Orcus

Mit „Orcus“ wird sowohl ein transneptunischer Asteroid im Kuipergürtel bezeichnet als auch der Gott der Unterwelt in der römischen Mythologie. Hier ist mit dem Begriff „Orcus“ ein Mikrocontrollerboard gemeint, das dem Anwender die Steuerung von drei unabhängigen Modulen über TCP/IP oder einer Webpage ermöglicht. Diese Module sind eine mit Hilfe zweier Servos in X- und Y-Richtung frei schwenkbare Webcam die auch beliebige Scanpfaden ablaufen kann, ein einzeiliges LED-Punktmatrix-Display mit ca. 5 cm Zeichenhöhe, sowie acht LED-Spots, die über logische und zeitliche Verknüpfungen ein- und ausgeschaltet werden können.

Das Orcus-Board lauscht für jedes Modul separat jeweils an einem TCP/IP-Port auf Verbindungsanfragen und kann über einfache Text-Befehle gesteuert werden.

IP-Adresse:	`10.27.210.80`
Domain-Name:	`http://orcus.da.private.hm.edu/`
Display-Port:	`666`
LED-Port:	`667`
Servo-Port:	`668`
Username:	`user`
Passwort:	`1234`

Kamera-Live-Bild :	`http://user:1234@10.27.210.203/mjpg/video.mjpg`
Kamera-Screenshot:	`http://user:1234@10.27.210.203/snapshot.jpg`

Da die Übertragung via TCP/IP unverschlüsselt in Klartext erfolgt, kann entweder Telnet verwendet werden (beispielsweise `'telnet 10.27.210.80 666'`), um eine Verbindung zu einem der Module aufzunehmen, oder das Programm Putty (IP-Adresse und Port angeben; Connection Type: RAW). Es wird aber empfohlen, den Client `'Orcus.exe'` für Windows und `'orcus'` für Linux zu verwenden. Dieser vereinfacht die Steuerung und erlaubt die Abarbeitung von Skripten.
Alternativ bietet das Orcus-Board auch eine eigene passwortgesicherte Administrator-Webpage, über die alle drei Module einfach zu bedienen sind. Um diese aufzurufen, reicht es auf, die IP-Adresse der Orcus-Board in die Adresszeile eines Browsers einzugeben.
Besteht eine Verbindung zu einem der Module, dann leuchtet an der Vorderseite des Orcus-Board die entsprechende blaue LED auf, um zu signalisieren, dass dieses Modul in Verwendung ist und somit nicht mehr von einem weiteren Computer erreicht werden kann. Eine Ausnahme bildet hier die Webpage des Orcus-Board, die jederzeit zusätzlich zu den Modul-Verbindungen beliebig oft aktiv sein darf. Wenn die Webpage mindestens einmal aufgerufen ist, so leuchtet die gelbe LED an der Vorderseite des Orcus-Board auf.
Eine langsam blinkende grüne LED zeigt den normalen Betrieb des Boards an. Wird der Reset-Knopf betätigt, wird das Board in seiner Grundkonfiguration neu gestartet.

Orcus-Client

Um die Kommunikation mit dem Orcus-Board zu erleichtern, steht ein Client zur Verfügung, der jeweils in einer Version für Windows (`Orcus.exe`) und Linux (`orcus`) von der **internen Webpage** heruntergeladen werden kann. Die Windows-Version des Clients ist sofort einsatzbereit, wobei der Linux-Version erst mit dem Befehl `'chmod u+x ./orcus'` die Berechtigung zum Ausführen erteilt werden muss. Der Client bietet eine Kommandozeileneditierfunktion und eine Befehls-History, die über die Pfeiltasten aufgerufen werden können. Außerdem können mit dem Client sehr einfach Skripte ausgeführt werden.

Der Client kann optional mit Parameter von der Kommandozeile aus gestartet werden. Ohne Parameter wird vorläufig noch keine Verbindung zum Orcus-Board aufgebaut, jedoch kann mit dem Befehl `'-help'` eine Liste an Client-Befehlen angezeigt werden. Beendet wird das Programm durch den Befehl `'exit'` oder durch die Tastenkombination [Strg]+[C]. Um direkt eine Verbindung zu einem der drei Module des Orcus-Boards aufzubauen, können folgende Komandozeilenparameter verwendet werden: `'display'`, `'led'` oder `'servos'`. Beispielsweise würde `'./orcus display'` unter Linux sofort eine Verbindung zum Display-Modul herstellen. Alternativ kann auch eine gewünschte Portnummer angegeben werden. Dabei wird immer die Standard-IP-Adresse `10.27.210.80` zugrunde gelegt. Es ist aber auch möglich, eine Kombination aus IP-Adresse und Port anzugeben. Dies ist allerdings nur dann sinnvoll, wenn sich die IP-Adresse des Orcus-Boards ändern sollte.

Zwischen den drei Modulen kann jederzeit mit den Befehlen `'-Display'` sowie `'-Led'` oder `'-Servos'` gewechselt werden. Dies kann man auch durch `'-d'`, `'-l'` oder `'-s'` abkürzen. Dabei wird eine möglicherweise bestehende Verbindung zu einem der Module geschlossen und zum Ziel-Modul neu aufgebaut. Um die aktuelle Verbindung nur zu schließen, nicht aber den Client zu beenden, dient der Befehl `'logout'`. Alle Befehle, die mit einem Minuszeichen beginnen, sind reine Client-Befehle. Informationen in eckigen Klammern werden vom Client generiert; alle anderen Informationen stammen vom Orcus-Board selber.

Wird eine Verbindung zu einem Modul erfolgreich aufgebaut, so meldet sich die jeweilige Passwortabfrage des Moduls. Hat man sich erfolgreich eingeloggt, so kann man durch den Befehl `'help'` eine Liste der gültige Kommandos für das Modul einsehen. Beachten Sie das `'-help'` die Befehle des Clients anzeigt.

