Hint for the txt-format-version; to read this version as PDF please load https://www.dk1ri.de/myc/commands.pdf

MYC Announcements and Commands

Author: DK1RI
Version V03.00.03 20240415
This paper is published in https://github.com/dk1ri as well

Introduction

This paper describes the announcement and command syntax.
For more details of the MYC system please check the reference.

Definitions and formats

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

Announcements

An announcement line uses a readable string using all characters except “. Some other characters have a special function: see definition of the sm format for strings. So the number of the command token is a readable figure, 1 eg for a hex 0x01 command.
A complete announcement of a device consist of one line with the basic announcement, lines of command announcements, lines for the reserved tokens, lines of rules , lines for translation and a one line I-announcement; in this sequence.
The command announcements describe the commands the device will understand.
Lines with the basic announcement, lines of command announcements and lines for the reserved tokens contain:
<command_token>;<commandtype>;<parameter>;<OPTION>
Two of the reserved tokens are used by the CR to identify a device, also, if more than one of the same device-group (same hardware and firmware) exist. (command_token 0x00, 0xxxff)
The basic announcement (command_token 0x00,) contains the description of the device. For details see [5]
The rules describe the restrictive conditions, when and how commands will not work for a device. By default any command is working at any time.
Translation lines give a translation of names use in the announcements to other lanuages.
The CR concatenate the announcements of the devices to a full announce-list.
The I-announcement is inserted by the CR to the full announce-list for each device. It shows some individual parameters of the device, so that RU and

SK can identify the device in all cases.
Some operating commands may have an influence on the status of other commands. This can be handled by rules as well.
Rules lines start with R", "S", "Q" or "T". "R" rules of devices are included in the full announcment list; others not.
"Q" rules are sent by the RU to the command-router. They are used for user management.
"S" rules are used by devices during configuration and are not included in the full announcement list. They are used during configuration.
"T" rules are sent by the RU to the command-router. These are optional translations of the descriptions within announcements. They can be used to handle other languages or modify the semantic function of a device. For details see [8].
Translation lines start with "L". The first "L" line contain the languages as name in this language as:
L;english;deutsch;francais
The first language is the language used in the announcements.
Other lines contain the translated words in the sequence of the first "L" line as:
L;switch;Schalter;bouton;pressure,Druck;pression;
L;speed;Geschwindigkeit;vitesse
Do not forget the ";" at the end of the first line.
Predefined words like ANNOUNCEMENTS should not be added.

Each command announcement line contain the unique command-number, the command-type and properties as necessary; in this sequence. A special type of properties are optional <OPTION>. Some <OPTION> will not result in transmitted data and will be at the end of a line. The first <des> of these properties is the uppercase name for the <OPTION>, followed by other parameters.

Each command belongs to a command-type. Command-types have two letters: the first denotes the operation type, the second the operating object / function
.
Operation type (first letter of command-type: o, a, r, s, i, j, z)
Active operating is denoted by "o" as the first letter, answer commands start with "a". If a function can operate and answer, the "as" notation should be used for the answer command: eg 33:as,as32 . The answer command should directly follow the operate command. The CR may extend the answer with the complete content and replace the "as32" by "ext32" Commands with "as" and "ext" use the same properties as the corresponding command.
The "ext" is always used to show the SK, that two commands belong together.
It is recommended to use corresponding read commands if possible. This will help the SK to have an actual state always.
The "r" and "s" types are identical to the "o" and "a" type and therefore not mentioned in the list below. These are used by simple devices as switches.
The devices usually send commands similar to SK; the "r" command means, the devices want another device to operate (require to operate); it is similar to an "o" commands. The result of these commands is defined by rules. The "s" command can be used to get data from the LD and send the

data to somewhere; it is similar to an “a” command (require to answer). For details see [9]. These commands are not part of the full announce-list created by the CR. A device can have “r” and “s” commands and “o” ans “a” commands as well.
The operation type “i” denotes information only. They have no data traffic.
The operation type “j” denotes information only. They have no data traffic. It is an operating command, which is used internally, usually by rules. An example is a reset under externally initiated conditions.
z is used for commands without function (placeholder)
“k” and “l” commands are not supported anymore. These are commands for the configuration phase. This is replaced by adding “14,CHAPTER,ADMINISTRATION” at the end of the announceline.

