ONLIBASIC

The ONLI extension to BASIC provides commands and functions for real-time work, including timed intervals, control of outputs and detection of external events. ONLIBASIC expects a negatively-going pulse of 150-200 microseconds duration at 100 Hz to be present on the IRQ line (pin 28B) of the processor card, and an 8254 or 8154 I/O chip at page 09. This chip may be positioned as IC8 on the processor card, and the lines accessed via the backplane, or may be provided together with the 100Hz IRQ circuitry on Acorn's Laboratory Interface Board. The I/O chip provides two ports of eight lines each, known as PORT0 and PORT1, and as individual lines with numbers 0 to 7 (PORT0) and 8 to 15 (PORT1). These lines may be defined individually as inputs or outputs. As outputs, they may be set high (turned on) and cleared low (turned off) port at a time or individually. As inputs they may be read as ports (high is on, and low is off), or treated individually as "interrupts". The two ports are polled on every IRQ pulse, and so "interrupts" must last at least 10 milliseconds to be detected.

A second 8154 or 8254 may be present at page 10. This provides two further ports, known as PORT2 (lines 16 to 23) and PORT3 (lines 24 to 31). These lines may be treated in the same way as those in ports 0 and 1, with the exception that they may not be used as interrupts.

In addition, up to eight independent 32-bit software clocks may be set to time out 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 command (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.

An ONLIBASIC program can be conceived of as a series of definitions of alternative outcomes, separated by HANGUP statements (see below), during which the interpreter stops while it discovers if events have occurred. Events are noted on every IRQ pulse, but are not acted upon until the next HANGUP statement is reached. For example, it is required that a response must occur within a given interval. 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 command 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 command, even though it may have occurred in real time before control was actually relinquished. In this way, impossible situations cannot arise.

ONLI Instructions

CLI - command.

Syntax: CLI

Clears the interrupt flag in the processor status register, enabling

the 100 Hz IRQ clock. Equivalent to machine code £58.

CLR - command.

Syntax: CLR ⟨factor⟩

Turns off I/O line number ⟨factor⟩, 0⟨= ⟨factor⟩ ⟨=31. If the line is already off or is defined as an input nothing happens.

Examples: CLR8
CLRA
CLR(Z+2)
CLR(A!Z)

DEFPORT - command.

Syntax: DEFPORT ⟨factor⟩ = ⟨expression⟩

Defines I/O port ⟨factor⟩, 0⟨= ⟨factor⟩ ⟨=3; each line in the port is defined as an input if the corresponding bit in the lowest byte of ⟨expression⟩ is clear, and as an output if it is set.

Examples: DEFPORT0=0 (lines 0 to 7 are all inputs)
DEFPORTA=255 (PORTA is to be all outputs)
DEFPORT1=£F0 (lines 8-11 inputs, 12-15 outputs)

HANGUP - command.

Syntax: HANGUP

Stops the interpreter, and waits until an interrupt occurs or a clock times out.

KILL - command.

Syntax: KILL

Cancels all requests for control of interrupts and clocks.

PORT - command.

Syntax: PORT ⟨factor⟩ = ⟨expression⟩

Each line defined as an output in port number ⟨factor⟩, 0⟨= ⟨factor⟩ ⟨=3, is turned on if the corresponding bit in the lowest byte of ⟨expression⟩ is set, and turned off if it is clear. Lines defined as inputs are unaffected.

Examples: PORT0=0 (all outputs in PORT0 are turned off)
PORTB=255 (all outputs in PORTB are turned on)
PORT2=£F (lines 8-11 ON, lines 12-15 OFF)

PORT - function.

Syntax: ⟨variable⟩ = PORT ⟨factor⟩

Each bit in the lowest byte of ⟨variable⟩ is set if the corresponding line in PORT number ⟨factor⟩ is high, and cleared if it is low, regardless of whether it is defined as an input or an output.

Examples: A=PORT0
A?3=PORTZ
FORZ=0TO3;A?Z=PORTZ;NEXT

RCLK - command.

Syntax: RCLK ⟨factor⟩a , ⟨factor⟩b , ⟨go entity⟩

The clock number ⟨factor⟩a, 0⟨= ⟨factor⟩a ⟨=7, is set running, and will cause a jump to line ⟨go entity⟩ after ⟨factor⟩b centiseconds. The time for which the clock runs will be x, where ⟨factor⟩b ⟩x⟩ ⟨factor⟩b -1. A clock requested for 0 centiseconds will be abandoned. If ⟨factor⟩a is already running, the new values of ⟨factor⟩b and ⟨go entity⟩ supercede the old ones.