Mit dem Befehl `'-keys'` oder `'-k'` gelangt man in einen speziellen Keys-Modus, der nur über den Client verfügbar ist. Dabei werden abhängig vom aktuellen Modul einzelne Tasten der Tastatur in Befehle umgesetzt und direkt abgeschickt. Der Keys-Modus listet beim Aufruf eine Übersicht über die gültigen Tasten auf und kann jederzeit mit der Escape-Taste beendet werden.

- Im SERVO-Modus lässt sich die Kameraposition mit den Tasten [W], [S], [A], [D] direkt steuern und die Servos über [.] ein- und über [L] wieder ausschalten. Die Kamera wird mit [,] ein- und mit [K] wieder ausgeschaltet.
- Im LED-Modus lassen sich die acht LED-Strahler mit den Tasten [1] bis [8] toggeln, also setzen wenn diese gerade zurückgesetzt sind und umgekehrt. Die Tastenreihe von [Q] bis [I] unterhalb der Zahlentasten dient zum Setzen (Einschalten) des jeweiligen LED-Strahlers. Die Tasten der Reihe darunter von [a] bis [k] setzen den jeweiligen LED-Strahler zurück (Ausschalten). Mit [,] können alle LED-Strahler gleichzeitig gesetzt werden und mit [.] werden alle LED-Strahler zurückgesetzt. Die Taste [-] dient dem toggeln aller LEDs.

- Im DISPLAY-Modus kann man direkt auf das LED-Punktraster-Display schreiben und mit der Backspace-Taste das letzte Zeichen zurücknehmen. Es sind alle Zeichen des Displays sowie die entsprechenden Symbol und Sonderzeichen zulässig. Beachten Sie, dass das Display die Zeichen verhältnismäßig langsam aneinanderreiht, wodurch kurze Verzögerungen entstehen können.

Der Client eignet sich auch, um Skripte auszuführen. Dazu werden in eine einfache Textdatei die auszuführenden Befehle gespeichert. Wird diese Textdatei als Standardeingabe in den Client geleitet, so werden die einzelnen Befehle nacheinander ausgeführt. Leerzeilen und Kommentare, also Zeilen die entweder mit `'//'` oder `'#'` anfangen, werden übersprungen. Um eine reine Textausgabe auf dem Terminal zu erzeugen dient der Client Befehl `'-draw (Nachricht)'`.

Beispiel:

```
// Scan.txt

-draw ######################## SCAN DEMO ########################

// Verbindungsaufbau zu Servos und Eingabe des Passworts
-servos
1234

// Kamera und Servos aktivieren
enable camera
enable servos

// Scanpunkte X Y und Geschwindigkeit festlegen
new scan 950 400 3 3
new scan 100 440 2 2
new scan 100 560 3 3
new scan 950 640 3 3
enable loop

// Scannen starten
start

// Client beenden
-exit
```

Mit dem Aufruf `'Orcus < Scan.txt'` unter Windows oder `'./orcus < Scan.txt'` wird das Skript gestartet wenn es im selben Verzeichnis liegt. Wie bei jedem Kommandozeilenprogramm lässt sich mit dem `'>'`-Zeichen die Ausgabe auch in eine Datei umleiten.

Display-Modul

Das Display-Modul bietet ein einfaches Protokoll zur Übertragung von Nachrichten, das an POP3 angelehnt ist. Wie bei POP3 werden alle korrekten Nachrichten vom Board mit `'+OK'` begonnen und alle Fehlermeldungen mit `'-ERR'`.Es lassen sich mit dem Befehl `'SEND (Nachricht)'` bis zu vier kurze Nachrichten mit jeweils einer maximalen Länge von 62 Zeichen speichern. Eine dieser Nachrichten kann mit `'DISPLAY (n)'` auf einem Display angezeigt werden. Alternativ kann man auch mit `'WRITE (Nachricht)'` eine Nachricht direkt auf das Display schreiben. Bei der Anzeige auf dem Display wird nicht zwischen Groß- und Kleinschreibung unterschieden. Gültig sind alle Buchstaben ohne Umlaute und alle Zahlen, sowie Space und die Zeichen `','`, `'?'`, `'!'` und `'$'`. Folgende Sonderzeichen werden ebenfalls unterstützt:

- `'#'` = Animiertes Herz
- `'>'` = Animierter Fisch
- `'*'` = Animierter Stern
- `'@'` = Animierter Smiley
- `'+'` = Scroll
- `'{'` = Recall
- `'('` = Links
- `')'` = Rechts
- `'&'` = Clear
- `'"'` = Umschalten in Uhr-Modus
- `'%'` = Plus eine Stunde
- `'{'` = Plus eine Minute
- `'#'` = Plus eine Sekunde
- `'='` = Reset

Alle Zeichen außer „Reset“ symbolisieren eine interne Bedientaste des Displays und stellen deren Funktion dar. Wird das Display in den Uhr-Modus versetzt, so beginnt die Zeit zu laufen und kann über die entsprechende Anzahl von Zeichen gestellt werden. Der Modus wird mit Reset wieder verlassen. Es wird empfohlen, vor dem Wechsel in den Uhr-Modus ein Reset Zeichen zu senden, um die Uhr auf 12:00 Uhr zu stellen. Das Scroll-Zeichen toggelt den Zustand, ob die Nachricht über das Display läuft und sollte am Ende jeder Nachricht angehängt werden.

