

ONLIBASIC

ONLIBASIC seems to have its origins with an earlier ONLI system developed by Stephen Lea for the control of psychological experiments in Cambridge. Presumably Acorn later adopted this and created their version as an extension to System BASIC.

Initially it looks like ONLIBASIC was intended to be used to provide (digital) stimulus, responding to and timing responses.

The version of ONLIBASIC that we have now also includes support for analogue I/O and graphical output and comes in variants for the System and Atom computers.

Digital Input and Output

ONLIBASIC can interface to up to 32 digital input/output lines arranged as four Ports each with 8 individual lines.

The Ports are designated PORT0 to PORT3, individual Port Lines are numbered 0 to 31:

- PORT0 lines numbered 0 to 7
- PORT1 lines numbered 8 to 15
- PORT2 lines numbered 16 to 23
- PORT3 lines numbered 24 to 31

Each Port Line can be individually configured as an input or output.

As outputs, they may be set high (turned on) and cleared low (turned off) a Port at a time or individually.

As inputs they may be read as ports (high is on and low is off).

PORT0 and PORT1 lines may be treated individually as 'interrupts'. (Strictly speaking they are not interrupts but are regularly polled by an interrupt driven clock).

Analogue Input and Output

ONLIBASIC can interface to a pair of Digital to Analogue Converters (DACs) and an 8-channel Analogue to Digital Converter (ADC).

The DACs are designated DAC0 and DAC1 and the ADC as ADCn, where n is the input channel 0 to 7.

Both the DAC and ADC are 12-bit resolution i.e. a range of 4096 values with a resolution of 2.5mV, this gives a range of voltages of -5.12V to +5.15V:

- 0 represents -5.12V
- 2048 represents 0V
- 4095 represents +5.12V

Clock

A regular 100Hz IRQ interrupt controls the timing elements of ONLIBASIC including the sampling of inputs and the provision of internal software clocks.

ONLIBASIC expects a negatively-going pulse of 150-200µsecs duration at 100Hz to be present on the IRQ line.

Graphics

The System Computer version of ONLIBASIC adds support for graphical plotting on the 4x25 Teletext VDU. This provides a single colour graphical space of 78 x 75 pixels. The Atom version includes the COLOUR statement from Floating Point BASIC enabling colour graphics.

Hardware

Digital Input and Output

ONLIBASIC expects to see an INS8154 Peripheral Interface Adapter (PIA) device at one of two addresses: \$0900 for PORT0 & PORT1 and \$1000 for PORT2 & PORT3.

The port addresses are allocated as follows:

- PORT0 PA0 to PA7 Port Lines 0 to 7 at \$0900
- PORT1 PB0 to PB7 Port Lines 8 to 15 at \$0900
- PORT2 PA0 to PA7 Port Lines 16 to 23 at \$1000
- PORT3 PB0 to PB7 Port Lines 24 to 31 at \$1000

The INS8154 at \$0900 can be afforded by either the optional second INS8154 IC8 on the 6502 CPU Board or IC36 on a Laboratory Interface Board. A second INS8154 for PORT2 and PORT3 can be provided by a further Laboratory Interface Board (with address configuration links changed from the default).

The advantage of using a Laboratory Interface Board is that the individual inputs and outputs are optically isolated from the computers power. Outputs are able to drive 3 Amps at up to 48 Volts whilst inputs can be driven from a contact closure and LEDs display the state of each line.

This board also produces a 100Hz clock which serves as the ONLIBASIC timing clock.

Analogue Input and Output

ONLIBASIC expects to see an INS8255 PIA at address \$1A00 and a MC6522 Versatile Interface Adapter (VIA) at address \$1A30.

The DACs are connected to the ports of the INS8255 as follows:

- DAC0 bit 0 (LSB) to bit 7 PC0 to PC3
- DAC0 bit 8 to bit 11 (MSB) PA0 to PA7
- DAC1 bit 0 (LSB) to bit 7 PC4 to PC7
- DAC1 bit 8 to bit 11 (MSB) PB0 to PB7

The ADC is connected to the MC6522 as follows:

- ADC bit 0 (LSB) to 7 PB0 to PB3
- ADC bit 8 to 11 (MSB) PA0 to PA7
- ADC Clock PB7
- ADC Start Conversion CA2
- ADC Conversion Complete CA1
- ADC Input Multiplexer PB4 to PB6

The Analogue Interface Board has a pair of National Semiconductor DAC1220 and a single National Semiconductors ADC1210 with an 8-channel input multiplexer (CD4051).

