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 V02.12.12 20230403
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 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>
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.
The CR concatenate the announcements of the devices to a full announce-list.
The I-line 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 them in all cases.

Some operating commands may have an influence on the status of other commands. This is handled by rules as well.
Rules lines start with "R" and are included in the full list.
“Q” rules are sent by the RU to the command-router and are not included in the full announcement list. 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.

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” Depending on the SK the SK will not use these commands.
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. If large numbers of stacks used, the SK must initiate a mass of data at start otherwise. So the SK may not support this and will ask the state of the actual stack only.
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 operating command means, the devices want another device to operate. The result of these commands is defined by rules. The r command denote what they can or want to send. The s command can be used by the LD to check a status. For details see [9]. These commands are not part of the full announce-list.
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, 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 computer 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 prefered 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 and time (and other special) formats 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 and may be higher but not lower.
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 different 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 a set of commands. If a command restores a stored status to the actual one, the SK must be informed about the state of the changed commands. This can be done by rules – and the CR will inform the SK- or the FU inform the SK directly. This is preferred.
There are no answers for operating commands.
Infos and answers use the same format.
Infos / answers start with the corresponding data of the answer commnad with added data.
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 as a text-element. Depending on the implementation some others cannot be used as well.
In this case the ASCII notation 0xx (xx 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 must follow the operate announcement and must be simplified. 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. This answer command directly follow the operating command.

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

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.
2,op,VFOA;1;30000 ,{3500000 to 3800000};lin;Hz
3,oo,ext2;1;255;25;s;50;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.
Usually 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 and the parameter is mandatory in the announcements.
The number_of_stack parameter may have a description.
It is recommended to limit the number of stack to < 100 to keep a SK user friendly. For higher numbers the selector shoud be splitted: see <des_stack with MUL 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
this can be used (example):

Definition:
200:id;1;DEF_ALPHA1,AA,aa,0x21to0x2
DEF_ALPHA is unique (uppercase) name, It is recommended to use one stack only.

usage:
100;om;1;35,{DEF_ALPHA}
The commandrouter will replace the label by the content.
For details see: Descriptions

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

…;<ty>,METER,<pa>... <pa> is a value in ms. Valid for answer commands only. The SK should update the value

after that time. Used for metering info, if the device is not sending infos by itself.

…;<ty>,CHAPTER,<sm>;.... <sm> is a readable string; put at the end of an announcement line.
Submenus are separated by a "_". <ty> must be the stringlength (without usage)

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; Meta-commands are not longer used for this.

announce: <c>;ix;... 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;1;DEF_xx,<s> definition of a string. xx is a name; unique within the announce-list.
The CR will insert the definition in the commands before distributing.

example: 200;id;1;DEF_ALPHA,11,0x20to0x24,),0x19," Definition of an alphabet for restriction of allowed characters,
0xxx as hex representation is allowed; must be used for ranges, ";" , "," and """.
reserved values are AA for upper case letters
aa for lower case letters
11 for figures

announce: <c>;iz<,des> 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>;..
<des> is something like x y z
info for SK only to display in more than one dimension.
must be at the end of the announcement, no transmitted data.
<number_of_row> are the number of rows, columns...
Number of positions must match the product of the dimensions
<ty> is a stringlength (not used)
if used, <des pos> should not be used.

announce: <c>;xr[,<des>];number_of_stacks[,<des_stack>];0[,<des pos0>]...;n[,<des posn>][;<OPTION>...]
reset (0) or set (1) posm (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 (pos0 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

announce: <c>;xs[,<des>];number_of_stacks[,<des>];0[,<des pos0>]…;n[,<des posn>][;<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 stacks

announce: <c>;zt[,<des>];number_of_stacks[,<des_stack>];0[,<des>];..n[<des posn>][;<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 a os command.
After posn pos0 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>];number_of_stacks[,<des_stack>];0[,<des>];...;n[,<des>];pos0[,<des>];...;posn[,<des>][;<OPTION>...]
set one of n positions momentary active, pos0 is idle always
pos0 and pos1 required as minimum.
There is no answer command

command: <c> push button switch, no parameter, if there is pos0 and pos1 only
<c><n> set posn momentarily
<c><n><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 10 values from 0 to 9. The real values are given in <des>; 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.
Multiple identical groups of these function can be addressed by the number_of-stacks parameter.

