MYC Announcements and Commands

Author: DK1RI
Version V02.12.1 20180107
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 http://dk1ri.de/myc/Definitions.pdf

Announcements

An announcement line uses a readable string in <sm> format. 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 basic announcement contains the description of the device. For details see [5]
The command announcements describe the commands the device will understand.
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. (0x00, 0xxxff)
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 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.

Each 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. This will be defined with the <OPTION>. The first <des> of these properties is the uppercase name of the <OPTION>, with must be defined.

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

Active operating is denoted by “o”, an answer request by “a” as the first letter. If a function can operate and answer, the descriptions in the other announcement should be identical. For example a memory, which can be written and read, must have the same unique description, to allow other devices to identify, that it is the same memory. Alternatively the description can be "as5" eg, where 5 is the command-token of the other command (see also announcement / command optimizing below). The command type may also have the description ext5, which shows the relationship. The CR will resolve the “as” line and add a “ext” notation.

The operation type “i” denotes information only. They have no data traffic.

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 SK to denote what they can or want to send. For details see [9].

The “k” and “l” types are identical to the “o” and “a” type and therefore not mentioned in the list below. The CR will not include them in the full announce-list and also ignore them. They are used during a configuration phase of a device. The device will not have rules for these commands. So special care may be necessary when using them.

Operating object / function

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 allowed 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 must be used in any case.

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 and simple SK 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. 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 announcements depend on the device.
All 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]

General syntax for a command-announcement

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

Commands and properties

The command tokens use numeric n byte format. The shortest format possible is always used.
For properties the same rule apply. For memory content a string and time format 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 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 0xxx240, 0xxx254 and 0xxx255 lines and some others of other devices, but the 0xxx240 command of the CR.
The CR will include 0x00 of other devices into the complete command-list but answer them by itself.
A 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 and simple SK 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 send send not used command-token.

General syntax for a command

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

Info

Some devices can send answers without a corresponding command as info. Not all devices will send infos.
The are no infos for operating commands.
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>]

Announcement / Command optimizing

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

Long announcement lines
If a 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

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 ";" at the end of the first line.

Save space in announcements
If a function can operate and answer, one announcement can 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

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. A 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
4,oo,ext2;1;255;1;s;0,250;b,up fixed stepsize of 250 for scanning, stepsize will be omitted in the command

List of Standardized Command-types

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

There are <OPTION>s for all commands. These are at the end of an announcement.
Following is defined:

…;CHAPTER,<sm>;.... <sm> is a readable string; put at the end of a announcement line
info for SK only, for sorting the commands to menus; no transmitted data.
Submenus are separated by a “_”

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

commands, which denote, which commands are enabled / disabled :
At start all commands are enabled. To disable commands rules should be used. The oc /oc, od /ad commands are not supported anymore.

info commands:

Announce: <c>;ix;... ix commands have the same syntax as ox commands. They are for information only and no data transfer.
command: -
answer / info: -

Other metacommands:

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

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

OPTION for all switches:
...;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

Multiple identical groups of these switch functions can be adressed by the number_of_stacks parameter. This may simplify the HI. If you have a lot of switches selecting 4 values eg, the HI will provide a addition selection with the number of the switch. If the number_of_stacks is 1 this number (0) is omitted.

announce: <c>;or[,<des>]...[,<des>];number_of_stacks;pos0[,<des>]...[;posn,[<des>]][;<OPTION>...]
reset (0) or set (1) posm (number from 0 to n)
1st <des> for reset, 2nd for set, if different.
command: <c>0|1 simple switch (pos0 in announcement only) 0 must be omitted and 1 stack
<c><n>0|1 reset or set set position <n>
<c><m><n>0|1 reset or set set position <n> of stack <m>

announce: <c>;ar[,<des>]...[,<des>];number_of_stacks;pos0[,<des>]...[;posn,[<des>]][;<OPTION>...]
read reset or set of position