Timing

Up to eight independent 32-bit software clocks may be set to timeout after an interval specified to the nearest 10 milliseconds (1 centisecond). The maximum time that may be specified is $2^{31}-1$ centiseconds (about 248 days).

The current time in centiseconds since the last START Statement (see below) is held in the 32-bit variable TIME. If TIME is read on two occasions, the difference will give the time elapsed in between. If TIME exceeds $2^{31}-1$, it will go to -2^{31} and then increment through 0 again. Even if one or both of the TIME readings is negative, the difference will always be positive and correct, provided it does not exceed 248 days.

Interrupts

An ONLIBASIC program can be conceived of as a series of definitions of alternative outcomes, separated by HANGUP statements, during which the interpreter stops while it discovers if events have occurred. Events are captured on every IRQ pulse, but are not acted upon until the next HANGUP statement is reached.

Control of an interrupt is requested (see RQIN below), a clock is started (see RCLK below) and then a HANGUP is specified.

If the clock times out before a response is detected, the interpreter will jump to the line specified in the RCLK Statement and deal with the programmed outcome; this includes relinquishing control of the interrupt (see REL below). Since the interpreter takes time to reach the next HANGUP, an interrupt might occur before control can be relinquished. This interrupt will be noticed on the next IRQ pulse, but nothing can happen until the HANGUP is reached. Thus, the interrupt will be cancelled by the REL Statement, even though it may have occurred in real time before control was actually relinquished. In this way, impossible situations cannot arise.

ONLIBASIC Statements and Functions Summary

Statements

CLEAR<colour>	Prepares the System Teletext VDU screen for graphics providing an 78x75 bitmap display (System version only)
CLI	Enables Interrupts
CLR<port_line>	Clears an individual Port Line
COLOUR<colour>	Prepares the Atom screen for colour graphics (Atom version only)
DAC<dac_channel>=<dac_value>	Writes a value to a DAC channel
DEFPORT<port>=<direction>	Defines the individual lines of a Port as input or output
DRAW <x>,<y>	Draws a line on the System Teletext VDU screen (System version only)
HANGUP	Halts BASIC execution and waits until either a digital input triggers an interrupt or a clock times out
KILL	Releases all clock and input events
MOVE<x>,<y>	Moves the cursor position on the System Teletext VDU screen (System version only)
PLOT<mode>,<x>,<y>	Plots on the System Teletext VDU screen (System version only)
PORT<port>=<port_value>	Writes to a digital Port
RCLK<clock>,<timeout>,<line_number>	Generates an event when a Clock expires
REL<event>	Release a specific event, either an input Port Line interrupt or Clock timeout
RQIN<polarity>,<port_line>,<line_number>	Causes a jump to <line_number> when input <port_line> changes state as defined by <polarity>
SEI	Disables Interrupts
SET<port_line>	Sets an individual output Port line
START	Initialises the system and sets TIME to 0
TIME<time>	Set the current value of TIME

Functions

ADC<adc_channel>	returns a value from an ADC channel
PORT<port>	returns the input state of a Port
REACT<pattern>,<pattern>,<time>,<time>	measures a reaction time
TIME	returns the current value of TIME

Parameter Definitions

<adc_channel>	an ADC input channel value 0 to 7
<clock>	internal clock number between 0 to 7
<colour>	either 0 to 6 on a System Computer representing: 0:white, 1:cyan, 2:magenta, 3:blue, 4:yellow, 5: green, 6:red or 0 to 3 on an Atom computer representing: 0:green, 1:yellow, 2:blue, 3:red
<dac_channel>	a DAC output channel value 0 or 1
<dac_value>	0 to 4095 representing -5.12V to +5.12V
<direction>	an 8-bit mask defining the direction of individual Port Lines, 0: input and 1: output
<event>	values between 0 and 15 refer to an input Port Line, whilst a value between 16 and 23 refers to a Clock 0 to 7
<line_number>	a BASIC line number or label
<mode>	see PLOT Statement below
<pattern>	either a 16 or 32 bit pattern of input Port Lines
<polarity>	defines the transition direction of an input which triggers an interrupt, 0 refers to a high to low transition whilst 1 refers to low to high
<port_line>	a digital input/output Port Line 0 to 31
<port_value>	an 8-bit value defining the state of the 8 lines of a Port (assuming they are configured as output)
<port>	a digital input/output Port 0 to 3
<time>	any valid BASIC integer value (it will even accept a negative value, it's actually a 4-byte value which wraps around when it reach's \$FFFF so becomes a negative BASIC value after \$7FFF)
<timeout>	clock run time in centiseconds
<x>	horizontal position on the System Computer Teletext graphics screen between 0 and 77 (0 if far left)
<y>	vertical position on the System Computer Teletext graphics screen between 0 and 74 (0 is bottom of screen)