announce:

<c>;xp[,<des>]...;number_of_stacks[,<des_stack>];number_of_valuesx[,<des>];<sequence>[,<des>];unitx,<des>];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

The number_of_stacks > 1 is used if more one identical range controls are available and should be addressed individually. number_of_stacks is “0-based, and not transmitted if number_of_stacks = 1.

for <sequence> the following is defined, more details about real ranges are shown in <des>
number_of_valuesx..is the maximum of the transmitted value!
The real data are calculated by the SK using sequence and {ranges} of <des>. If the sequence is not lin, {ranges are mandatory.
Real values the valuex.. must match the highest possible value to be transmitted

lin:	linear numbering
log:	logarithmic sequence
date:	In the description the format yyyy[mm[dd]] must be used.
time:	In the description the format hh[mm[ss]] must be used

Datetime: In the description the format: yyyymmddhh[mm[ss]] must be used.
degree: In the description the format dd[d]mm[.mm..] or ddmmss must be used

announce: <c>;yo,ext<c>,[<des>];number_of_stacks[,<des_stack>];number_of_steps,<des>;step_size[,<des>];step_time[,<des>];step_time-unit,<des>[;a,xx],[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, so the extxx notation should not be used.
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 the op command. So these values must not exceed the valuex of the op announcement.
CR do not check, whether oo and op matches.
Number_of_steps and step_size has a label in <des> only (no translation to real values)
A “0” in the step_size announcement will not allow movement in this dimension.
Real range for step_time are given in <des>. 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.
The command is: <c><number_of_steps><step_size><steptime> or <c><number_of_stacks><number_of_steps><step_size><step_time>
Value of 0 for all parameters an announcement in a dimensions mean fixed value as per <des> of number_of_steps. In fact, this mean: move to a default value. Any command <c>xxxxxx has the same result.
A <c>0x000x000x00” at transmission for all values in a dimension will stop this dimension (no movement).
When the “oo” command is used with more than one dimension, this may have unwanted results. A command must have have values for all dimensions.

Example:
1;op,range;1;10;lin;-;5,CHAPTER;range — 1 stacks, 1 dimension
2;oo,range;1;10;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;-; — 10 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

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 content of the memory has the full range of the defined properties, if not restricted by <des> 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. The length of the transmitted element-number depend on this. If a memory has 300 cell e.g., the 5th cell will be addressed by 0x00x04.

A device may have some internal memory, which stores the state of another command or memory and is not readable directly. To write and restore to these memories two ou commands can be used. Because the restore command may change the state of device, the device must send appropriate infos.

When accessing a memory with the type string, the access is done with the complete string. There is no access to positions within the string.
The answer of a read command will provide the complete string with stringlength.

It is recommended, to limit the number of rows. Cols .. to less the 100. otherwise the HI may be less comfortable. Use additional dimensions instead.

announce: <c>;xm[<,des>]; <ty>[,<destype>];m_cols[,<desranges>];[<m_rows[,<desranges>]...[,n,ADD,<des>]...

write a memory element of type <ty> with optional restrictions
as given in <des> of <ty>
m_cols, … is the number of columns .. the device can handle
<des> may be the name of row/col,
any dimension is possible;
a optional [;n,ADD] (at the end): n additional positions
The transmitted position z of an element with ADD=0 is
z =…n_x *m_col*m_row + n_col* m_row + n_row

0<=nx<mx !
With ADD!=0: n additional positions:
z= mx * m_col * m_row + n
fixed type <ty>
The length of the transmitted position z result from
m_row * m_col *...

operate command or answer / info: <c><n><data> write to / answer for position n
answer command: <c><n> read position <n> (“0” based)
answer: <c><n><data>

announce: <c>;xn,[,ext<c>]...[,<des>]; <ty>[,<destype>],<n>[,<des>];m_cols[,<desranges>];[<m_row[,<desranges>]..
sequential access of n elements for memory starting
with m_cols / m_rows...
may work as extension to a xm command
If end of memory is reached next element is written to n=0
one element allowed, but make no sense
string s are transmitted with the individual string-length
With the announcement of one dimension (m_cols=0) it is a
FIFO, stack or stream.

