# Neo6502 API

The Neo6502 API is a messaging system. Messages are passed through a block of memory stored from $FF00 to $FF0F which is allocated as below.

There are *no* methods of directly accessing the hardware.

| Address | Contents | | |
|---|---|---|---|
| $FF00 | Group. API Commands are grouped by functionality ; so group 1 is the system function, and group 2 is the console I/O function for example.<br>Once a non zero value is written here the system will respond by setting values in the other registers appropriately, at the end they will clear this location. | | |
| $FF01 | Function. A command within the group, so for example Group 1 command 0 writes a value to the console, and Group 1 command 1 reads the keyboard | | |
| $FF02 | Return any error values, 0 = no error. | | |
| $FF03 | Information | 7 | Set to ‘1’ if the ESCape key has been pressed. This is not automatically reset. |
| | | 6 | Unused |
| | | 5 | Unused |
| | | 4 | Unused |
| | | 3 | Unused |
| | | 2 | Unused |
| | | 1 | Unused |
| | | 0 | Unused |
| $FF04-B | Parameters, known as Params 0 through 7. These can be combined to form 16 or 32 bit integers. | | |
| $FF0C-F | Reserved | | |

In the include file neo6502.inc the value of the first is the identifier ControlPort. This also has addresses of WaitMessage, SendMessage and some helper functions.

Messages are sent as follows.

| | |
|---|---|
| 1 | Wait for any pending command to complete. There is a subroutine WaitMessage which does this for the developer |
| 2 | Set up the parameters if any. For example printing a character to the console is done by putting its ASCII value into $FF04. |
| 3 | Setting the function code at $FF01 |
| 4 | Writing the command to $FF00. This h*as* to be done last as setting it to non zero before the parameters are set up may cause the message to be processed. On a technical point, both implementations process the message immediately on write. |
| 5 | Optionally, wait for completion. Some commands, e.g. 2,2 which reads from the keyboard queue return a value in a parameter. Things like writing to the console do not need to wait for completion, as any subsequent command will wait for the command to complete as per 1. |

There is a support function SendMessage which inlines the command and function e.g. this code from the Kernel.

```
    jsr  KSendMessage        ; send command 2,1 read keyboard
    .byte 2,1
    jsr  KWaitMessage        ; waiting for message to be sent back
    lda  DParameters         ; read result
```

You could write this as the following – it's just more longwinded.

```
    lda  #1                  ; do command 2,1
    sta  DFunction
    lda  #2
    sta       DCommand
Loop:
    lda  DCommand            ; signal done by this being zero
    bne  Loop
    lda  DParameters         ; get result
```

| Group | Func | Description |
|---|---|---|
| 1 | 0 | Resets the messaging system and its components. Should not normally be used. |
| | 1 | Return the value of the 100Hz system timer in Params 0-3 |
| | 2 | Return the state of keyboard key Param0 in Param0 |
| | 3 | Execute BASIC *(Loading currently does not work)* |
| 2 | 0 | Write character Param0 to the console. 32-127 are standard ASCII, 8 is Backspace, 13 Return. There are other console codes documented later. |
| | 1 | Read and remove a key press from the keyboard queue into Param0 , this is the ASCII value of the keystroke. If there are no key presses in the queue, Param0 is zero.<br>Note that this method is *not* for games, where key presses and releases replace a joystick. The system maintains a bit array to check if keys are pressed. |
| | 2 | Check to see if the keyboard queue is empty. If it is Param0 is $FF, otherwise it is $00 |
| | 3 | Input the line the screen is currently on to YX as a length prefixed string, put the cursor on the line below the line input, handles multiple line input. |
| 3 | 1 | Display the directory |
| | 2 | Load a file from name Param0/1 (Length prefixed) to address Param2/3, error code in Param0 |
| | 3 | Save a file to name Param0/1 (Length prefixed) from address Param2/3 length Param4/5 bytes, error code in Param0 |
| 4 | 0-15 | Binary mathematics operations |
| | 16-31 | Unary mathematics operations |
| | 32-47 | Miscellaneous operations. |

Console Codes (tbc)