Beispiele:
// Eine Nachricht, angeführt von einem Smiley und einem Herz scrollt über das Display
```
'WRITE @# HALLO WELT!+'
```

// Es wird die Uhr auf 20:15 gestellt
```
'WRITE ="%%%%%%%{{{{{{{{{{{{{{'
```

Befehle

STATUS

Zeigt die Anzahl der gespeicherten Nachrichten an sowie die Gesamtzahl der davon belegten Bytes. Außerdem wird angegeben ob das Display eingeschaltet ist.

```
STATUS
+OK Status: 3 Messages  26 bytes
Display enabled
```

WRITE (message)

Auf dem Display wird die übergebene Nachricht angezeigt. Das Display wird gegebenenfalls eingeschaltet. Ein '=' –Zeichen am Anfang der Nachricht sorgt dafür, dass das Display zuerst gelöscht wird, ansonsten wird die Nachricht an die aktuelle angehängt. Wenn auf dem Display noch eine Nachricht scrollt oder nach dem Scrollen gestoppt wurde, wird die neue Nachricht verworfen.

```
WRITE =Hallo Welt # > * @+
+OK Written on display:
=Hallo Welt # > * @+
```

LIST

Alle vier Nachrichten sowie deren Anzahl an Bytes werden aufgelistet. Längere Nachrichten werden abgekürzt.

```
LIST
+OK List:
 1. 27 byte  Hallo Welt wie geht  ...
 2. 23 byte  Bender ist der Groes ...
 3.  0 byte
 4.  0 byte
```

SEND (m)

Eine neue Nachricht wird abgespeichert.

```
SEND Hallo Welt!!
+OK Stored message 3  12 bytes:
Hallo Welt!!
```

SHOW (n)

Die ganze Nachricht mit der Nummer (n) wird angezeigt. Die Identifikationsnummer richtet sich nach der Nummer unter der die Nachricht unter dem Befehl „LIST“ geführt wird.

```
SHOW 2
+OK Show message 2  23 bytes
Bender ist der Groesste
```

DELETE (n)

Die Nachricht mit der Nummer (n) wird gelöscht.

```
DELETE 1
+OK Deleted message 2
```

DISPLAY (n)

Die Nachricht mit der Nummer (n) wird an das Display geschickt und dort angezeigt.

```
DISPLAY 1
+OK Message 1 send to display
```

CLEAR

Die Ausgabe auf dem Display wird gelöscht.

```
CLEAR
+OK Display cleared
```

SCROLL

Die Ausgabe auf dem Display fängt an von rechts nach links zu laufen oder wird gestoppt, wenn diese bereits scrollt.

```
SCROLL
+OK Toggled display scroll
```

ENABLE

Das Display wird aktiviert.

```
ENABLE
+OK Display enabled
```

DISABLE

Das Display wird ausgeschaltet.

```
DISABLE
+OK Display disabled
```

NOOP

Es wird „+OK“ zurückgegeben.

```
NOOP
+OK
```

RESET

Alle Nachrichten werden gelöscht und das Display wird zurückgesetzt.

```
RESET
+OK Reset
```

HELP

Eine Liste der verfügbaren Befehle wird angezeigt.

LOGOUT

Die Verbindung wird beendet.

```
LOGOUT
+OK Logging out
```

Servos-Modul

Das Modul SERVOS dient zur Steuerung zweier Servo-Motoren auf denen eine Webcam angebracht ist, die sich in X- und Y-Richtung schwenken lässt. Mit dem Befehl `'enable servos'` werden die Servos aktiviert. Mit Richtungsanweisungen wie `'left'`, `'right'`, `'up'` oder `'down'` lässt sich die Kamera schwenken oder mit `'set servos (x) (y)'` auf eine bestimmte Position setzen.
Durch den Befehl `'new scan (x) (y) (sx) (sy)'` kann ein neuer Zielpunkt an eine Liste von maximal 15 Punkten angehängt werden. Dabei kann für beide Servos jeweils eine Geschwindigkeit in den Stufen 1 (langsam) bis 5 (schnell) angegeben werden. Diese Punkte in der X-Y Ebene werden nach dem Befehl `'start'` nacheinander angefahren. Erst wenn ein Punkt sowohl für X als auch für Y erreicht ist, wird der nächste Punkt angesteuert. Ist die Liste am Ende angelangt, so stoppt die Kamera an diesem Punkt. Mit `'enable loop'` kann man erreichen, dass die Liste permanent durchlaufen wird. Auf diese Weise lässt sich sehr einfach ein Raumüberwachung realisieren. Der Schwenkbereich des Servos in X-Richtung geht von 0 bis 1800 und in Y-Richtung von 0 bis 1750. Die Grundposition ist X: 1750 und Y: 350.
Das Lifebild der Webcam ist mit einen Webbrowser entweder über die Orcus-Webpage oder unter der auf der ersten Seite angegebenen Adresse zu erreichen. Dabei ist zu beachten, dass die Kamera erst mit dem Befehl `'enable camera'` eingeschaltet werden muss. Diese benötigt etwa 25 Sekunden zum Booten und ist erst nach dieser Zeit unter der IP-Adresse zu erreichen. Loginname und Passwort finden Sie ebenfalls auf der ersten Seite dieses Handbuchs.
Ein Standbild („Schnappschuss“) kann ebenfalls erzeugt werden (siehe erste Seite des Handbuchs). Wenn alle Verbindungen zum Servos-Modul, ob über TCP/IP oder Webpage, geschlossen werden, dann fahren die Servos in die Grundposition und die Kamera und die Servos schalten sich ab. Dies geschieht nicht, wenn sich die Servos in einem aktiven Scanmodus befinden.

Befehle

`enable servos`

Die Servos werden aktiviert und in die Grundposition gebracht.

```
enable servos
Servos enabled
```

`disable servos`

Die Servos werden in die Grundposition gefahren und dort abgeschaltet.

```
disable servos
Servos disabled
```

status

Die aktuelle Position der Servos wird angegeben sowie der Status der Kamera und ob im Moment ein Scanvorgang läuft. Wenn die Servos deaktiviert sind, wird eine entsprechende Nachricht angezeigt.

```
status
Servo X position:  900
Servo Y position:  550
No scanning
Camera disabled
```

set servos (x) (y)

Die beiden Servos werden in die angegebenen Position gefahren. Dabei müssen die Werte für Servo X im Bereich 0 bis 1800 liegen und für Servo Y im Bereich von 0 bis 1750.

```
set servos 200 500
New servo positions: X:  200  Y:  500
```

set servo x (x)

Der Servo in X-Richtung wird auf die Position (x) gefahren. Diese muss im Bereich von 0 bis 1800 liegen.

```
set servo x 100
New servo X position:  100
```

set servo y (y)

Der Servo in Y-Richtung wird auf die Position (y) gefahren. Diese muss im Bereich von 0 bis 1800 liegen.

```
set servo y 100
New servo Y position:  100
```

left [n]

Der Servo in X-Richtung wird um [n] Schritte nach links gedreht.
Wird keine Zahl angegeben, so wird der Servo um 50 Schritte gedreht.

```
left 300
New servo X position:  400
```

right [n]

Der Servo in X-Richtung wird um [n] Schritte nach rechts gedreht.