operate command or answer / info: <c><n><m><data> write n elements to memory, starting at position m (“0” based)
<c><n>0<data> write n elements to FIFO, stack...
answer command: <c><n><m> read n elements from memory, start at position m (“0” based)

announce: <c>;xf[,<des>]...[,<des>];<ty>[,<destype>];<m>,<des> FIFO, stack, stream or similar functions
no longer supported use xn instead

announce: <c>;xa[,<des>]...[,<destype>];<ty>[,<destype>]...[;<ty>[,<destype>]] array of different elements of types <ty> with optional
restrictions as given in <des>;
For more than 255 elements use additional command.

operate command or answer / info: <c><n><data> write one element to (read from) array, nth position
(“0” based)
<n> should be omitted for 1 element array.

	OPTION syntax for Individualization command, see [6]
answer command: <c><n>	n is the nth element (“0” based)
announce: <c>;xb[,<des>]...[,<destype>];<ty>[,<destype>]...[;<ty>[,<destype>]	sequential access mode for array can work as extension to a oa command one element allowed, but make no sense
operate command or answer / info: <c><n><m><data>	write m elements to memory, starting at position n (“0” based) m and <n> must not exceed the number of elements m = 0 means no data transfer If end of memory is reached next element is written to n=0.
answer command: <c><n><m>	read m elements from memory, start at position n (“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 10 elements is a good choice.
Depending on the (Web) SK the handling of oa / aa and ob /ab command may be very similar.

More about Descriptions

The description should help the SK to find the correct 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 no (or default) labeling.
{ranges} will be used to define the real range of allowed values.
Additional description may follow. This is not used now.
Not all parts are allowed always:
[label][<,desranges>][;additional] or
[label][,{ranges}][;additional]
For labeling english is preferred. If other languages are required, the translation can be done by the SK.
Labels must not contain both “_ and “to” : they must be distinguishable from the x_ytoz notation.
For <desranges> and < des_stack> see below.

Description of OPTIONs:

[additional] only; used for comments

Description of command-type for all commands:
[as|ext][,label]
Label must not start with ”as” or “ext”. This is a label reserved for command optimization (see above)..

Descriptions for switches
Usually any button of switches has a label. {ranges} are not allowed.

Descriptions for range commands (op / op):
max[,label][,<desranges>][,additional] (for each dimension)
The count within <desranges> must match the count leading the <des>:

8,{1_1to5,10_20to40}	will result in 8 values: 1 2 3 4 5 10 30 40 transmitted by 0 to 7.

If the display ranges are not natural figures (as in 0.1_ 0,1to 0,5) a range must be given in all cases.

4,name	transmitted as 0 to 3, display spacing is 1, displayed as 0 1 2 3, name is the label
999,name1,{1_1to999}	transmitted as 0 to 998, displayed as 1 to 999; name1 is the label
1000,{0.1_0.0to99.9}	transmitted as 0 to 999, displayed as 0.0 to 99.9
11,{1_0to10}	transmitted as 0 to 10, displayed as 0 to 10
1003,{1_0to999,10_200to220}	transmitted as 0 to 1002 (2 byte), displayed as 0 to 999, 200 210 220,
11,{1_0to8,a,b}	transmitted as 0 to 10, displayed as 0 to 8, a, b

Descriptions for memory type commands:
There are two optional kinds of descriptions:

For the memory positions of om, on, am, an commands:
[label][,<desranges>][;additional] for position of om commands and start of on commands.
[label] for number of elements of on commands.
For ADD: additional [label] is allowed; the following {…} is mandatory:
n,ADD,[label],{<destrange>}
<desranges> should match the value leading the <des>.

