This is a description of the inputs and outputs of the parser, not specifically the functionality of it.

**Build**

Run the make.ps1 on PowerShell. This will output 4 files. 2 c files and their corresponding h files:  
lex.yy.c – contains the functionality to lex tokens.  
y.tab.c - contains the parser functionality

The entry function to the parser is in y.tab.c in the form of parseCommand(char \*)

**Functionality**

All commands should be entered in lower case.

Call parseCommand (written in parser.y and generated in y.tab.c) with the string that the user enters. This will return a struct. Its definition is in parse\_types.h. After parsing, the struct will contain a command code, and 1 or more arguments. All the command codes are defined in parser.h.

If the struct has the command code of error, the first byte in the arguments has the value of the reason. Each error code is defined in parser.h as well. Its definition name should suffice to say what happened.

If a command code has arguments, they will be in the args array. All number based argument definitions should be defined in parse\_types.h, like the different gpio ports and other hardware ports. As can be seen, each port is given a number.

If an argument is a string, it will be null terminated.

All the command codes and arguments have been thoroughly tested for validity and bounds, so no error checking needs to be done on the struct post parsing.

**Examples**

The struct:

{  
 command\_code: 5,  
 args\_len: 1,  
 args: [3]  
}

Means that the command is a **qsfp gpio read** command of the **lpmode** port.

The struct

{  
 command\_code: 22,  
 args\_len: 11,  
 args: [2, 5, ‘c’, ‘h’, ‘a’, ‘r’, ‘\0’, ‘h’, ‘i’, ‘!’, ‘\0’]  
}

Means to do a **PEK write over iic** on page **2** with starting address **5** with **char**acter data and write the message **hi!**

The struct

{  
 command\_code: 255,  
 args\_len: 1,  
 args: [9]  
}

Means that the command that got parsed was **invalid** and that it specified an **invalid port for the PEK**.

**Command Specific Limitations**

* When writing a hex value, prepend the value with “0x”
* Use single or double quotes when writing a data string (for example, the last argument in the pek iic write function)
* No command should be more than 50 characters