Context free grammar

Program

```
<program>       -> <stmts> $
<stmts>         -> <stmt> <stmts>
                |  λ
<stmt>          -> <if_stmt>
                |  <assign>
                |  <serial_stmt>
```

Serial

```
<serial_stmt>   -> SEND <string>
                |  BAUD <nr>
<string>        -> "[letter|nr]*"
                |  "[letter|nr]*" + <string>
                |  <expr>
<letter>        -> [a-z|A-Z]
```

IF

```
<if_stmt>       -> IF <logic_expr> <assign>
```

Assign

```
<assign>        -> SET <assignterm>
<assignterm>    -> <id> <expr>, <assignterm>
                |  <id> <expr>
                |  <id> [+=|-=] <value_expr>
```

Expression

```
<expr>          -> <value_expr>
                |  <logic_expr>
<value_expr>    -> <term>
                |  <value_expr> [+|-] <term>
<term>          -> <value> [*|/|%] <term>
                |  <value>
                |  READ
<logic_expr>    -> <logic_value>
                |  <logic_value> [OR|AND] <logic_expr>
                |  NOT <logic_value>
                |  <value_expr> [<|>|>=|<=] <value_expr>
```

Version 2

Identifier

```
<id>            -> FLAG#<set>
                |  PIN#<set>
                |  PINA#<set>
                |  COUNTER#<set>
                |  TIMER#<set>
```

Number

```
<nr>            -> [0-9]+
                |  [0-9]+ <suffix>
<suffix>        -> ms | s | m | t | d | u
```

Set

```
<set>           -> [<nr>|<range>]
                |  <set>, [<nr>|<range>]
<range>         -> <nr>..<nr>
```

Value

```
<value>         -> <nr>
                |  <id>

<logic_value>   -> HIGH
                |  LOW
                |  <id>
```