command:	<c>	simple switch (pos0 in announcement only)
	<c><n>	read reset or set of position <n>
answer / info:	<c>0|1	
	<c><n>0|1	<n> omitted if one position only
announce:	<c>;os[,<des>]...[,<des>];number_of_stacks;pos0[,<des>];...posn[.<des>][;<OPTION>...]	set one out of n positions active, reset others
command:	<c><n>	set position <n>, reset the others, <n> needed always
announce:	<c>;as[,<des>]...[,<des>];number_of_stacks;pos0[,<des>];...posn[.<des>][;<OPTION>...]	read active position; one out of n
command:	<c>	
answer / info:	<c><n>	position <n> is active
announce:	<c>;ot[,<des>]...[,<des>];number_of_stacks;[pos0[,<des>];...posn[,<des>][;<OPTION>...]	set one of n positions active toggling one position to the next one active at a time, can be an extension of a os command.
command:	<c>	
announce:	<c>;at[,<des>]...[,<des>];number_of_stacks;pos0[,<des>];...posn[.<des>][;<OPTION>...]	read active position; one out of n works as the “as” command.
command:	<c>	
answer / info:	<c><n>	position <n> is active
announce:	<c>;ou[,<des>]...[<,des>];number_of_stacks;pos0[,<des>];...;posn[,<des>][;<OPTION>...]	set one of n positions momentary active, pos0 is idle always
command:	<c>	push button switch, no parameter, if there is pos0 and pos1 only
	<c><n>	

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

announce: <c>;au[,<des>]...[<,des>];number_of_stacks;pos0[,<des>];...;posn[,<des>][;<OPTION>...]
read active position; one out of n
If the internal momentary switching takes a while, the state can be checked by this command. If finished, returns pos0.
Works as the “as” command.

command: <c>
Answer / info <c><n>

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 returns to 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 stepwise change is a acceleration. So the application of these controls can be very different.
<number_of_values> are “0” based binary values with n bytes. The real values are given in <des>; see “More about Descriptions” below
Sequence of parameters must not be changed.
Multiple identical groups of these function can be adressed by the number_of-stacks parameter.

announce: <c>;op[,<des>]...;number_of_stacks;number_of_valuesx,<des>;<sequence>,<des>;unitx,<des>;number_of_valuesy,..
go to value (“0” based) number_of_values is of type <n> also
Example (User display is in steps of 10):
2;op;1;50001,{3500000 to 3800000,7000000 to 7200000};lin;Hz
<sequence> see brlow
more number_of_values blocks means more than one dimension

command: <c><m><n> go to 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

The stacknumber > 1 is used if more one identical range controls are available and should be addressed individually. The <des> show the names of the stacks.

for <sequence> the following is defined:
lin: linear numbering
log: logarithmic sequence
date: date format. In the description the format yyyy[mm[dd]] or yyyy[:mm[:dd]] must be used.
time: time. In the description the format hh[mm[ss]] or hh[:mm[:ss]]must be used.
Datetime: format: yyyymmddhh[mm[ss]] or yyyy:mm:dd:hh[mm[:ss]]

comments: The SK calculate the real distance of one step using the <des>; {3500000 to 3800000,7000000 to 7200000} and number_of_values of 5000002 result in a resolution of 1.
The SK decides, how accurate the display is and may do some rounding.
If the displayed step would vary with ranges, a additional announcement with ext<c> description can be used.

announce: <c>;ap[,<des>]...;number_of_stacks;number_of_valuesx,<des>;lin|log,<des>;unitx,<des>;number_of_valuesy,..
read position

		for details see op command
command:	<c><m>	mth stack m = 1 omitted
answer / info:	<c><n>	one dimensional (n is “0” based) one stack
	<c><n><n><n>	three dimensional one stack
	<c><m><n>	one dimensional mth stack
	<c><n>	0 omitted for 1 stack

announce: <c>;oo,ext<c>,[<des>]...;number_of_stacks;number_of_steps[,<des>];steptime,<des>;steptime-unit;stepsize[,<des>];<ty>,<des>...
move position stepwise

Some explanation:
Can work as extension of a op command ext<c> (dimensions and stepsize in announcement must match the op command)
If not, SK knows nothing about the real ranges. Information announcement ip can be used. CR do not check, whether oo and op / ip matches.
Stepsize is the size of the change of number_of_values of the op command; not related to its <des>. If the real stepsize is modified by <des> of stepsize, the real change depend on both <des>. This may be error-prone and is not recommended.
Real ranges for stepsize are given in <des>. The unit of steptime is for information of the SK and not transmitted
A number_of_steps of “0x00” at transmission in a dimension will stop this dimension.
2 + multiple of 5 parameters means more than one dimension, dimensions must match the dimensions if the related command.