Operating object / function (second letter of command-type: r, s, t, u, p, o, m, n, f, a, b)
Each operating object denotes a general function like “s” for a switch and exactly defines the number and type of the properties. From this other devices will know the details of the devices function and the length of a properties and therefore the length of a command. The list below show the announcement templates, corresponding command, answer / info for all defined command-types.
In some cases, different operating objects can be chosen. A frequency control will obviously get a range “p” type, because it covers a range of values, which can be easily defined with a min and max value. But what about an address ranging from 1 to 10? This can also be realized as a switch with 10 positions or a range type command. The command will act identical; the display on the SK may be different. If the real values are not a range or each position require a non sequential, individual label the switch may result in a simpler SK.
A rule of thumb is, that the range, p command-type will be used, if there are "many" equal distance values in a range.
Optional properties are defined for some command-types and are optional for an announcement. If they are defined in an announcement, they must be used in a command as defined.

In general a FU is a very simple construction with limited communication bandwidth. So command communication should be simple and short, but announcements are communicated rarely or never, so they can be longer and descriptive in a readable format.
Announcements are stored in the devices, They are unique to a hardware / firmware combination and will never change (except, if the version changes as well). Therefore they can be stored also in a database or with the CR.
The CR may find the details of the attached devices somewhere and probably will not ask the devices for detailed announcements.
The CR, LD, RU should have more compute power. They should be able to build up a complete MYC System by reading the announcements without operator interaction also in a varying environment. To ensure this, the description in the announcement should be sufficient and not too short. This is valid for SK as well. If not – as for simple SK – they will be handled in a special manner.
For security reasons the CR will connect known devices only. So it must know a place, where all "its" possible devices are listed.
May be, that not all devices are active every time, but the CR, LD, RU and SK must be able to handle this situation. So, the answer of a (announcement) 0xxxf0 command to the CR is not static. It may vary, when devices disappear.

All devices must have announcements for the mandatory reserved commands. Additional reserved announcements depend on the device.
Every FU and simple SK belong to a unique device-group. This device-group has a unique name and has a not changeable set of announcements and firmware.
The CR collects the announcements of all device and concatenate the lists to a full list, excluding controlling devices (all SK and higher level CR). It also drops the announcement of individualization and some other lines but add its own lines for reserved commands (0 and 0xxxf0 – 0xxxfd).
Additional 16 token are used for communication with the controlling devices, so that 0xxxe0 - 0xxxff are reserved in the full list.
How announcements can be called by the commands 0x00 and 0xxxf0 is described in [5]
The preferred representation of announcements in programs and files is
DATA"<announceline>"
DATA"<announceline>"
…
So the character " should be not used in announcements.

General syntax for a command-announcement

<c>;<ct>[,<des>][<;pa>[,<des>]]...[;<pa>[,<des>]]...

Commands and properties

The command tokens use numeric n byte big-endian format. The shortest format possible is always used.
For properties the same rule apply. For memory content a string is allowed as property as well.
If a device has less than 239 (0, 240- 255 are reserved,) commands, the communication with the CR should use one byte format for the command token. The number of bytes used is given in the basic announcement-line.
If the CR has more than 223 commands (0, 240 - 255 is reserved, 16 for SK communication), it will communicate SK with 2 or more bytes. In this case, the first byte of translated command-tokens must not be 0 (but can be 1), because one byte 0x00 is reserved for the basic announcement.
Shortest possible format must be used for properties in any case. So the CR, LD, RU and SK know the format by the announcements.
A command consist of a command-token and properties (parameters) depending on the command-type.
The command-token is unique within a device; but see command optimizations below. A line with the 2nd instance of a command-token and different command-type will be ignored by the CR.
The number and type of the properties of a command and answers must match the announcement. If the announcement describe a min and max property and unit, the command will obviously have one property only for the value. For details see List of Standardized Command-types below.

There is no rule for FU for the numbering of the command-token, but simple sequenced numbering is recommended. The CR will put the command-tokens of the known FU, RU and lower level CR together, so that all commands are sequenced with translated command-tokens in the same order as the original announcement lines. The translated command-tokens start with 0x01 (0x0100, 0x010000,.. (no "0" as first byte) in a simple sequence without gaps. There may be gaps in the list, if a known device disappears. The CR will include all known devices from start to avoid renumbering for higher level CRs.
The full list will not have 0xxxF0, 0xxxFE and 0xxxFF lines and some others of other devices, but the 0xxxF0 command of the CR.
The CR will include 0x00 of other devices into the complete command-list but answer them by itself.
An announce-list from a lower level CR will have the own (CR) basic announcements at the beginning and the I-announcement the end. The CR will not change the sequence, so that the hierarchical structure is visible.
The reserved tokens of the individual devices – if forwarded - are translated by the CR as well; the own (reserved) tokens of the CR are 0xxxF0 ...

All devices except FU must interact with the CR with translated tokens.
The CR will translate the inline command-tokens of the content of commands, announcements and rules as well.

There is no special “not valid” command-token. If a device must answer but have no data, it will answer with a not used command-token.