ONLIBASIC Statements and Functions (alphabetical order)

ADC Function

Returns a value from an ADC channel.

Syntax: <result> = ADC<adc_channel>
where <adc_channel> is in the range 0 to 7

Returns a 12-bit value between 0 and 4095 representing -5.12V to +5.12V in 2.5mV steps

Example:

X=ADC(0)

Must be preceded by a START statement to enable the ADC clock.

CLEAR Statement (System version only)

Prepares the System Teletext screen for graphics giving a 78x75 bitmap display.

Syntax: CLEAR <colour>
where <colour> is one of the following:
1 = red
2 = green
3 = yellow
4 = blue
5 = magenta
6 = cyan
7 = white

Writes a Teletext graphics colour code to the first character position of each of the 25 screen rows in preparation for drawing graphics.

Example:

CLEAR 1

CLI Statement

Enables interrupts, takes no parameters.

Syntax: CLI

Clears the interrupt flag in the processor status register enabling the 100Hz clock, equivalent to machine code \$58.

CLR Statement

Clear an individual output Port Line.

Syntax: CLR<port_line>
where <port_line> is in the range 0 to 31

Turns off Port Line <port_line>, if the Port Line is already off or is defined as an input nothing happens.

Examples:

CLR8	clears Port Line 8 i.e. Port1 bit 1
CLRA	clears the Port Line defined by variable A
CLR(Z+2)	clears the port Line defined by variable Z+2

COLOUR Statement (Atom version only)

This works on the Atom's bitmapped display in the same way as COLOUR <mode> in the Atom Floating Point ROM.

Syntax: COLOUR <colour>
where <colour> is one of the following:

- 0 = green
- 1 = yellow
- 2 = blue
- 3 = red

Installs a colour mode specific Plot function

Example:

COLOUR(3)

DAC Statement

Writes a value to a DAC channel.

Syntax: DAC <dac_channel>=<dac_value>
where <dac_channel> is either 0 or 1
<dac_value> is a 12-bit value between 0 and 4095 representing -5.12V to +5.12V in

2.5mV steps

Examples:

DAC0=0	sets the analogue output on DAC0 to -5.12V
DAC1=2048	sets the analogue output on DAC1 to 0V
DAC0=A	sets the analogue output on DAC0 to the value stored in variable A

DEFPORT Statement

Sets the direction of individual Port Lines on 8-bit port.

Syntax: DEFPORT<port> = <direction>
where <port> is in the range 0 to 3
<direction> is an 8-bit mask, with 0 for input and 1 for output

Each line of the Port is defined as an input if the corresponding bit in <direction> is 0 or as output if it is 1.

Examples:

DEFPORT0=0	Port Lines 0 to 7 are all defined as inputs
DEFPORTA=255	all Port Lines of Port variable A are defined as outputs
DEFPORT1=\$FO	Port Lines 8 to 11 are defined as inputs, port Lines 12 to 15 as outputs

DRAW Statement (System version only)

Draws a line on the System Teletext display.

Syntax: DRAW <x>, <y>

where <x> is between 0 and 77

<y> is between 0 and 74

This Statement is the same as PLOT 5, <x>, <y>

Example (draws a purple frame):

```
CLEAR 2
```

```
MOVE 0,0
```

```
DRAW 0,74
```

```
DRAW 77,74
```

```
DRAW 77,0
```

```
DRAW 0,0
```

HANGUP Statement

Halt BASIC execution until an event occurs, takes no parameters.