Examples: RCLK0,1000,a
RCLKY,(A!Z),400
RCLK(Z/4),(A!Z*100),(520+Z)
RCLKY,(T*100),(C!X)

REACT - function.

Syntax: ⟨variable⟩= REACT ⟨factor⟩a , ⟨factor⟩b , ⟨factor⟩c , ⟨factor⟩d

This function is designed for reaction timing. A pattern of outputs is set (the stimulus) for a time specified in units of 200 microseconds, and the function waits for a limited time for one of a specified set of inputs to occur (the reaction), whereupon it returns the time between setting the outputs and the detected input, accurate to 200 microseconds. ⟨factor⟩a and ⟨factor⟩b are 32-bit numbers, each representing all four ports, with their lowest bytes as PORT0. ⟨factor⟩c and ⟨factor⟩d are times, specified in units of 200 microseconds. On entry, the interrupt flag is set, disabling the 100 Hz 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 corresponding bits in ⟨factor⟩b. As soon as any input line, whose corresponding bit is set in ⟨factor⟩b, goes high, the number of 200-microsecond cycles elapsed is returned in ⟨variable⟩. After ⟨factor⟩d cycles (the stimulus duration) the outputs are changed to the corresponding bit-pattern in ⟨factor⟩a. If no response is detected after ⟨factor⟩c cycles, the function returns with ⟨variable⟩ = ⟨factor⟩c. If any of the responses specified in ⟨factor⟩a is detected immediately after the outputs have been set, the function returns with ⟨variable⟩ = 0. The interrupt flag is cleared on return. If the reaction time is less than the stimulus duration (⟨factor⟩d), the outputs will be in the state specified by ⟨factor⟩b on return. If ⟨factor⟩d is made equal to ⟨factor⟩c, and no response is detected, the outputs will be in the state specified by ⟨factor⟩a on return. Since the centisecond clock will have been stopped for the duration of the command, it may be updated by the use of the command TIME (see below); eg. TIME = TIME + ⟨variable⟩ /50.

Examples

R=REACT0,31,25000,5000 (line 4 is turned on for 1sec; valid responses are lines 0,1,2 and 3; the response must occur within 5 secs).

A!Z=REACT32,17,10000,50 (line 4 is turned on for 1 centisec and then turned off with line 5 turned on; only line 0 is a valid response, and it must occur within 2 secs).

T=REACT£10,£2F,X,(B!Y) (line 5 is turned on for B!Y cycles, and then changed to line 4; any of lines 0,1,2 or 3 is a valid response, which must occur within X cycles)

REL - command.

Syntax: REL <factor>

Relinquishes control of an interrupt (following RQIN) or abandons a clock (following RCLK). A value of <factor> between 0 and 15 causes the appropriate I/O line to be ignored. A value of <factor> between 16 and 23 abandons clock number <factor>-16 (ie. the corresponding clock in the range 0 to 7). If line number <factor> is defined as an output, or control of the interrupt or clock has not been requested, nothing happens.

Examples: REL0
REL2
REL(A!Z)

RQIN - command

Syntax: RQIN <factor>a , <factor>b , <go entity>

Control of an interrupt on line number <factor>b is requested. If the least significant bit of <factor>a is set, a transition from low to high will cause a jump to line <go entity>. If the least significant bit of <factor>a is clear, 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 <factor>a will cause a jump until control is relinquished with a REL or KILL command. If <factor>b is defined as an output, nothing happens. If control of <factor>b is already requested, the new values of <factor>a and <go entity> supercede the old ones.

Examples: RQIN1,0,a
RQIN0,Y,390
RQINB,(A!Z),(460+Z)
RQIN(Z/4),(C!X/2),(J!Z)

SEI - command

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 - command

Syntax: SET <factor>

Turns on I/O line number <factor>, 0<= <factor> <=31. If the line is already on, or defined as an input, nothing happens.

Examples: SET8
SETX
SET(Z+6)
SET(A!Z)

START - command

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 command must always be used after entry into BASIC.

TIME - command

Syntax: TIME = <expression>

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

Examples: TIME=T
TIME=TIME+T/50
TIME=A!Z*100+B!Z*6000

TIME - function

Syntax: <variable> = TIME

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

Examples: A=TIME
A=1000-TIME
A=TIME/100
A!Z=TIME-X