General syntax for a command

<c>[<p>]...[<p>]

Info / answers

Some devices can send answers without a corresponding command as info. Not all devices will send infos.
Sometimes a command has an additional effect depending on the commands in the past. The FU may send infos in this case to inform the SK.
Devices may store the status of some commands. If a command changes a stored status the SK must be informed about the state of the changed commands. This can be done by info with the changed states. So an operate command may answer with infos. Or it can be done by rules – and the CR will inform the SK- or the FU inform the SK directly.
There are no direct answers for operating commands (with same command token)
Infos and answers use the same format.
If a FU send infos, this must be given by the 0xxxfa command; for details see [5]

General syntax for info

<c>[<p>]...[<p>]...[data]

Data transfer type

These data transfer types are specific for the existing implementation of the CR. Other CRs may work in a different way.
The CR will handle incoming data using 6 different transfer types. The differences are due to the fact, that for some commands the CR will know the length of a command immediately when the CR got the command-token. For others the length vary and the CR must calculate the length in real time. For strings the CR must read the length of the string first.

0:	some switches	no property, just forward command
1:	switches, range commands, om, of, on numeric all answer commands	all properties numeric and not changed at runtime
2:	on with string:	properties fixed numeric and one string
3:	oa	either one string or numeric
4:	ob	any number of mixed string / numeric
5:	ix, answers of operating commands	do nothing, not applicable

Announcement / Command optimizing

The following optimizations may be used by all devices.
Some are resolved by the CR, the SK should understand the others.

Character font in announcements
The characters ”.”, “;”, “””, “_”, “{“ and “}” cannot be used in a text-element (label). Depending on the implementation some others cannot be used as well.
In this case the ASCII notation 0xnn (nn is the ASCII code) can be used.

Long announcement lines
If an announcement is too long to fit in one line more lines can be used. The command-token and command type must be the same.
So

11;aa,Control;a,Preset;a,Motor_cw;a,Motor_ccw;
11;aa;a,Limit;a,Underlimit;a,Overlimit
(the second line must immediately follow the first one) is the same as
11;aa,Control;a,Preset;a,Motor_cw;a,Motor_ccw;a,Limit;a,Underlimit;a,Overlimit
The CR simply paste the lines together omitting command-token and command-type of the second line. Take care on the ";" (or ",") at the end of the first line and the sequence of the lines!

Save space in announcements
If a function can operate and answer, the answer announcement can be simplified. It must directly follow the operate announcement. The CR will resolve the this to a full line and add the "ext" (extend) description. So the relationship is not lost.
5;op,Rotatoroffset;360;lin;degree
6;ap,as5 Nothing else must follow
will be resolved by the CR to
6;ap,ext5,Rotatoroffset;360;lin:degree
There is one answer "as" command for an operating command They are using the same properties.

Optimize command transmission
If a command should be very short and a parameter should be part of the command-token, the following can be used
Instead of a the two byte command
2;os;1;0,manual;1,preset
you use 2 one byte command-token
2;ou;1;0,idle;1,manual
3;ou,ext2;1;0;idle;1,preset ext as extend. The extend is an information for SK, that it is one switch and to use the same property.

different method
Some functions can be accessed with different methods. An oscillator can be controlled with a value (by memory or knob) or as scan function. The "ext" (extend) description show the relationship to the other command, and that the same property is used.
2,op,VFOA;1;30001,{3500000 to 3800000};lin;Hz
3,oo,ext2;1;255;25;50;s;b,up

number of stacks

If a function with the same effect appears multiple for a FU, announcements can be simplified using the stack feature. The command works on one function at a time. This may simplify the SK. If you have a lot of switches selecting 4 values eg, the SK may provide an additional selector for the number of the switch.
Sometimes this function can be realized by a memory function as well, but the display on the SK may be different.
The number_of_stack parameter is not transmitted if there is one stack only.
The number_of_stack parameter can be used for switch and range commands only and the parameter is mandatory in the announcements.
It is recommended to limit the number of stack to < 100 to keep a SK user friendly. For higher numbers the selector should be split: <des_stack> with MUL: see below.
announcement example: 2;os;2,stack;0,off;1,on
command example: 0x020x010x01 this will switch the 2nd switch.
More complex examples are shown in descriptions below.

avoid doubling of descriptions (ALPHA)
The following can be used for defining allowed characters of a string (example):
Definition:
200;id,DEF;alpha,0x22,(, ,a,,,aa,AA,11,1_0x3to0x20
(→ “, (, space, a, colon, a to z AtoZ, 0 to 9, 0x3 to 0x20)
AA → 1_1_AtoZ; aa → 1_atoz; 11 → 1_0to9
hex and binary notation allowed
“ is used in the announcement as delimiter → 0x22 must be used
alpha is unique label (without colon)
CODING is a special predefined form of ALPHA There is no definition necessary.

usage:
100;om;5,{alpha};10

List of Standardized Command-types

These command-type templates contain a description and format of the properties, where necessary.