Syntax: HANGUP

Halts the BASIC interpreter and waits until an input interrupt to occur or a clock times out.

KILL Statement

Releases all clock and input events.

Syntax: KILL

Cancels all requests for control of interrupts and clocks.

MOVE Statement (System version only)

Moves graphics cursor position on the System Teletext display:

Syntax: MOVE <x>,<y>

where <x> is between 0 and 77

<y> is between 0 and 74

This Statement is the same as PLOT 12, <x>, <y>

Examples:

```
MOVE 0,0
```

moves to the origin, bottom left corner of the screen

```
MOVE 77,74
```

moves to the top right corner of the screen

PLOT Statement (System version only)

Plots on the System Teletext display.

Syntax: PLOT <mode>,<x>,<y>

where <mode> is one of the following:

- 0: move relative to last position
- 1: draw coloured line relative to last position
- 2: invert colour of line relative to last position
- 3: draw black line relative to last position
- 4: move to absolute position
- 5: draw coloured line to absolute position
- 6: invert colour of line to absolute position
- 7: draw black line to absolute position
- 8: move relative to last position
- 9: plot coloured point relative to last position
- 10: invert colour of point relative to last position
- 11: plot black point relative to last position
- 12: move relative to last position
- 13: plot coloured point at absolute position
- 14: invert colour of point at absolute position
- 15: plot black point at absolute position

and

<x> is in the range 0 to 77

<y> is in the range 0 to 74

Examples:

PLOT 4,0,0	moves to the origin, bottom left corner of the screen
PLOT 1,10,10	draw a coloured line between the current position and +10x and +10y pixels
PLOT 10,30,40	inverts the pixel at 30,30 i.e. if black it becomes coloured, if its coloured it becomes black (clear)

PORT Statement

Sets the value of an 8-bit port.

Syntax: PORT <port>=<port_value>

where <port> is in the range 0 to 3

<port_value> is the 8-bit value to be written

Each line defined as an output in Port <port>, is turned on if the corresponding bit in the lowest byte of <port_value> is set, and turned off if it is clear. Lines defined as inputs are unaffected.

Examples:

PORT0=0	all outputs on PORT0 are turned off
PORTB=255	all outputs on PORT variable B are turned on
PORT1=\$0F	Port Lines 8 to 11 are turned ON and port Lines 12 to 15 turned OFF

PORT Function

Returns the current value of a Port's inputs.

Syntax: <result> = PORT<port>

where <port> is in the range 0 to 3

Returns an 8-bit <result> containing the state of each of the inputs on the specified <port>.

Examples:

A=PORT0

returns the state of the inputs on Port0

A?3=PORTZ

returns the state of the inputs on Port Z and stores them at address A+3

RCLK Statement

Generate an event when a clock expires.

Syntax: RCLK <clock>,<timeout>,<line_number>

where <clock> is a value between 0 and 7

<timeout> is a time in centiseconds

<line_number> can be a BASIC line number or label

<clock> is set running, and will cause a jump to line <line_number> after <timeout> centiseconds.

The time for which the clock runs will be x, where <timeout> >x> <timeout> -1. A clock requested for 0 centiseconds will be abandoned.

If <clock> is already running, the new values of <timeout> and <line_number> supersede the old ones.

Example (prints a message every 2 seconds):

10 START

20 RCLK 0,200,40

30 HANGUP

40 PRINT "CLOCK TIMEOUT"

50 GOTO 20

REACT Function

Measure Reaction Time.

Syntax: <result> = REACT <pattern_2>,<pattern_1>,<time_2>,<time_1>
where <pattern_1>: 32-bit stimulus output pattern
<pattern_2>: 32-bit pattern / 16-bit reaction input pattern
<time_1>: 32-bit count for timeout in units of 200μseconds
<time_2>: 32-bit counter for output event in units of 200μseconds

This function is designed for reaction timing. A pattern of outputs (the stimulus) is set for a defined time, the function then waits for a second defined time for a specified input to occur (the reaction), whereupon it returns the time between setting the outputs and the detected input with a resolution of 200μseconds.

<pattern_1> and <pattern_2> are 32-bit values representing the four Ports, with their lowest byte as PORT0. They provide the output patterns for the two phases of the function, the least significant 16-bits of <pattern_1> define the reaction input(s).

