Hinweis für die Textformat-Version; um diese Version im PDF Format zu lesen, laden Sie bitte https://www.dk1ri.de/myc/Rules.pdf

MYC Rules

Author: DK1RI
Version V02.00.04 20260616
This project can be found in https://github.com/dk1ri also.

Introduction

The program for the logic device LD https://dk1ri.de/myc/logicdevice.zip is integrated into the commandrouter now; corrections and enhancements will be found there
https://dk1ri.de/myc/commandrouter.txt or https://dk1ri.de/myc/commandrouter.pdf
Because some condition could not be described with rules defined in this document in version earlier than V02.00.01, the syntax was modified and may be not compatible with the earlier version. The LD with version V02.00.01 and later will support the new syntax. The rules of existing FUs (by DK1RI) will be changed if necessary.

This paper describes the syntax of rules of the MYC system.
Rules are used by the logic-device, which is part of the command-router.

Definitions and formats

see https://dk1ri.de/myc/Definitions.txt or https://dk1ri.de/myc/Definitions.pdf

Some explanation / General Rules behavior

Rules announcements start with “R;” or “S;” or “Q;”
“R” rules are valid always, “S” rules are used during configuration only. “Q” rules are used for login.
The LD uses “R” and “S” rules and do not distinguish between “R” and “S” rules.

“R” , “S” rules

These rules may have these properties:

- Rules are defined by normal devices (FU) or the rule device (RU).
- Rules can block any command (left side) completely. The blocking is defined by conditions (right side).
- By default all commands are working always. A rule will block a command under a specific conditions.
- Commands of a FU should not be executed by change of a status of another FU. A rule can initiate this: eg the device can be switched off under special conditions.
- The LD can send a fixed command if a special status is set one time, but cannot vary the command depending on the status.
- A rule cannot use descriptions. It uses transmitted values only.
- Rules are not checked completely for errors or inconsistencies! Wrong rules may crash the program!
- A rule can restrict a range of numeric parameters used by commands

Each rule is one announcement line (for FU) or a (MYC string) answer or info of an "an" command (for RU).
The "oo" and "ou" command has a temporary effect on a device only, so a status cannot be stored for these commands and the commands cannot be used for conditions (right side).
For the "xo" command the status of the corresponding "xp" is updated with the steps parameter, if the "xp" commands is used for conditions (right side) and the "xo" is not blocked.
"oa" and "aa" commands have a limited number of memory and the status can be used as condition.
"ob" and "ab" in rules are not supported now (may be later).
Operating commands can be blocked only (left side); only answer commands can be used for conditions (right side).
The SK may observe rules or not. The CR ignore rules but ignores commands with wrong number of parameters or wrong parameter. The CR send the commands to the LD, The LD requires commands with correct format and check the commands for rules.
The LD (and the CR) do not use <des>, it uses transmitted values only.

A FU will have rule announcements to inform about dependencies, which exist between the commands and status within the FU.
If there are dependencies between different devices, the rules-device RU will provide these rules.

The syntax of the rules is very simple. Lines are readable, because rules must be easily programmed as a part of the announcements of FU.

normal blocking rules
Normal blocking rules use operating command-token (for left side).
Command-token of rules of devices are translated to command-token for transmission. Rules generated by the rules device use translated command-token always.
All "R" rules are always handled by the LD in the sequence it get the rules, starting with rules of FU.
The LD read commands and answers and use both to store a status of the system
The LD has the status for switch, range and array memory commands of the system at any time and know all "R" and "S" rules. The LD has no knowledge of the content the memories of the devices (of "xm", "xn", "xf" commands); so rules cannot have the content of these memories as a condition for blocking any command.
The LD checks incoming commands against all rules. The LD will block the command of the left side if the status of the LD matches a rule. There is no way to unblock a blocked command, but blocking ruls can be skipped.
If a matching !$~ rule is found, the following rules are not checked (skipped). If then a matching !!$~ rule is found, the commands have its previous (unblocked) status again and other blocking rules may block commands.

Direct blocking rules
Direct blocking rules use hex data for the left side. This is used, when a command should not blocked completely, but only with specific parameters.
The hex data must exactly match the incoming data. "~~" can be used for bytes, that can have any value
use: R;?<command> …

Direct command rules
Direct commands use hex data for the left side.
Direct commands are handled independent of blocking.
If a command should be generated by the LD under some conditions,
use: R;<command> …
Example is a reset under some conditions. The command will be executed on time only.
A simple device like switches use “r” or “s” commands, which require a rule (provided by the RU) with direct commands.
FU may have direct commands, but this make no sense: the FU will do this action by itself and send send an info.
The “AFTER” separator between left side and right side is not supported anymore.

"Q"-rules
“Q” rules are produced by the RU only and sent to the CR only and are not forwarded. They are used by the CR only, because the CR knows the interface of the user. They are used for user management.
As default the command-router allow all commands for all devices. The CR can block commands of a device or a complete device, if necessary, for a specific user using “Q” rules.
The sequence of “Q” rules is important!

Format for Q rules:
special rules:

Q;*;*	login required for all. All command allowed (default at start)
Q;*:!*	login required for all. All command forbidden, except commands mentioned before

Q;<username>;<c> [<c>]….	allow command (used together with Q;* !*)
Q;<username>;[!]<c> [!<c>]….	allow command (used together with Q;* *)
Q;<username>;!*	disallow all except those mentioned before

A basic command-token of a device only means all commands of the device.

Rules syntax for *R” and “S” rules*

<left side> IF|UNLESS <right side>

left side:
Left side token <c> must be operating command token only, It is not allowed to block answer token.
A command can be blocked completely only. If you want to block a paramter of a commnd (one specific stack eg) you must split the command.