Some commands, which have the operating type “o” and “a” for the operating function, the announcement template is identical. The transmitted format of the operating command and the answer of the answer command is identical as well. In these cases the operating type is as “x” in the template, which

must be replaced by “o”, “r” / ”a”, “s” as appropriate in the real announce-list.
“y” must be replaced by “o” or “r”, if there is an operating commands only; “z” by ”a” or “s” for answer commands only.

There are <OPTION>s for all commands. These are given at the end of an announcement.
Following is defined (to be defined in this sequence):

…;<other_OPTION>…	not yet defined
…;0,ALL	for command with operating object r, o, a commands with more than one element: the SK must send a full (operating) command for each positions / dimension and send the command without parameter for an answer command. The device send a full answer for each positions / dimension. This can simplify storage of bits by using the r command.
…;<ty>,METER,<pa>...	<pa> is a mandatory value in ms. Valid for answer commands am, an, aa and ab with one element only. <ty> depends on <pa>. The SK should update value with this time interval. Used for metering e.g., if the device is not sending infos by itself. For commands with more than one element a additional command as “ext” command for the specific parameter can be used. If the time is very short and time to get the data is long, the SK must be able to handle this. This may be critical, if a device has more than one METER commands.
…;<ty>,CHAPTER,<sm>;....	<sm> is a readable string; <ty> is the stringlength. Submenus are separated by a “_”.

Some notes about CHAPTER
CHAPTER is an info for SK only, for sorting the commands to menus; there are no transmitted data.
Using a GUI the number of controls of a page should be limited, so that the elements can be clearly represented.
Take into account, that one command may have more than one controls: additional selectors for memory command eg or multiple controls for multiple dimensional range commands.
14,CHAPTER,ADMINISTRATION is used for some reserved token, but can be used for other commands as well.

Meta-commands

Meta commands are used to control the system, they do not initiate actions.

At start all commands are enabled. To disable commands rules should be used.

announce: <c>;ixx;... ix commands have the same syntax as xx commands. They are for information only: The CR will not forward them.
command: -
answer / info: -

announce: <c>;id;DEF;xx,{<s>} definition of a string. xx is a name; unique within the announce-list.
The CR may rename the definition in case of conflict eith different devices.
The CR may insert the definition into the commands before distributing.

example: 200;id;DEF;alpha,11,1_0x20to0x24,),0x19 Definition of an alphabet for restriction of allowed characters,
reserved values are AA for upper case letters
aa for lower case letters
11 for figures

announce: <c>;iz<,label> no command
This command can be used as a placeholder or when an announcement is needed only (as for the reserved 0xxxfa command)

Switches:
one dimension for normal switches
two dimensional for crosspoint or similar

OPTION for all switches:
...;<ty>,DIMENSION,<number_of_row>,<des>;DIMENSION,<number_of_cols>,<des>;.. not supported anymore; use stacks instead

announce: <c>;xr[,<des_c>];number_of_stacks[,<des_stacks>];0[,< label>]...;n[,< label>][;<OPTION>...]
reset (0) or set (1) position (number from 0 to n)

Operate command or answer / info: <c>0|1 simple switch (pos0 in announcement only) 0 must be

		omitted, 1 stack
	<c><n>0|1	reset or set set position <n>, 1 stack
	<c><m><n>0|1	reset or set set position <n> of stack <m>
answer command:	<c>	simple switch (one position in announcement only)
	<c><n>	read reset or set of position <n> 1 stack
	<c><m><n>	read reset or set of position <n> more stacks
		with the ALL OPTION <n> must be omitted

announce: <c>;xs[,<des_c>];number_of_stacks[,<des_stacks>];0[,<label>]…;n[,< label>][;<OPTION>...]
set one out of n positions active, reset others; 2 or more positions required.

Operate command or answer / info:	<c><n>	set position <n>, reset the others, <n> needed always, 1 stack
answer command:	<c><m>	with more than one stack

announce: <c>;zt[,<des_c>];number_of_stacks[,<des_stacks>];0[,<label>];..n[< label>][;<OPTION>…]
answer command; the device toggles to the next position and answer the new position
one active at a time, can be an extension of an os command.
After position n position 0 is set.
With one position: switch on – off – on …

Operate/answer command :	<c>	toggling for one stack
answer / info:	<c><n>	position <n> is active now

announce: <c>;yu[,<des_c>];number_of_stacks[,<des_stacks>];0[,<label>];...;n[,<label>]<;OPTION>...]
set one of n positions momentary active, position 0 is idle always
position 0 and position 1 required as minimum.

command:	<c>	push button switch, no parameter, if there is pos0 and pos1 only
	<c><n>	set position n momentarily

<c><m><n> for multiple stack

This command may change the internal state of the device. The device must inform the SK by appropriate info about this. Or this may be defined by rules.