For all memory data type <ty> see <destype>

Details for <des_stack>
If the stack number is high, it is recommended that the SK arrange the stack in rows and columns (and more).
The SK can use separate selectors as defined by <des_stack> 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 can be used if the row and cols are used, and there is a limited number of additional stacks. If the value is "0", rows and cols are used only and this value should have an appropriate name.. Otherwise the maxrow * maxcol + ADD is transmitted independent of the selected row and col values.
ADD requires MUL.
Format of stack:
<number_of_stacks>[<des_stack>]
<number_of_stacks>[,<name_of_stack>],[<des_ranges>]
or (if MUL is used)
<number_of_stacks>[,<name_of_stack>],{n1,n1_name,<des_ranges>[MULn2,n2_mane,<desranges>]… [ADDn_ADD,ADD_name,<des_ranges>]}
The des_ranges for ADD should start with a meaningful value as label. It should denote, that the value is calculated using the other values.
Resulting values must be unique.
Note, that is no "," around MUL and ADD

Examples using the range command:
2;op;100,test;255;lin;- simple selector with name test for values from 0 to 99
2:op;4,{t1,t2,t3,t4};255;lin;- simple selector without name for values t1, t2, t3, t4
2:op;4,{4,test,{t1,t2,t3,t4}};255;lin;- simple selector with name test for values t1, t2, t3, t4 (no name for selector)
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 4 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)

Details of <desranges>
<des_ranges> is used only, if real values and transmitted values are different. The transmitted values have the range 0 to x.
The transmitted value is the sequence-number of the displayed value (number or alphanumeric).

mixed combination of ranges and fixed values are allowed. Fixed values may be alphanumeric:
{[fixed_value],[“step-distance“_“from“to“to“]...}
The ranges in the description are usually non-overlapping values usually in ascending order. Values must be unique.
<desranges> are allowed for op/ap commands and step-time of oo commands, position of om, am, on, an commands and within stacks
With version 2.12.09 (202204) the step-distance was added to allow different distances within one subrange:

1;om;a;3,name,{1,2,3},comment	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}	transmitted as a, b, d, displayed as 0, 1, 2. (3 element memory)
1;om;a;21,{0.1_1.0to3.0}	transmitted as 0 to 20, displayed as 1.0, 1,1... 3.0
1;om;a;999,name,{1_1to990,2_1000to1008,a,b}	transmitted as 0 to 998, displayed as 1 to 990, 1000 to 1006 (step 2), a, b
1;om;a;2,{1,2};2,{3,4}	transmitted as 1 to 4, displayed as 1, 2 for row and 3, 4 for col

Details of <destype>
For <ty> of all memory commands: A description of the allowed values. For string type values, it is a restriction of the allowed characters. For others it is a restriction of the range and a translation to different values shown by the SK. CODING is an information for the SK, if complex decoding is necessary (as for time):
[label] is not numeric and not identical to CODING; (lower case preferred)
Format:

[,CODING][,label][,<desranges>]	for strings: type is number (length of string)
[,CODING][,label][,<desranges>]	for others

Examples:

1;om;2,{a,c,d}	string of 2 characters (max); this limits the allowed characters, others will be ignored by the device.
1;om;2,{atoz,AtoZ}	string of 2 characters (max); letters only allowed
1;om;3,{DEF_ALPHA}	similar as above, use the predefined alphabet DEF_ALPHA as well; see meta-commands above.
1;om;b,{a,c,d}	byte; stored and transmitted as 0,1,2 Others will be ignored.
1;om;w,{1,0_0to100.0}	word (2 bytes); 0 to 100.0 only, transmitted as 0 to 1000, other numbers are ignored.
1;om;b,{0.5_0to100},%	will allow percent values with 0,5% steps from 0 to 100%

The device will ignore commands with values out of range and may produce an error message.

For type byte (b): describe the individual bits,. All 8 bits must be given for low and high value starting with MSB_low, MSB_high.…
it cannot be used together with {…}
[,CODING][[label][,0,1,0,1,…..,0,on]
32;oa;1;b,name,0,on,1,1,2,2,3,on,4,on,5,on,6,0,7.on

CODING:
If a translation of transmitted values to displayed valued cannot be described by <des>, CODING can be used. How to implement the decoding is up to the SK
The following is defined now (more may follow):
UNIXTIME: b,4294967295,UNIXTIME
DAYSEC b:86400,DAYSEC
DAYMIN b,1440,DAYMIN
DAYHOUR b,24,DAYHOUR
YEARDAY b,365,YEARDAY
YEARMON b,12,YEARMON

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 then, 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.
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

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