Wird keine Zahl angegeben, so wird der Servo um 50 Schritte gedreht.

```
right
New servo X position:  350
```

up [n]

Der Servo in Y-Richtung wird um [n] Schritte nach oben gedreht.

Wird keine Zahl angegeben, so wird der Servo um 50 Schritte gedreht.

```
up
New servo Y position:  550
```

down [n]

Der Servo in Y-Richtung wird um [n] Schritte nach unten gedreht.

Wird keine Zahl angegeben, so wird der Servo um 50 Schritte gedreht.

```
down 100
New servo Y position:  450
```

enable camera

Die Kamera wird eingeschaltet.

```
enable camera
Camera enabled
```

disable camera

Die Kamera wird abgeschaltet.

```
disable camera
Camera disabled
```

new scan (x) (y) (sx) (sy)

Mit diesem Befehl wird ein neuer Zielpunkt an die Liste der Scanpunkte angehängt. Zusätzlich werden die Geschwindigkeiten für die Servos in X und Y-Richtung angegeben.

```
new scan 0 550 3 2
New scanning position
 1 X:    0 Y:  550 Speed X: 3 Speed Y: 2
```

list

Die Scanliste wird angezeigt. Im Scanmodus werden die Einträge von oben nach unten abgearbeitet und die jeweilige Position wird mit der jeweils gewählten Geschwindigkeit angefahren. Erst wenn beide Servos die Zielposition erreicht haben, wird die nächste Position angefahren. Wenn der Scan Loop aktiviert ist, dann wird die Liste wieder von der Position 1 aus durchlaufen. Sonst verweilen die Servos in ihrer letzten Position.

```
list
Scan list entries:  3 Loop OFF
 1 X:    0 Y:  550 SpX: 3 SpY: 2
 2 X: 1800 Y:  400 SpX: 3 SpY: 1
 3 X:  900 Y:  550 SpX: 5 SpY: 5
```

delete list

Alle Einträge der Scanliste werden gelöscht, der Scan Loop deaktiviert und der Scanmodus wird gestoppt.

```
delete list
Scan list deleted
```

start

Der Scanmodus wird aktiviert und die Scanliste wird abgearbeitet.

```
Start
Started scanning
```

pause

Das Scannen wird angehalten und es kann ab der aktuellen Position fortgefahren werden.

```
pause
Paused scanning
```

stop

Das Scannen wird angehalten und wird bei Wiederaufnahme mit dem ersten Eintrag der Scanliste fortgesetzt.

```
stop
Stopped scanning
```

enable loop

Wenn die Scanliste durchlaufen wurde, dann wird diese automatisch wieder von vorne begonnen.

```
enable loop
Enabled scan loop
```

disable loop

Wenn die Scanliste durchlaufen wurde, dann bleiben die Servos auf der letzten Position.

```
disable loop
Disabled scan loop
```

help

Eine Liste der verfügbaren Befehle wird angezeigt.

reset

Die Servos werden in die Grundposition gefahren, die Scan Liste wird gelöscht, das Scannen wird beendet und der Scan Loop wird deaktiviert.

```
reset
Servo reset
```

logout

Die Verbindung wird beendet.

```
logout
Logging out
```

LED-Modul

Mit dem LED-Modul ist es möglich, acht LED-Spots programmgesteuert ein- und auszuschalten. Die LEDs werden im Folgenden als „Pins“ bezeichnet und werden durch DO1 bis DO8 angesprochen. DO1 bezieht sich auf den linken oberen Spot. Das Setzen eines Pins erreicht man durch den Befehl `'set pin [DOX]'` und das Rücksetzen mit `'reset pin [DOX]'`. Den Zustand wechselt ein Pin mit `'toggle pin [DOX]'`. Nach einem Befehl können mehrere Pins durch Kommas getrennt angegeben werden. Die Bezeichnung `'ALL'` wählt alle Pins aus. Außerdem kann jedem Pin jeweils eine eigene logische Verknüpfung und eine einfache Signalerzeugung zugewiesen werden. Damit sind ausgefallene Blinkmuster möglich.

Logic

Das logische Verhalten der einzelnen Pins wird durch eine Zeichenfolge beschrieben, die an die Boolesche Algebra angelehnt ist. Durch Klammerung kann man die Prioritäten und Reihenfolge der Auswertung bestimmen und somit komplexe logische Verschaltungen realisieren.

In der folgenden Liste sind die unterstützten Operatoren hierarchisch angeordnet.

- `(...)` Klammerung
- `NAND`
- `NOR`
- `NOT`
- `AND`
- `XOR`
- `OR`

Es können alle Pins in die Verknüpfung mit einbezogen werden, die bei der Auswertung durch den jeweiligen momentanen Booleschen Wert '1' oder '0' ersetzt werden. Außerdem können auch die festen Werte '0' und '1' im logischen Term als feste Konstanten auftauchen.
Jeder gültige logische Term ist syntaktisch so aufgebaut, dass zuerst der Zielpin angegeben wird, dem einer der Zuweisungsoperatoren `'='`, `'<-'` oder `'<='` folgen muss. Diesem Pin wird der gesamte folgende Ausdruck zugeordnet.

Danach muss immer ein Boolescher Wert angegeben werden. Folgt diesem ein Operator, so muss ein weiterer Boolescher Wert angeschlossen werden, so dass jeder Operator immer von zwei Booleschen Werten umgeben ist. Ein NOT kann immer nur vor einem Booleschen Wert stehen.
Der logische Ausdruck kann beliebig durch Klammern verschachtelt werden. Es muss jedoch darauf geachtet werden, dass jede geöffnete Klammerung auch wieder geschlossen wird.
Ein `'('`-Zeichen kann nur vor einem Booleschen Wert stehen, wohingegen ein `')'`-Zeichen immer nur nach einem Booleschen Wert angegeben werden kann. Es können beliebig viele Klammern vor einem Booleschen Wert geöffnet beziehungsweise nach einem Booleschen Wert geschlossen werden.