Range controlled functions

one dimensional form for potentiometer, frequency generator,...
two dimensional form for joysticks...
three dimensional form for robotics...
The semantic meaning of the range may be very different. A location range (somewhere on the earth e.g.) mean, that a stop stops at a specific location. The stepwise movement is a change in distance: it is some speed.
For a range of speed a stop means a velocity of 0 and a step-wise change is an acceleration. So the application of these controls can be very different.
<number_of_values> result in "0" based binary values with n bytes. A number_of_values = 10 means transmitted values from 0 to 9. The real values are given in <des_range>; see "More about Descriptions" below.
There may be more than one announce line working on the same real object. Eg, with one line you change the location, with the next line the speed.
When you set a moving object to a location in regular time intervals you will have a loop (after passing the end: start at beginning again).
Sequence of parameters must not be changed.

announce: <c>;xp[,<des_c>];number_of_stacks[,<des_stacks];number_of_valuesx[,<des_op_ap_oo>];<sequence>[,<label>];unitx,[<label>] [;number_of_valuesy,..]

go to value ("0" based). number_of_values is of type <n>
Example (User display is in steps of 10):
2;op;1;50001,{10_3500000to3800000,10_7000000to7200000};lin;Hz
for <sequence> see below
more number_of_values blocks means more than one dimension

Operate command or answer / info:	<c><m><n>	go to / read (1 dimensional) n of mth stack
	<c><m><n><n>	2 dimensional of mth stack
	<c><n>	go to n for one stack; 0 for one stack only must be omitted
answer command:	<c><m>	mth stack
	<c>	one stack

number_of_valuesx..is the maximum of the transmitted value!

More details about <ddes_op_ap_oo > see below shown in: More about Descriptions
Real values the valuex.. must match the highest possible value to be transmitted
For <sequence> the following is defined:
lin linear numbering

announce: <c>;yo,[ext<c>],[<des>];number_of_stacks[,<des_stacks>];number_of_steps,<label>;step_size[,<*des_op_ap_oo*>]
[;step_time[,<*des_op_ap_oo*>][;step_time-unit,<label>[;a,<label>],[4,LOOP|6,LIMIT]
move position stepwise; works only as extension of an op command

Some explanation:
Multiple oo lines for the same op command are allowed.
oo announcements should directly follow the op (or ap) announcements,
The optional CHAPTER should be omitted for the oo announcements: the chapter is identical with the op command.
number_of_stacks[,<des_stack>] and number of dimensions must be identical to the op command; number_of_steps and stepsize in announcement must match (not exceed) the op command.
CR do not check, whether oo and op matches.

Real range for step_time are given in <des_op_ap_oo>. The unit of step_time is for information of the SK and not transmitted
With the optional (one byte) field something like a,0,down is possible. This is information for the SK. Use two commands, if up and down is required.
The last part is optional information of SK only: 4,LOOP means, that the device will loop the values; with 6,LIMIT it will stop at limits. Default is LOOP.
Values of number, step_size and step_time set to “0” in the announcement for a dimension mean, that there is no function with this dimenension.
A number_of_steps = 0 in the announcement and step_size and or step_time != 0 for a dimensions mean fixed value. In fact, this mean: move to a default value. The default values can be described by <label>. Any (allowed) command <c>xxxxxx has the same result. By this, one of many dimensions can be set to default.
When the “oo” command is used with more than one dimension, the command must have have values for all dimensions always.

Example:
1;op,range;1;10;lin;-;5,CHAPTER,range 1 stacks, 1 dimension
2;oo,range;1;10;3;1;10;sec;a,down;5,CHAPTER,range
3;oo,range;1;1;1;10;sec;5,CHAPTER,range
4;op,range;2;10;lin;-;20;lin;-; 2 stacks, 2 dimensions

5;oo,range;2;10;2;3;sec;12;2;4;sec;6,LIMIT

command:	0x020x000x000x00	stops (1 dimension ranges < 256) immediately
	0x030x000x000x00	goto default (3), all parameters in the announcement are 0
	0x050x000x000x000x000x020x010x02	stops 1st dimension; start 2nd dimension

Command has a data destination like memory, data channel or text field

The description of memory was changed in 202305 to be in line with stacks.
It is not intended to replace data streams with these commands, but it is not defined clearly, where simple data transmission ends and streaming is starting. So some of these commands can be used for simple data streams as well.
There is no “undefined” value, when reading a memory. For details see [11].
The data content of the memory has the full range of the defined properties, if not restricted by <des_range> The position of the memory cell is “0” based.
The number of elements of a memory are defined in the announcement. Any number is allowed, but there are some recommendation for restriction.