Value of 0 for steptime or stepsize announcement mean fixed value as per <des> and values are not transmitted
With the <ty> field something like a,0,down,1,up is possible. It may be defined here, for which dimension the command should work. If <ty> is z, this value will be not transmitted. String is not supported.

command:	<c>0x000x000x000x00	stops (1 dimension)
	<c>0x010x000x000x000x000x000x000x00	stops (2 dimension)
	<c>0x000x000x00	stops (1 dimension fixed steptime or stepsize)
	<c>0x020x010x080x00	move 2 times,with steptime 1, 8 steps each down one dimension
	<c>0x02	move 2 times, fixed steptime and stepsize and <ty> = z

announce: <c>;oq,ext<c>[,<des>]...[<,des>];<defaultpositionx>[,des]...;<defaultpositiony>[,des]....
go to default; works only as extension of a op command
Example: 2;oq,ext1;0;1 , two dimensions

command: <c>

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] http://dk1ri.de/myc/Definitions.pdf
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 ore 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.

operating commands
announce: <c>;om[<,des>]...[,<des>]; <ty>[,<des>]... [,<des>];m_cols[,<des>]...[,<des>];[<m_row[,<des>]...[,<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;
position z of a element is x+y*(m_rows) +...
fixed type <ty>

command: <c><n><data> write to position n

announce: <c>;am[<,des>]...[,<des>]; <ty>[,<des>]... [,<des>];n_rows[,<des>]...[,<des>];[<n_cols[,<des>]...[,<des>]]...
m_cols, … is the number of columns .. the device can handle
position z of a element is x+y*(m_row-1 +...
fixed type

command: <c><n> read position <n> (“0” based)
answer / info: <c><n><data>

announce: <c>;on,[,ext<c>]...[,<des>]; <ty>[,<des>]... [,<des>];m_cols[,<des>]...[,<des>];[<m_row[,<des>]...[,<des>]]...
sequential access mode for memory
may work as extension to a om command
string s are transmitted with the individual stringlength

command: <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
If end of memory is reached next element is written to n=0
one element allowed, but make no sense

announce: <c>;an[<,des>]...[,<des>]; <ty>[,<des>]... [,<des>];n_rows[,<des>]...[,<des>];[<n_cols[,<des>]...[,<des>]]...
sequential access mode for memory
may work as extension to a am command
one element allowed, but make no sense

command: <c><n><m> read m elements from memory, start at position n (“0” based)
answer / info: <c><n><m><data> If end of memory is reached next element is read from pos1

announce: <c>;of[,<des>]...[,<des>];<ty>[,<des>];<m>,<des> FIFO, stack, stream or similar functions for <ty> with optional
restrictions as given in <des> of <ty>

	<m> is number of data elements, which can be sent with one command each string is transmitted with its own stringlength nothing known about the complete memory size!!!
command: <c><m><data>	FIFO stack.. send m data to stack
announce: <c>;af[,<des>]...[,<des>];<n>,<des>;<ty>[,<des>]	FIFO, stack, stream or similar functions
command: <c><m>	read m elements
answer / info: <c><m><data>	pop stack
announce: <c>;oa[,<des>]...[,<des>];<ty>[,<des>]...[;<ty>[,<des>]]	array types <ty> with optional restrictions as given in <des>; can be mixed but not with command-token
command: <c><n><data>	write to array, nth position (“0” based) <n> must be omitted for 1 element array.
announce: <c>,aa[,<des>]...[,<des>];<ty>[,<des>];<ty>[,<des>]...	array OPTION syntax for Individualization command, see [6]
command: <c><n>	array, n is the nth element (“0” based) <n> must be omitted for 1 element array.
answer / info: <c><n><data>	array, nth element (“0” based)
announce: <c>;ob[,<des>]...[,<des>];<ty>[,<des>]...[;<ty>[,<des>]]	sequential access mode for memory can work as extension to a oa command one element allowed, but make no sense
command: <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 If end of memory is reached next element is written to n=0.
announce: <c>;ab[,<des>]...[,<des>];<ty>[,<des>]...[;<ty>[,<des>]...	sequential access mode for memory can work as extension to a aa command one element allowed, but make no sense
command: <c><n><m>	read m elements from memory, start at position n (“0” based)

answer / info: <c><n><m><data> if end of memory is reached next element is read from n=0

More about Descriptions

The description should help the SK to find the correct parameters and to do the correct user readable labeling. If there is no description the SK will use the full range as defined by the property type and the labeling will be accordingly.
It can also be used to limit the range of allowed values. Hex Values as 0x00 to 0x7f are allowed.
The first "part" within the description is used for this; additional description separated by "," may follow.
The following rules apply for the three types of commands:
Descriptions for switches:
Usually any button of switches has a label. So the first value of the description is used as label. Others, separated by "," may follow.