Beispiel für den Aufbau einer logischen Verknüpfung:

```
Zielpin = <BW> <OP> ( ( <BW> <OP> <BW> ) <OP>
( <BW> <OP> ( <BW> <OP> ( <BW> <OP> <BW> ) ) ) <OP> <BW> )
```

```
DO1 = DO1 and ( ( not DO2 or DO3 ) xor
( DO2 nand ( 1 nor ( DO1 and DO4 ) ) ) nor 0 )
```

Der gesamte logische Term liefert nach einer Auswertung immer entweder '1' oder '0' zurück, das aus den Teilergebnissen der Auswertung berechnet wird.
Bei einer positiven Auswertung mit '1'-Pegel, können abhängig vom Pin-Mode folgende Ereignisse ausgelöst werden:

- Der Zielpin folgt dem booleschen Wert (Logic output)
- Der Zielpin wechselt seinen booleschen Wert (Logic toggle)
- Das Pattern wird gestartet (Pattern start)
- Das Pattern wird an- oder ausgeschaltet (Pattern toggle)
- Das Pattern wird pausiert (Pattern pause)

Um einem Zielpin eine neue logische Verknüpfung zuzuweisen, verwendet man den Befehl `'set logic [DOX = Logic]'`.

Pattern

Ebenso wie das logische Verhalten kann jedem Pin auch ein individuelles zeitliches Verhalten zugewiesen werden. Dabei werden zwischen den booleschen Zuständen 'High' und 'Low' verschiedene einstellbare Wartezeiten festgelegt. Dadurch wird eine einfache Signalgenerierung realisiert. Außerdem ist es möglich, beliebig verschachtelte Schleifen einzubauen, die entweder unendlich oft durchlaufen werden oder in einer auswählbaren Anzahl an Wiederholungen. Das Pattern wird durch `'set pattern [DOX = Pattern]'` festgelegt. Der Zuweisungsoperator `'='` kann auch durch ein `':'` ersetzt werden und ordnet das Pattern einem bestimmten Zielpin zu. Danach muss eine Zeichenkette folgen, die das zeitliche Verhalten des Pegels bestimmt.

Folgende Operatoren können im Pattern eingesetzt werden:

- `High` Der Pin wird auf '1' gesetzt
- `Low` Der Pin wird auf '0' gesetzt
- `Toggle` Der Zustand des Pins wird gewechselt
- `zulässig` Es werden XX Millisekunden gewartet
- `Wait XX s` Es werden XX Sekunden gewartet
- `(...)loop` Endlosschleife
- `(...)repeat[XX]` Endliche Schleife die XX mal durchlaufen wird
- `{XXms|101101101}` Bitfeld mit XX Sekunden oder Millisekunden Wartezeit

Beispiel:

```
DO1: high wait 3 s low wait 2 s ( toggle wait 500 ms )repeat[10] low
```

Nach Start des Patterns wird der Pin DO1 für 3 Sekunden auf '1' gesetzt und danach für 2 Sekunden auf '0'. Die Schleife wird zehnmal durchlaufen, wobei der Pin jedes Mal invertiert wird und 500 Millisekunden seinen Zustand beibehält. Dadurch blinkt der Pin fünfmal auf. Danach wird der Pin dauerhaft auf '0' gesetzt.

Mit `'High'`, `'Low'` und `'Toggle'` wird der Pegel eines Pins bestimmt. Dieser wird solange beibehalten, bis er explizit geändert wird.
Mit `'Wait XX ms'` oder `'Wait XX s'` wird die Wartezeit festgelegt. Diese kann sowohl in Millisekunden als auch in Sekunden angegeben werden und muss im Bereich von 0 bis 65535 liegen. Stehen mehrere Wait-Anweisungen hintereinander, so addieren sich deren Wartezeiten. Bei der Kombination der Operatoren ist lediglich darauf zu achten, dass innerhalb einer Schleife mindestens eine Wait-Anweisung enthalten sein muss.
Stehen mehrere Anweisungen hintereinander, die einen Signalwechsel hervorrufen, so wird immer nur der letzte Zustand am Pin ausgegeben, den dieser vor einer Wait-Anweisung oder dem Ende des Patterns hätte.

Beispiel:

```
DO2: high low toggle high low high
DO3: high toggle toggle
```

Der syntaktisch richtige erste Ausdruck bewirkt lediglich, dass der Pin DO2 auf '1' geht. Die vorherigen Signalzuweisungen haben keinerlei Auswirkungen. Das zweite Pattern bewirkt, dass der Pin DO3 ebenfalls auf '1' gesetzt wird, da der Pin auf 'High' Pegel geht und danach zweimal invertiert wird. Da die Signalzuweisungen idealisiert keinerlei Zeit benötigen, gehen die Pins DO2 und DO3 in diesen Beispielen niemals auf '0' Pegel.

Bestimmte Teile des Patterns können durch Schleifen wiederholt werden. Um den Anfangspunkt einer Schleife festzulegen, wird das `'('`-Zeichen verwendet. Es ist darauf zu achten, dass jede geöffnete Klammer auch wieder geschlossen wird. Der gesamte Inhalt zwischen den Klammern wird bei jedem Schleifendurchgang von vorne wiederholt.
Wird eine Klammer mit dem Ausdruck `')loop'` geschlossen, so bewirkt dies eine endlose Wiederholung der Schleife. Sämtliche Anweisungen nach einer Endlosschleife können niemals erreicht werden.
Um ein definierte Anzahl von Schleifendurchgängen zu realisieren, kann der Befehl `'repeat[XX]'` verwendet werden. Der Parameter in den eckigen Klammern gibt die Anzahl der Durchläufe an und muss eine Zahl im Bereich von 0 bis 65535 sein. Der Ausdruck `')repeat[0]'` entspricht der Endlosschleife. Jede Schleife wird mindestens einmal komplett durchlaufen. Wurde die endliche Schleife gemäß der vorgegebenen Wiederholungen abgearbeitet, so wird mit der Rest des Patterns weitergemacht. Sollte eine endliche Schleife aufgrund einer anderen Schleife erneut erreicht werden, so werden die Wiederholungen erneut in vollem Umfang ausgeführt.