!$<c>[!$<c1>]...	<c> <c1> .. are blocked; $ is optional
!~	following rules are skipped
!!~	following rules are used again
?<command_as HEX> ...	exactly this command will be blocked
<command_as HEX> ...	commands will be sent to FU; generated by RU only. Not allowed (ignored) within a device

IF|UNLESS	separate left side and right side

right side:
The token used for the right side must be token for answer commands only. So a device require its own answer commands, if these are used for rules. If the RU want to use command token without answer command for the right side, it can provide commands like <c>;ax,as<c1>. This should be avoided.
The right side calculation uses a stack architecture: it calculates the each <condition> and uses push and pop to calculate the result.
<condition> [<logical operator> [<logical operator>] ..<condition>]….] (separated by one space each)
Execution is from left to right. Multiple operators are stacked: the last one is executed to the next condition first.
at least one <condition> is necessary
Each condition delivers a True or False.

<condition>:
For each transmitted parameter of a command a <parameter> must be given with the sequence of the transmitted data!
$~|$<commandtoken>&<parameter>[&<parameter>]..
$~ means: always true. With a commandtoken at the left side, this is blocked always; a direct command will be executed always
"$" and "&" are optional
As for transmitted data the last <parameter> is the "real value" at least; others <parameter> may be "positions" or "real values" depending on the command.

<parameter>: <|>|=|!<value>
<value>: n|s|"~" |<calculated value>|<n>TO<n>
<value> may be "~" (true always; match only, if for all positions the following position / data matches.)
<n>TO<n> compare operator "=" is allowed only. Allowed for numeric "real values" and positions only.

For "aa" commands for the position "<|>|=|!" can be used. But be careful: with mixed string and numeric real values, there may be never a match
If "s" (strings) is used as "real value", "=" will be used always.

<calculated_value>: will be defined later
A condition is true if all parameters are true (match)

Example:	status for or switch without stack and 6 switch positions:	000111
	rule with true result:	=3=1
	rule with true result:	>2=1
	rule with false result:	<4=1
	rule with false result (positions 3 is ignored, other must match):	!3=1
	status for or switch without stack and 6 switch positions:	111011
	rule with true result (positions 3 is ignored, other must match):	!3=1
	rule with false result (for true all positions must be "1")	=~=1

position are "0" based!

<logical operator>:
!|OR|AND|(|)

Special rule:

!!$~ IF ~	This rule is added by the CR, if a device has a “!$~” rule without following “!!$~” rule to avoid blocking of other devices.

Reserved characters / words

R;|S;|Q;	start of rule line
space	separator

left side:

!$ or !	block command
!$~ or !~	block all following rules, not previous
!!$~ or !!~	if !$~ is used by a device, !!$~ should follow (at the end of the rules at least) to avoid blocking for the next device.
?<commands>	direct blocking command
<command>	direct command
IF|UNLESS	separate left side and right side

right side:

“ “	mandatory separator between conditions and operators:
AND	logical operator AND
OR	logical operator OR
!	logical operator negation
( )	bracketing

within conditions, no space:

$	command-token following (optional)
&	parameter-condition following (optional)
< > = ! TO	compare operator, TO can be used for a range commands (“op”) only
~	any value
+ - * /	arithmetic operator
n	numeric value
s	string (without length) must not contain spaces and comparison the operator “=”. “=” as comparison operator allowed only.

Some hints for conditions

Usually all commands are valid always and can be blocked by rules. A rule describes, if a command will not be executed under some conditions. There is no way to execute commands blocked by a rule of a device.
A device usually has no rule with direct commands. It may use direct commands for initiating answer commands eg; but can send infos instead. So this should be avoided.
Rules with a direct command are executed once always if conditions are “TRUE”; any command will initiate this. The condition must change to “FALSE” and then to “TRUE” again to execute the command again.

Rules with the same commandtoken on the left side are “Ored” and cannot be “ANDed”. “AND” can be on the right side of a rule only. It is recommended to define all conditions for a commandtoken within a device in one rule only.
The LD stores the complete status of commands (not for “m” , “n” annd “f” commands)
The LD create a list of blockd commands after each change of the status.
A “~” in “location” matches all locations.

A restriction of the values of a command can be done.

The “!$~” rule block all commands if the condition is “True”. So rules following are ignored until a “!!$~” rule is found.
Float values for condition of “xa” commands must be entered in the coded hex format. This may be changed later.
Be careful to use commandtoken for the right side and left side!! Commands may be blocked forever!

Examples

R;!$2 IF 5=0	means: Command 2 will not work if status of command 5 is 0
R;!$2 IF 6>2=1	means: Command 2 will not work if status of the any of pos 3 to pos x of command 6 has data 1 (example for an “xr” command)
R;!$4 UNLESS 7=0=3500000TO3799999 OR 7=0=7000000TO7199999	means: Command 4 will work only unless state 7 is in the range of .....
R;!$2 IF 3=at	condition with string
Q;!* 1 30TO50	commands 1 and 30 to 50 forbidden for all
Q;* 0TO100 !500TO550	0 to 100 allowed, 500 to 550 forbidden for all

Copyright

Dieses Dokument darf unverändert kopiert werden.
Die Ideen in diesem Dokument unterliegen der GPL (Gnu Public Licence,V2) soweit keine früheren, anderen Rechte betroffen sind.
Die Verwendung der Unterlagen erfolgt auf eigene Gefahr; es wird keinerlei Garantie übernommen.
This document can be copied without changes.
The ideas of this document can be used under GPL (Gnu Public License, V2) as long as no earlier other rights are affected.
The usage of this document is on own risk, there is no warranty.