Descriptions for range commands:
To separate the labeling from the other description bracket {} are used.
The number of values is given by the properties. So this labeling information is necessary only, if the real and transmitted values are different.
If the count within {} is lower then the number of counts, higher count result in the maximum given:

4,{1 to 3}	transmitted as 0 to 3, labels are 1 2 3 3
999,{1 to 999}	transmitted as 0 to 998, labels as 1 to 999
1000,{0.0 to 99.9}	transmitted as 0 to 999, labels as 0.0 to 99.9
4,{1,3,4,5}	transmitted as 0 to 3, labels as 1 3 4 5
401,{0 to 99, 200 to 500}	transmitted as 0 to 400 (2 byte), labels as 0 to 99 , 200 to 500

If the labeling for different ranges has different spacing, the following is used:

10,{5{1 to 5},lin,5{10 to 50},lin}	labels are 1 2 3 4 5 10 20 30 40 50

lin is optional in this case.
A string count "1":

6{1,2,through,10 to 12}	labels 1 2 through 10 11 12

Descriptions for memory type commands:
There are three optional kinds of descriptions.
The first describe the real values of memory positions for the om, on, am, an commands only, if real values and transmitted values are different. This

description come with the row / column property. The transmitted values have the range 0 to x (as given by this type of description), The labeling can use the labels for rows, columns...
This description is enclosed by {}.
If the number of <pa> do not match the <des>, the SK will try some rounding (defined by the SK). This is not recommended.
The following is allowed:

3,{1,2,3}	value 0,1,2 are the real values for 1, 2,3 (3 element memory)
3,{a,b,d}	a, b, d are real values for 0, 1, 2. (3 element memory)
21,{1.0 to 3.0}	real values are 1.0, 1,1... 3.0 for 0 … 20
999,{1 to 999}	real values 1 to 999 for 0 to 998
2,{1,2};2,{3,4}	the label for 0x01 could be “2 3”, for 0x03: “2 4”

The second type of description describe the allowed values of data and come as first description with <ty>. This is a restriction of the full range of <ty>. For string type values, it is a restriction of the allowed characters.
In other words: if you want to avoid a translation, you must use a string type.
This type of description is enclosed by {}

2,{a,c,d}	string of 2 characters; this limits the allowed characters, others will be ignored by the device.
2,{a to z,A to Z}	string of 2 characters; letters only allowed
b,{a,c,d}	byte; stored and transmitted as 0,1,2 Others will be ignored.
w,{0 to 100.0}	word (2 bytes); 0 to 100.0 only, transmitted as 0 to 1000, other numbers are ignored.

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

The third type of description separated by “,” can give additional information for the HI.

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 lowerversion 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.
The CR will possibly not do a complete check of the announcelines but ignore then, if it detects errors.
The CR checks all commands and parameters for validity but not for limitation given in the descriptions. It will ignore those commands and start a new command with the next byte.

That means, that the SK must check for answers and create an error if necessary.
It should also check the status of the devices in reasonable time intervalls.

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 non volid command-tokens.
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.

Copyright

Reference

[1] http://dk1ri.de/myc/MYC.pdf (german)
[2] http://dk1ri.de/myc/MYC_en.pdf
[3] http://dk1ri.de/myc/Description.pdf
[4] http://dk1ri.de/myc/commands.pdf
[5] http://dk1ri.de/myc/Reserved_tokens.pdf
[6] http://dk1ri.de/myc/Rules.pdf
[7] http://dk1ri.de/myc/commandrouter.pdf
[8] http://dk1ri.de/myc/Rules_device.pdf
[9] http://dk1ri.de/myc/skin.pdf
[10] http://dk1ri.de/myc/logicdevice.pdf
[11] http://dk1ri.de/myc/Definitions.pdf
[12] http://dk1ri.de/myc/spec_version.pdf