Beispiel:

```
DO4: ( high wait 2 s ( toggle wait 100 ms )repeat[40] low wait 3 s )loop
```

Der Pin DO4 bleibt für 2 Sekunden auf '1' und blinkt danach 20 mal auf. Nach 3 Sekunden '0'-Pegel beginnt die Signalerzeugung wieder von vorne.

Da viele Anwendungen ein Pattern mit gleichem Takt verwenden, gibt es zusätzlich die Möglichkeit, das Pattern durch Bitfelder zu erzeugen.

Beispiel:

```
DO5: { 100 ms | 1010101011100100001 }
```

Hierbei wird eine gemeinsame Wartezeit festgelegt, die für alle Signalzuweisungen des Bitfeldes gilt. Die Wartezeit kann ebenfalls in Sekunden oder Millisekunden angegeben werden. Da Bitfelder mit den übrigen Operatoren beliebig kombinierbar sind, werden sie durch geschweifte Klammern eingefasst.
Zuerst muss die Wartezeit angegeben werden, die durch ein `'|'`-Zeichen von den beiden Operatoren `'1'` und `'0'` getrennt wird. Leerzeichen innerhalb des Bitfeldes werden übersprungen. Es wird bei Erreichen eines Bitfeldes immer sofort mit der ersten Wertezuweisung angefangen, der erst dann die gemeinsame Wartezeit folgt. Es ist zu beachten, dass nach der letzten Wertezuweisung im Bitfeld ebenfalls noch die angegebene Zeit gewartet wird, bevor das Bitfeld verlassen wird. Bitfelder sind übersichtlich zu lesen und sparen Programmspeicher ein.

Wurde einem Pin ein Pattern zugewiesen, so läuft dieses nicht automatisch ab, sondern muss erst angestoßen werden. Wird ein Pattern gestoppt und wieder gestartet, setzt es nicht an der gestoppten Stelle ein, sondern wird von Anfang an abgespult. Das Pattern kann durch den Befehl `'start pattern [DOX|ALL]'` in den Modi 'Pattern user', 'Pattern start' 'Pattern toggle' und 'Pattern pause' manuell ausgelöst werden.

Pin-Mode

Den Pins kann jeweils ein eigener Pin-Mode zugewiesen werden, der sein Verhalten abhängig von den logischen und zeitlichen Verknüpfungen beeinflusst. Wenn diese im auswählten Pin-Mode nicht verwendet werden, so wird die Verknüpfung in eckigen Klammern angezeigt. Der Zuweisungsoperator im logischen Term ändert sich ebenfalls entsprechend.

User

Der Pin wechselt nur bei direkten Befehlen seinen Zustand. Sind logische oder zeitliche Verknüpfungen festgelegt, so werden diese ignoriert und in eckigen Klammern angezeigt. Jeder Pin, der nicht benötigt wird, sollte in den Modus 'User' gebracht werden, um den Prozessor nicht unnötig zu belasten.

Logic output

Die logische Verknüpfung legt den Pegel des Pins in Echtzeit fest. Der Pin ist genau solange auf '1' wie der Boolesche Ausdruck erfüllt ist, danach wechselt er auf '0'. Das Pattern wird in diesem Modus ignoriert. Das Setzen oder Rücksetzen des Pins per Befehl ist nur dann möglich, wenn kein logischer Term für diesen Pin definiert ist.

Logic toggle

Jedes Mal wenn die logische Verknüpfung erfüllt ist, wechselt der Pin seinen Pegel, der auch beibehalten wird, wenn der logische Term eine '0' zurückgibt. Auf diese Weise ist es möglich, einen Pin zum Beispiel durch einen Tastendruck in einen gesetzten Zustand zu bringen und durch einen weiteren Tastendruck wieder zurückzusetzen. Das `'<='` im logischen Term zeigt an, dass der Zielpin nicht direkt den Rückgabewert der logischen Verknüpfung annimmt, sondern durch dieses Triggerevent den Zustand wechselt. Ein direktes Setzen oder Rücksetzen des Pins über einen Befehl ist zulässig. Das Pattern wird für diesen Modus nicht verwendet.

Pattern user

Das Pattern kann in diesem Modus nur durch den Befehl `'start pattern [DOX|ALL]'` ausgelöst werden. In der Zeit in der das Pattern aktiv ist, ist der Pattern-Modus 'On' und kann durch den Befehl `'stop pattern [DOX|ALL]'` angehalten werden. Die logische Verknüpfung wird nicht verwendet.

Pattern start

In diesem Modus stößt die logische Verknüpfung das Pattern an, wenn der logische Term eine '1' zurückgibt. Ebenso kann das Pattern auch durch den Befehl `'start pattern [DOX|ALL]'` ausgelöst werden. Das Pattern läuft solange, bis es entweder sein Ende erreicht oder per Befehl gestoppt wird. Danach kann es durch die logische Verknüpfung erneut gestartet werden. Während der Pattern-Modus 'On' ist, hat eine erneute positive Auswertung des logischen Terms keine Auswirkungen.

Pattern toggle

Das Pattern wird in diesem Modus durch die logische Verknüpfung gestartet und gestoppt. Wenn die logische Verknüpfung positiv ausgewertet wird, während der Pattern-Modus 'Off' ist, dann wird das Pattern angestoßen. Ergibt die logische Verknüpfung erneut '1', während das Pattern gerade läuft, dann wird das Pattern beendet und der Pin verbleibt auf seinem letzten Pegel. Ein erneuter Anstoß des Pattern, der auch per Befehl erfolgen kann, startet das Pattern von vorne.