A device may have some internal memory, which stores the state of another command or memory and is not readable directly. In case of changes the device must send appropriate infos.

When reading / writing a memory with the type string, the access is done with the complete string with stringlength. There is no access to positions within the string.

announce:	<c>;xm[<,des_c>]; <ty>[,<des_type>];m_positions[,<des_memory>	write a memory element of type <ty> with optional restrictions as given in <des_range> of <ty> dimensions of the memory are given by <des_memory> for <des_memory> see below m must be transmitted always.
operate command or answer / info:	<c><m><data>	write to / answer for position n
answer command:	<c><m>	read position <n> (“0” based)

announce: <c>;xn,[,des_c]; <ty>[,<des_type>];m_positions[,<des_memory>;n_elements[,<label>]

sequential access of i (i <= m)elements for memory starting with position j. (j < = n_positions)
may can work as extension to a xm command
If end of memory is reached next element is written to m=0
There is no <des_range> for n !!!
m and n must be transmitted always.
For use by humans a small figure of n_elements is recommended

operate command or answer / info: <c><m><n><data> — write n elements to memory, starting at position m (“0” based)

answer command: <c><m><n> — read n elements from memory, start at position m (“0” based)

announce: <c>;xf[,<des_c>];<ty>[,<des_type>];<m>[,<label>]

FIFO, stack, stream or similar functions, not more than m elements should be sent / read. There is no <des_range> for m!
For usage for humans it is recommended to limit m to 100!

operate command or answer / info: <c>2<data><data> — 2 elements are sent / read
answer command: <c><n> — n elements are expected

announce: <c>;xa[,<des_c>];<ty>[,<des_type>][<ty>[,<des_type>]]…

array of different elements of type <ty>
For more than 255 elements use additional command.
for one element only n is not transmitted

operate command or answer / info: <c><m><data> — write one element to (read from) array, mth position (“0” based)

answer command: <c><m> — m is the mth element (“0” based)

announce: <c>;xb[,<des_c>];<ty>[,<des_type>][;<ty>[,<des_type>]]...

sequential access mode for array
can work as extension to a oa command
for one element only n and m is not transmitted

operate command or answer / info:	<c><m><n><data>	write n elements to memory, starting at position m (“0” based) m and <n> must not exceed the number of elements n = 0 means no data transfer If end of memory is reached next element is written to n=0.
answer command:	<c><m><n>	read n elements from memory, start at position m (“0” based)

Hint for xa and xb command:
These are very powerful commands and the number of memory elements is not restricted by the protocol.
A high number of elements may result in a SK, which is difficult to handle. With the existing Web SK up to 100 elements are possible, but not more than 10 are recommended.
Depending on the (Web) SK the handling of oa / aa and ob /ab command may be very similar.
Take into account, that input of non valid data may result in no data sent. So manual input of many different datatypes may be not easy.

More about Descriptions

General format: [label][<,des_range>|<des_memory>|...][,additional]
The description should help the SK to display the real parameters and to provide the correct user readable labeling. If there is no description the SK will use the full range as defined by the property type and use this as labeling.
It is used as well for restriction of characters / values for data of memory.

Description of OPTIONs:
[additional] only; used for comments

Description of command-type <des_c>:
[as|ext]|[,label]

Descriptions for stacks and memory positions <des_stacks>, <des_memory_position>:
with one selector / dimension:
[,<label>][,<des_range>]

with with more than one selector / dimension::
{n1,[,[label][,<des_range>][MULn2[,<label>][,<des_range>]… [ADDn_add,[,label][,<des_range>]}

number_of_positions is the possible number of combinations of n1, n2….
If the stack / memory element number is high, it is recommended that the SK arrange this in rows and columns (and more).
The SK can use separate selectors as defined by <des_memory> fields separated by MULs and ADD.
The number of MULs (selectors) is not limited, but MULs within the MULs are not supported.
Only one ADD should be at the end (if any). ADD requires MUL.
ADD can be used if the row and cols are used, and there is a limited number of additional positions. If the selected value is "0", rows and cols are used only. Otherwise the maxrow * maxcol * ...+ ADD is transmitted independent of the selected row and col values.
The <des_range> for ADD should start with a meaningful name as label. It should denote, that the value is calculated using the other values.
Resulting values must be unique.
Note, that there is no "," around MUL and ADD. A ","before MUL or ADD means an empy label
Examples:

2;op;100,test;255;lin;-	simple stack selector with name test for values from 0 to 99
2:op;4,{t1,t2,t3,t4};255;lin;-	simple stack selector without name for values t1, t2, t3, t4
2:op;4,{4,test,{t1,t2,t3,t4}};lin;-	simple stack selector with name test for values t1, t2, t3, t4

2;op;10004,{100,colMUL100,rowADD5,tx,{use_row_col,t1,t2,t3,t4}};255;lin;-
is a matix (stackselector from 0 to 99 (named col), a stackselector from 0 to 99 (named row)) and an additional stacksselector t1 – t4, named tx.
The transmitted value is col * max_row + row for tx = 0 and max_col * max_row + tx for tx > 0.

