

OS COMMAND REFERENCE



[THE BLAZINGCORE SERIES]

WITH SUPERIOR NUMBER CRUNCHING ABILITIES AND PERIPHERAL HANDLING ON OUR CUSTOM EMBEDDED OS,
RAPID PROTOTYPING IS NOW EASY... AND BLAZING FAST.

OS

The following members listed under OS contain methods that allow the BCore Operating System to natively handle peripheral, memory and communication protocols of components onboard the BCore Microcontroller Board.

Members	Description
Comm1	Handles methods for serial communication on Comm1
Comm2	Handles methods for serial communication on Comm2
Idle	Set the OS to Idle mode for a specified period of time
Interrupt	Provides access to methods for the Interrupt Pins
LED1	Provides methods to toggle the onboard LED1 on or off
LED2	Provides methods to toggle the onboard LED2 on or off
Memory	Provides access to methods for handling data arrays stored in memory
PB1	Returns logical state of Push Button 1 onboard the BCore Board
PB2	Returns logical state of Push Button 2 onboard the BCore Board
Reset	Commands the BlazingCore OS to perform a soft reset on itself.
Sleep	Set the OS to Sleep mode for a specified period of time
Timer	Provides access to system timer methods

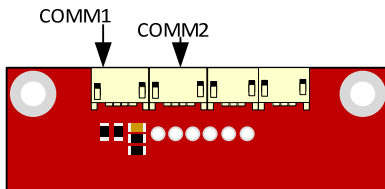
UART

Serial communication on the BCore is available through the use of UART.

The BCore board has up to 4 UARTs supported by the BCoreOS under the name of Comm1/2/3/4.

UART settings on the BCore are;

- Full duplex, 8bit transmission
- No parity, One Stop Bit
- Baud Rate: 115200bps (Default)
- 4 word deep FIFO Data Receive Buffer
- COMM1 has a fixed baud rate of 115200bps, 8N1. More options are available for COMM2/3/4 under the namespace of BCore.



COMM1 / COMM2

Handles methods for serial communication

Methods	Description
HasData	Returns state of Data Receive buffer
Rx	Returns byte data stored in Data Receive buffer
Tx	Transmit specified byte data.

.HASDATA

Returns state of Data Receive buffer

```
Variable = Comm1.HasData()
Variable = Comm2.HasData()
```

Returns;

- Logical **True** if data is present in the data receive buffer,
- Logical **False** if the buffer is empty.

.RX

Returns byte data stored in Data Receive buffer

*Data is retrieved one byte at a time.

```
Variable = Comm1.RX()
```

```
Variable = Comm2.RX()
```

.TX

Transmit specified byte data

*Transmission is done one byte at a time.

```
Comm1.TX(byte Data)
```

```
Comm2.TX(byte Data)
```

WORKING EXAMPLE

The following code is a working example of the use of the above commands to perform checking, receiving and transmission of data between the 2 COMM Ports.

Data that is received on either port is echoed to the other port.

Tests may be conducted with the help of a HyperTerminal or equivalent software.

Code:

```
'SIMPLE ECHO PROGRAM BETWEEN THE TWO COMM PORTS
Public Sub Main()
Dim RX as integer

    Do
        If OS.Comm1.HasData() Then
            rx = OS.Comm1.RX()
            OS.Comm2.TX(rx)
        End If

        If OS.Comm2.HasData() Then
            rx = OS.Comm2.RX()
            OS.Comm1.TX(rx)
        End If
    Loop

End Sub
```

POWER SAVING MODE

BCore features the following commands to set the core to sleep or idle mode to save on current, thus drawing less power and enabling battery powered applications to run for a decent amount of time on a single charge.

For reference;

Board/s	Mode	Current Consumption
BCore100	Normal	48mA
	Idle	< 1mA
	Sleep	< 1mA

.IDLE

Set the OS to Idle mode for a specified period of time

`OS.Idle`(ByVal Duration As Integer)

Duration: Length of Time to be in Idle Mode (in ms)

.SLEEP

Set the OS to Sleep mode for a specified period of time

`OS.Sleep`(ByVal Duration As Integer)

Duration: Length of Time to be in Idle Mode (in ms)

INTERRUPT

BCore has 3 Interrupt Pins; 2 Interrupt On Change Pins and 1 Timer Interrupt.