Pattern pause

Das Pattern wird in diesem Modus durch die logische Verknüpfung pausiert und fortgesetzt. Wenn die logische Verknüpfung positiv ausgewertet wird, während der Pattern-Modus 'Off' ist, dann wird das Pattern angestoßen. Ergibt die logische Verknüpfung erneut '1', während das Pattern gerade läuft, dann wird das Pattern pausiert und der Pin verbleibt auf seinem letzten Pegel. Ein erneuter Anstoß des Pattern, der auch per Befehl erfolgen kann, setzt das Pattern fort.

Befehle

`set pin [DOX|ALL]`

Der Pin wird auf '1'- Pegel gesetzt. Im Pin-Mode 'Logic output' ist ein Setzen des Pins nicht möglich, wenn eine logische Verknüpfung für den Pin definiert ist.

```
set pin do1, do3
DO1 = 1
DO3 = 1
```

`reset pin [DOX|ALL]`

Der Pin wird auf '0'-Pegel gesetzt. Im Pin-Mode 'Logic output' ist ein Rücksetzen des Pins nicht möglich, wenn eine logische Verknüpfung für den Pin definiert ist.

```
reset pin do2
DO2 = 0
```

toggle pin [DOX|ALL]

Ist der Pin auf '0'-Pegel, so wird der Pin auf '1'-Pegel gesetzt und umgekehrt. Im Pin-Mode 'Logic output' ist ein toggeln des Pins nicht möglich, wenn eine logische Verknüpfung für den Pin definiert ist. Es wird der neue Wert als Zustandsangabe zurückgegeben.

```
toggle pin do1, do2
DO1 = 0
DO2 = 1
```

set output [XXXXXXXX]

Mit diesem Befehl können alle acht digitalen Ausgangspins gleichzeitig verändert werden. Jedem Pin wird ein Parameter zugeordnet, der diesen beeinflusst. Damit ergibt sich ein Bitfeld aus acht Zeichen. Die Zuordnung erfolgt in aufsteigender Reihenfolge: DO1 wird das erste Zeichen zugewiesen, DO8 das letzte Zeichen.

- `'1'`: Der Pin wird gesetzt
- `'0'`: Der Pin wird zurückgesetzt
- `'T'`, `'t'`: Der Pin wechselt seinen Zustand
- `'X'`, `'x'`: Der Pin wird nicht beeinflusst

Ist ein Pin im Modus 'Logic output', so ist eine Änderung des Pegels nicht möglich, wenn eine logische Verknüpfung für diesen Pin definiert ist. Die neuen Zustände der Ausgangspins werden in aufsteigender Reihenfolge zurückgegeben.

```
set output 10XX0T1T
DO1 to DO8: 10011011
```

get output

Die Pegel der acht digitalen Ausgangspins werden in aufsteigender Form zurückgegeben.

```
get output
DO1 to DO8: 10011011
```

status [DOX|ALL]

Dieser Befehl listet alle Informationen zu den angegebenen Pins auf.
Angegeben werden jeweils der Pegel des Pins, die logische und die zeitliche Verknüpfung (Logic, Pattern), den Pin-Mode und der Pattern-Modus.

```
status do4
DO4 = 1  Pattern: ON   Pin Mode: Pattern start
DO4 <- DI1
DO4: ( { 100 ms | 101001001001010001110 } )repeat[84]
```

read [DOX|ALL]

Der Pegel des Pins wird zurückgegeben.

```
read d03
DO3 = 1
```

write [DOX = 1|0]

Mit diesem Befehl können in einem Verarbeitungsschritt mehrere Pins auf unterschiedliche Pegel gesetzt werden. Dem Pin muss entweder `'= 1'` oder `'= 0'` folgen, um diesem einen definierten Pegel zuzuweisen. Leerzeichen werden übersprungen. Im Pin-Mode 'Logic output' ist ein toggeln des Pins nicht möglich, wenn eine logische Verknüpfung für den Pin definiert ist. Es wird der neue Pegel als Zustandsangabe zurückgegeben.

```
write do3 = 1, do5 = 0, do1 = 1
DO1 = 1
DO3 = 1
DO5 = 0
```

enable user [DOX|ALL]

Der Pin wird in den Pin-Mode 'User' versetzt und kann somit nur durch Befehle gesetzt oder zurückgesetzt werden.

```
enable user do4
Enabled User DO4
```

enable logic output [DOX|ALL]

Der Pin wird in den Pin-Mode 'Logic output' versetzt und folgt nun dem Ergebnis der logischen Verknüpfung in Echtzeit.

```
enable logic output do1
Enabled Logic output DO1
```

enable logic toggle [DOX|ALL]

Der Pin wird in den Pin-Mode 'Logic toggle' versetzt. Die positive Auswertung der logischen Verknüpfung setzt den Pin, wenn dieser auf '0'- Pegel ist, oder setzt ihn zurück, wenn dieser auf '1'- Pegel ist.

```
enable logic toggle do1
```

```
Enabled Logic toggle DO1
```

enable pattern user [DOX|ALL]

Der Pin wird in den Pin-Mode 'Pattern user' versetzt und kann somit nur noch per Befehl gestartet beziehungsweise gestoppt werden.

```
enable pattern user do1
Enabled Pattern user DO1
```

enable pattern start [DOX|ALL]

Der Pin wird in den Pin-Mode 'Pattern start' versetzt. Das Pattern wird durch die logische Verknüpfung bei positiver Auswertung gestartet.

```
enable pattern start do8
Enabled Pattern start DO8
```

enable pattern toggle [DOX|ALL]

Der Pin wird in den Pin-Mode 'Pattern toggle' versetzt. Das Pattern wird durch die logische Verknüpfung bei positiver Auswertung gestartet oder gestoppt, wenn das Pattern bereits läuft. Dabei wird das Pattern immer von Vorne begonnen.

```
enable pattern toggle ro2
Enabled Pattern toggle RO2
```

enable pattern pause [DOX|ALL]

Der Pin wird in den Pin-Mode 'Pattern pause' versetzt. Das Pattern wird durch die logische Verknüpfung bei positiver Auswertung pausiert beziehungsweise an selber Stelle fortgesetzt.

```
enable pattern start do8
Enabled Pattern start DO8
```

show pin mode [DOX|ALL]

Der Pin-Mode des Zielpins wird zurückgegeben.