Inputs and Outputs must have been previously defined by DEFPORT.

- On entry, the interrupt flag is set, disabling the 100Hz clock, and thus preventing any interrupts from being detected and stopping all running clocks.
- All lines defined as outputs are set or cleared according to the bits in <pattern_1>.
- If any input line on Ports 0 and 1, whose bit is set in <pattern_1>, goes high, the number of 200μseconds cycles elapsed is returned in <result>.
- Otherwise, after <time_1> the outputs are changed to the bit-pattern in <pattern_2>.
- The interrupt flag is cleared re-enabling interrupts.

If no response is detected after <time_2> cycles, the function returns with <result> = <time_2>.

If any of the responses specified in <pattern_2> are detected immediately after the outputs have been set, the function returns with <result> = 0.

If the reaction time is less than the stimulus duration <time_1>, the outputs will be in the state specified by <pattern_1> on return.

If <time_1> is made equal to <time_2> and no response is detected, the outputs will be in the state specified by <pattern_1> on return.

Examples:

DEFPORT0=16

R=REACT0,31,500,100

line 4 is turned on for 1 second, valid responses are lines 0,1,2 and 3, the response must occur within 5 seconds

DEFPORT0=48

R=REACT32,17,200,50

line 4 is turned on for 500mS and then turned off with line 5 turned on, only line 0 is a valid response, and it must occur within 2 seconds

REL Statement

Release a specific event.

Syntax: REL <event>

where <event> values between 0 and 15 are input events and 16 to 23 are clock events (0 to 7)

Relinquishes control of an input (following RQIN) or abandons a clock (following RCLK). A value of <event> between 0 and 15 causes the appropriate input line to be ignored. A value of <event> between 16 and 23 abandons clock number <event>-16 (i.e. a clock in the range 0 to 7).

If the Port Line is defined as an output, or control of the interrupt or clock has not been requested, nothing happens.

Examples:

RELO

RELZ

REL(A!Z)

RQIN Statement

Generate an event when an input changes state.

Syntax: RQIN <polarity>,<port_line>,<line_number>

where <polarity> defines the direction of the input transition

<port_line> defines the Port Line to monitor

<line_number> is a BASIC line number or label

If the least significant bit of <polarity> is 1, a transition from low to high will cause a jump to line <line_number>. If the least significant bit of <polarity> is 0, a transition from high to low will cause a jump.

Once control of an interrupt has been requested, each transition of the direction specified by <polarity> will cause a jump until control is relinquished with a REL or KILL Statement.

If <port_line> is defined as an output, nothing happens. If control of <port_line> is already requested, the new values of <polarity> and <line_number> supersede the old ones.

Example (prints a message each time Port Line 0 transitions from low to high):

10 START

20 DEFPORT0=0

30 RQIN1,0,50

40 HANGUP

50 PRINT "Triggered"

60 GOTO 40

SEI Statement

Disables interrupts, takes no parameters.

Syntax: SEI

Sets the interrupt flag in the processor status register preventing the detection of interrupts and freezing clocks and TIME, equivalent to machine code \$78.

SET Statement

Set an individual output Port Line.

Syntax: SET<port_line>
where <port_line> is in the range 0 to 31

Turns on Port Line <output>, if the line is already on, or defined as an input, nothing happens.

Examples:

SET8	Sets Port Line 8
SETX	Sets Port Line X

START Statement

Initialises the system and sets TIME to 0, takes no parameters.

Syntax: START

Clears the interrupt flag and starts the centisecond clock running from 0. Relinquishes all interrupts and abandons clocks, resetting them to 0. This statement must always be used after entry into BASIC.

TIME Statement

Set the current value of TIME.

Syntax: TIME <time>
where <time> is an integer value

The centisecond clock is set to the value of <time>.

Examples:

```
TIME=T
TIME=TIME+T/50
```

TIME Function

Returns the current value of TIME.

Time is incremented by the 10mS timer interrupt

Syntax: <result> = TIME

<result> is set to the current value of the centisecond clock.

Examples:

```
A=TIME
A=1000-TIME
A=TIME/100
```

Error Messages (in addition to those from BASIC)

Error 149	Invalid parameter
Error 243	Invalid line number
Error 29	Unknown or missing function
Error 13	Invalid parameter