2;op;15,{5,{a,b,c,d,e}MUL3,test,{1,2,3}};255;lin;-	is matix (stackselector a to e (no name), a stackselector 0 to 2 (named test)
2;op;15,{5,{1_2_to 4,a,b}}MUL3,test,{1,2,3}};255;lin;-	is matix (stackselector 2,3,4,a,b (no name), a stackselector 0 to 2 (named test)

Descriptions for range commands <des_op_ap_oo>:
same as <des_stacks> for one selector, but for each dimension (non MUL or ADD)
<number_of_positions>[,<label>][,<des_range>]
Examples:

1;op;1;8,{1_1to5,10_20to40};lin;-	will result in 8 values: 1 2 3 4 5 10 30 40 transmitted by 0 to 7.
1;op;1;4,name ;lin;-	transmitted as 0 to 3, display spacing is 1, displayed as 0 1 2 3, name is the label
1;op;1;999,name1,{1_1to999};lin;-	transmitted as 0 to 998, displayed as 1 to 999; name1 is the label, additional: something else

Descriptions for switches (os, as, or, ar, ou, at):
Usually any button of switches has one label. {des_range} is not allowed.

Descriptions for type of memory commands (om, am, om, an, of, af, oa, aa, ob, ab) <des_type>:

For all memory data:
[<label>][,default_value][,<des_range>] <default_value> must not contain „{„ . <default_value require <label>.

Details for <label>:
Uppercase names should be used for reserved words (as labels) only.
Labels must not contain the characters “.” “,” “;” “{ “}” ”_”, and “to”; lowercase preferred
For labeling english is preferred. If other languages are required, the translation can be done by the SK..

Details of <des_range>
Format: {[fixed_value][,“step-distance“_“from“to“to“]…}
Mixed combinations of ranges and fixed values are allowed, but values must be unique.
Fixed values may be nuemric and alpha, ranges may be numeric or alpha. Alpha may be something like 1_atoz, 2BtoY but not 1_atoZ or 1_ato9.
Use more than one range instead or hex values.

- For <des_type> of data of memory commands:
<des_type> is used for restriction of characters / values to be used only. Other characters / values are ignored. There is no translation for strings.
Not valid numbers of numeric types or characters of strings are not modified but ignored.
There is no translation of displayed values to transmitted values.
Examples:

1;om;2,{a,c,d};10	string of 2 characters (max); this limits the allowed characters to a, c, d.
1;om;2,{1_atoz,1_AtoZ};10	string of 2 characters (max); letters only allowed
1;om;3,{alpha};10	similar as above, use the predefined alphabet alpha; see meta-commands above.
1;om;b,{0,1,3};10	byte; displayed as ASCII character of 0, 1, 3 and transmitted as 0,1,3 Other characters will be ignored.
1;om;w,{1_0to100};10	word (2 bytes); 1 to 100 allowed only
1;om;n, %,{5_0to100};10	will allow percent values with 5% steps from 0 to 100%

- For others (range commands, stacks, position of memory commands):
<des_range> is used, if real values and transmitted values are different, and / or the range of transmitted values is limited.
The transmitted values have the range 0 to x (inclusive)
The transmitted value is the sequence-number of the real value (number or alphanumeric). Other real values are ignored.
The ranges in the description are non-overlapping values usually in ascending order. Values must be unique.

1;om;a;3,name,{1,2,3},comment	position transmitted as 0,1,2, displayed as 1, 2, 3,(3 element memory), comment is ignored by SK
1;om;a;3,{a,b,d}	position transmitted as0, 1, 2 displayed as a, b, d. (3 element memory)
1;om;a;21,{0.1_1.0to3.0}	position transmitted as 0 to 20, displayed as 1.0, 1,1... 3.0
1;om;a;999,name,{1_1to990,2_1000to1008,a,b}	position transmitted as 0 to 998, displayed as 1 to 990, 1000 to 1006 (step 2), a, b
1;om;b{2_atoe};10	data for input restricted to a, c, e

CODING:
If for memory data the translation of transmitted values to displayed values cannot be described in a simple way, CODING can be used. The device must not send values out of range.
CODING is an information for the SK, if complex translation from display data to transmitted data is necessary (as for time).the SK decides about the displayed format, and do the translation.
Transmitted values are numeric always.
CODING (as ALPHA) is a special form of <des_range>, but the definition is fixed.
Format:
<ty>[,label][,{<codingname>}]
The following reserved names for <codingname> are defined now (more may follow):