- User Der Pin kann nur durch Befehle gesetzt oder zurückgesetzt werden
- Logic output Die logische Verknüpfung steuert den Pin in Echtzeit
- Logic toggle Die logische Verknüpfung setzt den Pin bei positiver Auswertung
- Patten user Die zeitliche Verknüpfung kann nur durch einen Befehl gestartet werden
- Pattern start Das Pattern wird durch die logische Verknüpfung gestartet
- Pattern toggle Das Pattern wird durch die logische Verknüpf und gestartet und gestoppt
- Pattern pause Das Pattern wird durch die logische Verknüpfung pausiert und fortgesetzt

```
show pin mode di1, di8, do1
DI1 = Input
DI8 = Pattern start
DO1 = Logic output
```

set logic [DOX = Logic]

Dem Zielpin wird eine logische Verknüpfung zugeordnet und bei korrekter Syntax in normierter Form zurückgegeben. Eine bereits bestehende logische Verknüpfung für diesen Pin wird überschrieben. Der logische Termin wird in eckige Klammern gesetzt, wenn die Verknüpfung im eingestellten Pin-Mode nicht verwendet wird. Bei positiver Syntaxprüfung wird der logische Term in normierter Schreibweise zurückgegeben.

```
set logic do1 = do2 and (not do5 or do3 or not do3)
DO1 = DO2 and ( not DO5 or DO3 or not DO3 )
```

clear logic [DOX|ALL]

Die logische Verknüpfung des Zielpins wird gelöscht und der dafür belegte dynamische Speicher wird wieder freigegeben.

```
clear logic do1
Logic cleared DO1
```

show logic [DOX|ALL]

Die logische Verknüpfung des Pins wird in normierter Schreibweise zurückgegeben. Die Antwort des Boards ist geringfügig abhängig vom Pin-Mode des Pins, so kann der Zuweisungsoperator `'='`, `'<='` oder `'<-'` sein. Der logische Termin wird in eckige Klammern gesetzt, wenn die Verknüpfung im eingestellten Pin-Mode nicht verwendet wird. Ist keine logische Verknüpfung definiert, so wird stattdessen `'L: empty'` zurückgegeben.

```
show logic do1, do4, do5
DO1 <- DO3 and DO8
[DO4 = DO5]
L: empty
```

set pattern [DOX = Pattern]

Dem Zielpin wird eine zeitliche Verknüpfung zugeordnet und wird bei korrekter Syntax in normierter Form zurückgegeben. Die Antwort des Boards ist geringfügig vom Pin-Mode des Zielpins abhängig, so wird die zeitliche Verknüpfung in eckigen Klammern dargestellt, wenn diese im Pin-Mode nicht verwendet wird.

```
set pattern do3 = high wait 3s ({100ms|1010101001})repeat[20] (toggle wait 1s)loop
DO3: high wait 3 s ( { 100 ms | 1010101001 } )repeat[20] ( toggle wait 1 s )loop
```

clear pattern [DOX|ALL]

Die zeitliche Verknüpfung des Pins wird gelöscht und der dafür belegte dynamische Speicher wird wieder freigegeben.

```
clear pattern do2
Pattern cleared DO2
```

show pattern [DOX|ALL]

Die zeitliche Verknüpfung des Pins wird zurückgegeben. Wird diese im Pin-Mode des Pins nicht verwendet, so wird das Pattern in eckigen Klammern dargestellt. Ist kein Pattern für diesen Pin vorhanden wird `'P: empty'` zurückgegeben.

```
show pattern do1, do2, do3
DO1: high wait 3 s low
[DO2: ( { 100 ms | 10101001010 } )loop]
P: empty
```

get pattern mode [DOX|ALL]

Es wird zurückgegeben, ob das Pattern des Pins im Moment läuft. Ist der Pin-Mode nicht 'Pattern user', 'Pattern start' oder 'Pattern toggle', so wird eine Hinweismeldung zurückgegeben.

```
get pattern mode do2, do3, do4
Pattern: DO2 is ON
Pattern: DO3 is OFF
DO4 is not in pattern mode
```

start pattern [DOX|ALL]

Ist der Pin im Modus 'Pattern user', 'Pattern start' oder 'Pattern toggle' so wird die zeitliche Verknüpfung angestoßen. Ansonsten wird eine Hinweismeldung zurückgegeben.

```
start pattern do2, do3
Pattern started DO2
DO3 is not in pattern mode
```

stop pattern [DOX|ALL]

Die zeitliche Verknüpfung des Pins wird gestoppt und bei Wiederaufnahme von Vorne gestartet.

```
stop pattern do2
Pattern stopped DO2
```

pause pattern [DOX|ALL]

Die zeitliche Verknüpfung des Pins wird angehalten und bei Wiederaufnahme an selber Stelle fortgeführt.

```
pause pattern do4
Pattern paused DO4
```

reset

Alle Pins werden auf '0'-Pegel gesetzt und alle logischen und zeitlichen Verknüpfungen werden gelöscht.

```
reset
Factory settings loaded
```

system status

Es werden die Anzahl der belegte und der freien Bytes angegeben, die für die logischen und zeitlichen Verknüpfungen zur Verfügung stehen.

```
system status
Free dynamic memory:  400 bytes
Used dynamic memory:    0 bytes
```

logout

Die Verbindung wird beendet.

```
logout
Logging out
```

Pinbelegung

Die Pinbelegung auf dem Olimex-Board. Pin 1 zeigt in Richtung Stromversorgung und Pin 33 zeigt in Richtung der RS-232-Schnittstelle.

Pin #	Name	Pin #	Name
1	Cam On/OFF	2	Servo On/Off
3	DO5	4	Display On/Off
5	LED Webpage	6	DO6
7	DO7	8	
9		10	
11		12	LED ‘Display’
13	LED ‘Servos’	14	LED ‘Leds’
15	LED ‘Status’	16	
17	3,3 V	18	3,3 V
19	GND	20	
21	DO8	22	Servo X
23	Servo Y	24	DO1
25	DO2	26	DO3
27	DO4	28	
29		30	
31		32	
33		34	