neu: blockspeedslow (blockabschnitt)
    Lässt die Lok in dem Blockabschnitt weiter fahren mit der Geschwindigkeit die bei der Lok unter 30kmh aufgegeben ist.

alt: LBH <blockabschnitt>
neu: blockspeedfast(blockabschnitt)
    Lässt die Lok in dem Blockabschnitt weiter fahren mit der Geschwindigkeit die bei der Lok unter 120kmh aufgegeben ist.

alt: VPL <von> <nach>
neu: blockmove(von, nach)
    Hilft die Zugverfolgung durch eine Loknummer von Blockabschnitt <von> nach Blockabschnitt <nach> weiter zu schieben. Unter spezielle Anforderungen kann dieser Kommando hilflich sein die Loknummer schneller bekannt zu haben in einem Blockabschnitt.

alt: VPA <von> <richtung>
neu: blockmoveauto(von, richtung)
    Hilft die Zugverfolgung durch eine Loknummer von Blockabschnitt <von> nach dem nächsten Blockabschnitt in <richtung> weiter zu schieben.

========== ENDE ==========