Fields	Description
Timer	Provides access to methods for BCore Timer Interrupt
Pin1	Provides access to methods for interrupt on Pin 1
Pin2	Provides access to methods for interrupt on Pin 2

TIMER

Provides access to methods for BCore Timer Interrupt.

Methods	Description
Init	Initialise the Timer Interrupt with a specified interval duration
Start	Start the Timer Interrupt
Stop	Stop the Timer Interrupt
State	Get the State of the Timer Interrupt
Read	Read Time of Timer Interrupt

.INIT

Initialise the Timer Interrupt to trigger at specific intervals, in msec

`OS.Interrupt.Timer.Init`(ByVal Period As Integer)

.START

Start the Timer Interrupt

```
OS.Interrupt.Timer.Start()
```

.STOP

Stop the Timer Interrupt

```
OS.Interrupt.Timer.Stop()
```

.STATE

Get State of Timer Interrupt

```
Variable = OS.Interrupt.Timer.State() As Integer
```

Returns:

'0': Timer Interrupt is Off

'1': Timer Interrupt is On

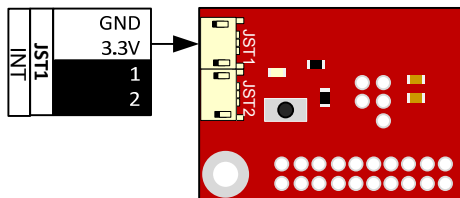
.READ

Get the time elapsed since last interrupt, in msec

```
Variable = OS.Interrupt.Timer.Read() As Integer
```

PIN1 / PIN2

Provides access to methods for BCore's Interrupt-On-Change Pins located on Pins 1 & 2



Methods	Description
Init	Initialise the Interrupt Pin with a specified mode
Start	Start the Interrupt
Stop	Stop the Interrupt
State	Get the State of the Interrupt

.INIT

Initialise the Interrupt Pins to trigger at a specific change in Pin State

```
OS.Interrupt.Pin1.Init(ByVal Mode As Integer)
```

```
OS.Interrupt.Pin2.Init(ByVal Mode As Integer)
```

Mode:

'0': Trigger on Rising Edge

'1': Trigger on Falling Edge

.START

Start the Interrupt

```
OS.Interrupt.Pin1.Start()
```

```
OS.Interrupt.Pin2.Start()
```

.STOP

Stop the Interrupt

```
OS.Interrupt.Pin1.Stop()  
OS.Interrupt.Pin2.Stop()
```

.STATE

Get State of Interrupt

```
Variable = OS.Interrupt.Pin1.State() As Integer  
Variable = OS.Interrupt.Pin2.State() As Integer
```

Returns:

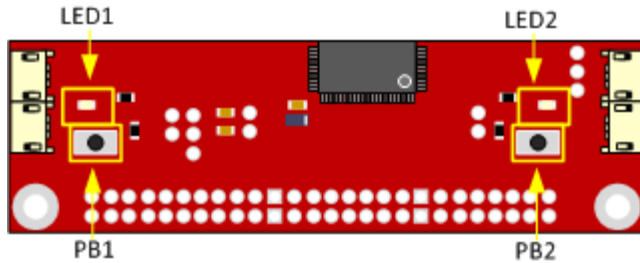
'0': Interrupt is Off

'1': Interrupt is On

.LED1 / LED2

Provides methods to toggle the onboard LED on or off

Methods	Description
High	Turn the LED on
Low	Turn the LED off



.HIGH

Turn the LED on

```
OS.LED1.High()  
OS.LED2.High()
```

.LOW

Turn the LED off

```
OS.LED1.Low()  
OS.LED2.Low()
```

.PB1 / PB2

Returns logical state of onboard push buttons

```
Variable = OS.PB1()  
Variable = OS.PB2()
```

Returns;

- 0; Button is not pressed
- 1; Button is pressed

The following program polls both the onboard buttons for its logical state 5 times a second to detect if any of the buttons are pressed.