UNIXTIME8:	t[,label],{UNIXTIME8}	8byte UNIX-time
UNIXTIME4:	w[,label],{UNIXTIME4}	4byte UNIX-time
DAYSEC	k[,label],{DAYSEC}	0 to 86400
DAYMON	n[,labe].{DAYMON}	0 to 28|29|30|31
YEARDAY	w[,label],{YEARDAY}	0 to 364|365
YEARDAY0	L[,label],{YEARDAY0}	0 to 4294967295 day from 1.1.00;
YEARNA	e[,label],{YEARNA}	-2147483648to2147483647; nearly all years before and after year “00”
LOG	<ty>[,label],{LOG,…}	logarithmic display, <ty> depend on {...}

The status of bits of a byte can be described by the r command.

<codingname> is used for <sequence> of op /ap commands as well. The number of the maximum transmitted value must match the <codingname>.

Errors and Error Handling

The CR is programmed to be in line with a set of specifications defined by <SPEC_VERSION>. See [12]
It must be downward compatible with earlier version in the same branch. It depends on the CR, how announcements / commands of other

<SPEC_VERSIONS> are handled.
In general it is assumed, that the MYC protocol is used by computers only (M2M), which are correctly programmed.
So only valid commands with valid parameters are send by the SK and all rules are fulfilled (and that the device is programmed correctly).
The CR will possibly not do a complete check of the announce-lines but ignore them, if it detects errors.
All devices checks all incoming commands and parameters for validity. They will receive all bytes as given by the announcement and by the parameters but ignore them if parameters are wrong. The devices will not send an error message as a response of a wrong command syntax or ignored rules, but will send at least the last error on request.
The CR will not check for limitation given in the descriptions and ignored rules.
That means, that the SK must check operated commands by using an answer command.
It is possible, that the SK will not get an answer, if the answer command is not valid, as given by rules.
It should also check the status of the devices in reasonable time intervals.
A logic device knows all rules and can do the complete check. The logic device could inform the SK about the wrong command. But this procedure is not yet decided.
Any command with correct syntax and parameters within the limits will be done or answered by a FU. In some cases a value may be not valid at that time. In these cases the FU will send a non valid command-token.
Slow devices can ask for a longer wait time for the CR to send the answer (for I2C eg).
The info command will be used by the CR to check if a device is ready after startup.
Some device are very complex with complex and sometimes hidden rules, and some rules are missing. So that device cannot block the wrong command, but detect some error.
In these cases the device may send back an info after an operate or answer command to denote, that the command was wrong:
<not_valid_token>:
The preferred not_valid_token is 0xxxEF
The SK must understand this.
This simplifies the communication with the SK sending a not valid answer command ans avoid to wait for a timeout.

Copyright

The usage of this document is on own risk, there is no warranty.

Reference

[1] https://dk1ri.de/myc/MYC.pdf (german)
[2] https://dk1ri.de/myc/MYC.en.pdf
[3] https://dk1ri.de/myc/Description.txt or https://dk1ri.de/myc/Description.pdf
[4] https://dk1ri.de/myc/commands.txt or https://dk1ri.de/myc/commands.pdf
[5] https://dk1ri.de/myc/Reserved_tokens.txt or https://dk1ri.de/myc/Reserved_tokens.pdf
[6] https://dk1ri.de/myc/Rules.txt or https://dk1ri.de/myc/Rules.pdf
[7] https://dk1ri.de/myc/commandrouter.txt or https://dk1ri.de/myc/commandrouter.pdf
[8] https://dk1ri.de/myc/Rules_device.txt or https://dk1ri.de/myc/Rules_device.pdf
[9] https://dk1ri.de/myc/skin.txt or https://dk1ri.de/myc/skin.pdf
[10] https://dk1ri.de/myc/logicdevice.txt or https://dk1ri.de/myc/logicdevice.pdf
[11] https://dk1ri.de/myc/Definitions.txt or https://dk1ri.de/myc/Definitions.pdf
[12] https://dk1ri.de/myc/spec_version.txt or https://dk1ri.de/myc/spec_version.pdf
[13] https://dk1ri.de/myc/webserver.txt or https://dk1ri.de/myc/webserver.pdf
[14] https://dk1ri.de/myc/ki.txt or https://dk1ri.de/myc/ki.pdf
[15] https://dk1ri.de/myc/communication.txt or https://dk1ri.de/myc/communication.pdf
[16] https://dk1ri.de/myc/Security.txt or https://dk1ri.de/myc/Security.pdf