MYC Command-router

Author: DK1RI
Version V01.13.1 20181022
This paper is published in https://github.com/dk1ri as well

Introduction

This paper describes the command-router and my principles when writing the program. I hope, that the documentation within program is sufficient.
Nevertheless for understanding the program you must understand the MYC principles and the protocol details. Description of working conditions for the command-router are distributed in the other documentation as well.
For more details of the MYC system please check the reference.

The program [13] is written in python and is developed using pycharm under windows

Definitions and formats

see http://dk1ri.de/myc/Definitions.pdf

Some limitations

This is my first python program, so enhancements are necessary.
Coding is done quite basic, so object oriented programming is rarely used.
The program is ***not ready*** and cannot be used as a command-router yet.
The code is not tested on a raspberry yet.

Main missing topics

see _still_missing in the program directory

Programming principles

The program uses polling. After initialization following functions are polled:
time-dependent task
read input devices (keyboard or network input)
check SK-input-buffer
send to LD
check LD-input-buffer
transmit command to device
read devices
check device-inputbuffer (answer or info)
send answers to SK-buffer
output to SK

The program should run on a raspberry pi with limited cpu resources.
Because the CR must analyze any incoming byte, many lists and arrays are created during

initialization. They are defined as v_xxx.py global variables. I am not sure, whether the used data structure is the most effective (in terms of speed). So enhancements may be necessary.
Sometimes the program lines are quite long and subarrays are hard to read. So I sometimes introduces variables like temp with a “lifetime” over a few lines. This enhances the readability, but may reduce speed. I also used those variables, when they are needed a few times, because a call of a variable may be faster than an element of a subarray.
There are 3 input parser subprograms: command_handling -analyzing the SK-buffer for commands -, ld_command_handling - analyzing the LD buffer - and device_handling – analyzing the device buffer. These subprograms use the same parser data_handling

Usage

The routing kernel is working for all commands and answers defined now.
Nevertheless the usage of the program now is suitable for tests only.
Input can be done by keyboard or via telnet by sending the ASCII numbers 0... 255 followed by a space.
Data are sent to the SK interface. To send data to the device interface type d<SPACE> and S<SPACE> back to the SK interface.
A file can be used to send multiple commands.
Data, which should be send to the device, are send to the console.

Testing

If the variable v_cr_params.test_mode is set to 1, log entries and some other messages are sent to the terminal.

For program testing the following is implemented:

Some tables generated at initialization are available as files in the check_output directory. So the created full announce-list can be checked easily.
There is a set of command-files. These are used to check the routing kernel and should check all command-types with different parameters. Files to check the error behavior will follow.
For Windows10:
Install python (3.6, 32bit)
open powershell (AltX → i)
write something like:
cd <location of CR programm>
C:\Users\<your name>\AppData\Local\Programs\Python\Python35-32\python.exe commandrouter.py

These checks are initiated by a<n><SPACE> from the terminal. <n> is the number of the checkfile in the checkfiles directory.
r<SPACE> initiate a random input to the SK and device input. This will produce error messages but should not crash the program.
The actual throughput for random data is about 3000 loops/s (checking and analizing all inputs) with about 20kByte/s. Throughput will be higher with correct data, because random data produce many errors with writing the log and displaying on the terminal

Implementation of the Hardware

The CR should be implemented on a Raspberry PI. A Raspberry has ethernet access and a complete operating system. Power consumption an price is also acceptable.
May be that a web server for the SK can also run on the same machine..
not yet tested.

Hierarchical MYC System

In a hierarchical system the CR resolves the announcements of lower level CR but otherwise keep the sequence. So only the actual CR delivers the announce-list with the basic announcement at start and the reserved tokens and the I-line of the CR at the end (tabs for clarification only):

```
<0>;c;...                                   #CR
        <c><m>...                           #normal device
        ...
        I; ..                               # I-line
        ...
        <c>;c;...                           #lower level CR start
                <c><m>...                   #normal device
                ...
                I;...                       # I-line
                <c><m>...
                ...
        I;...                               # I-line, end of lower level CR
        …
        <c><m>..
        I;...                               # I-line
        ...
I;...   ...                                 # I-line of  CR
```

Identical device-types with identical individualization are not allowed within a CR but for CR in different levels.

Error handling

The routing kernel of the CR uses bytearray. Any character is allowed as input.
The CR do some checks on the inputs:

- for correct command-token
- for positional parameters and element number lower than maximum
- for correct length of string

<data> fields are not checked (with few exceptions), especially restriction of <ty> given in <des>.
Error messages are sent to logfile only.
If a error is detected, received characters are ignored and the next byte byte is seen as the start of the next command / answer.

Multiuser

Some interfaces will support multiuser access. This is not supported now.

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/commandrouter.zip