Code:

```
Dim BUTTON1, BUTTON2 as Integer  
  
Do  
    BUTTON1 = OS.PB1()  
    BUTTON2 = OS.PB2()  
    Debug.Print Cstr(BUTTON1); " "; Cstr(BUTTON2)  
    Delay(200)  
Loop
```

EXTERNAL MEMORY CARD (MMC/SD-COMPATIBLE)

For the sake of backward compatibility, Sonata IDE still accepts OS.Memcard usages. However, no further developments will be made under `OS.Memcard`.

Please use the namespace `MCard` Instead. (Full listing located further down in this documentation)

MEMORY

Provides access to methods for handling data arrays stored in memory

Methods	Description
Copy	Copies elements from one array to another
CopyExt	Copies a specified range of elements from one array to another
FindFirst	Min Returns index of first occurrence of the smallest number in a specified range of elements
	Max Returns index of first occurrence of the largest number in a specified range of elements

.COPY

Copies elements from one array to another.

`OS.Memory.Copy` (*Source, Destination, Length*)

Parameter/s:

- **Source;** Variable Array to copy from
- **Destination;** Variable Array to copy to
- **Length;** Number of elements to copy over

Code:

```
Dim Arr1(20) as Integer
Dim Arr2(20) as Integer

Public Sub Main()
Dim I as Integer
'Initialise the Array with some values
For I = 0 To 5
    Arr1(I) = I
Next
OS.Memory.Copy(Arr1, Arr2, 3)
For I = 0 To 5
    DEBUG.PRINT CSTR(Arr2(I));", ";
Next
End Sub
```

Output :

```
0, 1, 2, 0, 0, 0,
```

.COPYEXT

Copies elements from one array to another.

`OS.Memory.CopyExt` (*Source, Source Start, Destination, Destination Start, Length*)

Parameter/s:

- **Source;** Variable Array to copy from
- **Source Start;** Index of Source to start copying elements from
- **Destination;** Variable Array to copy to
- **Destination Start;** Index of Destination to start copying elements to
- **Length;** Number of elements to copy over

Code:

```
Dim Arr1(20) as Integer
Dim Arr2(20) as Integer

Public Sub Main()

Dim I as Integer
'Initialise the Array with some values
For I = 0 To 5
    Arr1(I) = I
Next
OS.Memory.CopyExt (Arr1, 2, Arr2, 1, 3)
For I = 0 To 5
    DEBUG.PRINT CSTR(Arr2(I));", ";
Next

End Sub
```

Output :

```
0, 2, 3, 4, 0, 0,
```

.FINDFIRST

Performs a search function into specified data arrays

Methods	Description
Min	Returns index of first occurrence of the smallest number in a specified range of elements
Max	Returns index of first occurrence of the largest number in a specified range of elements

.MIN

Returns index of first occurrence of the smallest number in a specified range of elements

```
Variable = OS.Memory.FindFirst.Min(Memory, Start, Length)
```

Parameter/s:

- **Memory;** Data array to carry out the search
- **Start;** Index of Data Array to start searching from
- **Length;** Number of elements to search

Returns;

- **Index** of element at which the condition was satisfied

Code:

```
Dim Arr1(20) as Integer
Dim Arr2(20) as Integer

Public Sub Main()

Dim I as Integer
'Initialise the Array with some values
For I = 0 To 5
    Arr1(I) = I
Next
I = OS.MEMORY.FINDFIRST.Min (ARR1,0,10)
debug.print "pos = ";cstr(I)

End Sub
```

Output :

```
pos = 0
```

.MAX

Returns index of first occurrence of the largest number in a specified range of elements

Variable = `OS.Memory.FindFirst.Max`(*Memory*, *Start*, *Length*)

Parameter/s:

- **Memory;** Data array to carry out the search
- **Start;** Index of Data Array to start searching from
- **Length;** Number of elements to search

Returns;

- **Index** of element at which the condition was satisfied

Code:

```
Dim Arr1(20) as Integer
Dim Arr2(20) as Integer

Public Sub Main()

Dim I as Integer
'Initialise the Array with some values
For I = 0 To 5
    Arr1(I) = I
Next
I = OS.MEMORY.FINDFIRST.Max(Arr1,0,10)
debug.print "pos = ";cstr(I)

End Sub
```

Output :

```
pos = 5
```

.RESET

Commands the BlazingCore OS to perform a soft reset on itself.

`OS.Reset()`

.SLEEP

Set the OS to Sleep mode for a specified period of time

Note: Refer to Power Saving Mode in this chapter

`OS.Sleep(ByVal Duration As Integer)`

Duration: Length of Time to be in Idle Mode (in ms)

.TIMER

Provides access to control the chip's system timer.

Member	Description
<code>Init()</code>	Initialise OS Timer with a specified prescale value index.
<code>Read</code>	Returns time elapsed in terms of clock cycles (dependent on prescaler)
<code>Set</code>	Set Timer Value
<code>Start</code>	Start the Timer
<code>Stop</code>	Stop the Timer

.INIT

Initialise OS Timer with a specified prescale value index.

`OS.TIMER.INIT(prescale index)`

Prescale Index	Prescale Factor
1	1 (default)
2	2
3	4
4	8
5	16

.READ

Returns time elapsed in terms of clock cycles (dependent on prescaler)

`Variable = OS.TIMER.Read()`

.SET

Set Timer Value to start from

`OS.TIMER.Set(value)`

.START

Start the Timer

`OS.TIMER.Start()`

.STOP

Stop the Timer

`OS.TIMER.Stop()`

TIMER WORKING EXAMPLE

The following is an example code on using the system timer with a prescaler set to a factor of 2.

Code:

```
Dim TIME as Long

Public Sub Main()

    OS.TIMER.INIT(2)
    OS.TIMER.Set(0)
    OS.TIMER.Start()

    TIME = OS.TIMER.Read()
    OS.TIMER.Stop()

    Debug.Print "CYCLES ELAPSED: ";Cstr(TIME)

End Sub
```

LATEST DOCUMENTATION

All of our documentations are constantly updated to provide accurate and/or new information that we feel would help you with developing with our products.

The latest documentation may be obtained from our website: <http://www.aiscube.com/main/downloads.html>

HOW YOU CAN HELP

You can help us to improve our documentations by emailing to us or posting a thread in our forum, reporting any mistakes/typos or errata that you might spot while reading our documentation.

Email: TechSupport@aiscube.com

Forum: <http://forum.aiscube.com/index.php>

DISCLAIMER

All information in this documentation is provided 'as-is' without any warranty of any kind.

The products produced by AIS Cube are meant for rapid prototyping and experimental usage; they are not intended nor designed for implementation in environments that constitute high risk activities.

AIS Cube shall assume no responsibility or liability for any indirect, specific, incidental or consequential damages arising out of the use of this documentation or product.

COPYRIGHT© 2009 - 2011 AIS CUBE. ALL RIGHTS RESERVED.

ALL PRODUCT AND CORPORATE NAMES APPEARING IN THIS DOCUMENTATION MAY OR MAY NOT BE REGISTERED TRADEMARKS OR COPYRIGHTS OF THEIR RESPECTIVE COMPANIES. AND ARE ONLY USED FOR IDENTIFICATION OR EXPLANATION FOR THE OWNER'S BENEFIT. WITH NO INTENT TO INFRINGE.

SONATA IDE AND BLAZINGCORE(BCORE) ARE TRADEMARKS OF AIS CUBE IN SINGAPORE AND/OR OTHER COUNTRIES. ALL IMAGES DEPICTING THE BLAZINGCORE OR ANY PART OF IT IS COPYRIGHTED.

ALL OTHER TRADEMARKS OR REGISTERED TRADEMARKS ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS.