

# **FIDE**

# **System Library**

(v1.3)

© 2008 AIS Cube. All rights reserved.

The FlamingICE(FI) and FIDE are either registered trademarks or trademarks of AIS Cube in Singapore and/or other countries.

Microsoft, Windows, Visual Basic and Visual C# are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Adobe and Acrobat are trademarks of Adobe Systems Incorporated.

## **Brief Overview**

The System Library is documented into 2 major sections, namely **OS** and **Commands**.

The OS section documents the instruction sets used to call on the functions of that which is directly under the FlamingICE™ operating system.

The Commands section documents the instruction sets that are used mainly for interacting with devices, such as sensors, that are attached to specific I/O ports on the FI™ boards.

More details are stated along with the instruction sets as follow.

# OS

## GLCD

### OS.GLCD.Init

Call the OS to initialize the GLCD.

```
OS.GLCD.Init()
```

### OS.GLCD.Colour

Specifies the fore-colour of the fonts used in the GLCD.

```
OS.GLCD.Colour(colour RGB)
```

### OS.GLCD.Back\_Colour

Specifies the back-colour of the fonts used in the GLCD.

```
OS.GLCD.Back_Colour(colour RGB)
```

### OS.GLCD.Plot

Plot graphics onto the GLCD pixel-by-pixel, programmatically. A faster way to set images on to the GLCD would be to use `SetRegion` and `FillRegion`.

```
OS.GLCD.Plot(x coordinate, y coordinate, colour RGB)
```

### OS.GLCD.Fill

Fill the GLCD with the 8-bit RGB colour stated. *Eg. '0' is Black and '255' is white.*

```
OS.GLCD.Fill(colour RGB)
```

### OS.GLCD.SetPosition

Set the starting position of the cursor on the GLCD. Used for text.

```
OS.GLCD.SetPosition(x coordinate, y coordinate)
```

### OS.GLCD.FontSize

Set the font size for writing text on the GLCD.

```
OS.GLCD.FontSize(font size as integer)
```

- Where '0' indicates small fonts and '1' indicates big fonts.

### Example:

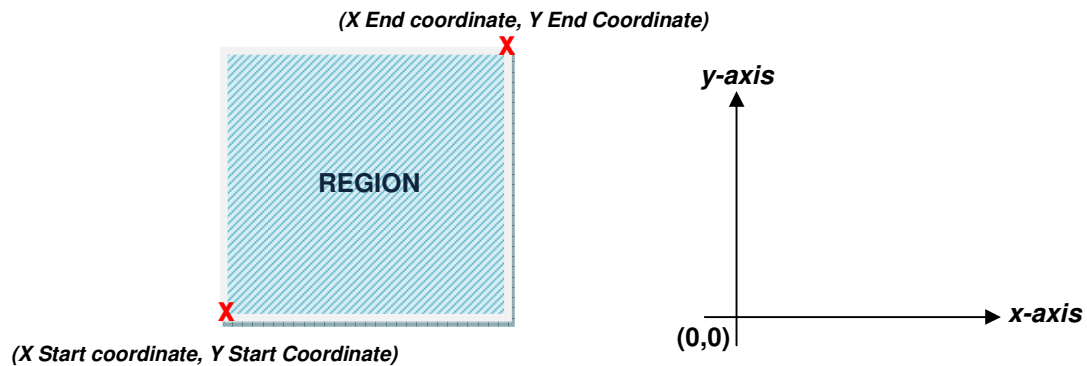
'The following program initializes the GLCD, fills the GLCD with the  
'colour white, sets the position of the cursor to the x,y point (5,10),  
'sets the font size to large, and prints the lines at the cursor  
'position.

```
OS.GLCD.Init()  
OS.GLCD.Fill(255)  
  
OS.GLCD.SetPosition(5, 10)  
OS.GLCD.FontSize(1)  
  
OS.GLCD.Print "TEST 1"  
OS.GLCD.Print "Test 2"
```

### OS.GLCD.SetRegion

Declare rectangular region in the GLCD to be filled with the image that is read from the SD card.

```
OS.GLCD.SetRegion( X Start coordinate, X End coordinate, Y Start  
coordinate, Y End coordinate)
```



### OS.GLCD.FillRegion

Fill Region declared with image that is read from the SD card to the buffer. Max length is 256. If image is larger than 256bytes, use `FillRegion_Cont` to continue filling the region from where it left off.

```
OS.GLCD.FillRegion(start position, length)
```

**Note<sup>1</sup>:** Start position zero-offset.

**Note<sup>2</sup>:** More information about reading from the SD will be stated further into the documentation.

## OS.GLCD.FillRegion\_Cont

Continue filling the region from the last position.

```
OS.GLCD.FillRegion_Cont(start, length)
```

### Example:

'The following program fills the set region with a 32x32 pixel image on the top left of the GLCD.

```
OS.GLCD.Init()  
OS.GLCD.Fill(255)  
  
OS.GLCD.SetRegion(25, 56, 70, 101)  
OS.SD.Read(4, 16)  
OS.GLCD.FillRegion(0, 256)  
OS.SD.Read(4, 17)  
OS.GLCD.FillRegion_Cont(0, 256)
```

## OS.GLCD.Print

Works just like `Debug.Print`. Text is written onto the GLCD using this command.

```
OS.GLCD.Print <string>
```

### Example:

```
Dim I As Integer  
I = 5  
OS.GLCD.Print "Number "; CSTR(I)
```

## TIMER

### OS.TIMER.INIT

Initialize Timer with specified prescaler.

A prescaler takes a frequency and reduces it by a pre-determined factor.

*Eg.* Converting a 1 MHz frequency to a 100kHz frequency (prescale factor of 10).

```
OS.Timer.Init(Prescaler)
```

### OS.TIMER.START

Start the timer.

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

## OS.TIMER.STOP

Stop the timer.

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

## OS.TIMER.READ

Get current value of timer.

```
OS.Timer.Read()
```

## OS.TIMER.SET

Set time to start timer at. (Zero-offset)

```
OS.Timer.Set (Time)
```

### Example:

'The following program initializes the timer with a prescale factor of '1, with variable I at a starting value of 10000, the timer is set to 'run for exactly 1 second each time and the current value of the timer 'after 1 second is read and printed to the debug window.

```
Dim I As Integer
```

```
Public Sub Main()
```

```
Delay 500
```

```
I = 10000
```

```
Debug.Print CStr(I)
```

```
OS.Timer.Init(1)
```

```
Do
```

```
    OS.Timer.Set(0)
```

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

```
    DELAYUS 1000
```

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

```
    I = OS.Timer.Read()
```

```
    Debug.Print CStr(I)
```

```
    Delay 500
```

```
Loop
```

```
End Sub
```

### Output :

```
10000
```

```
2517
```

```
2517
```

```
2517
```

```
2517
```

```
.
```

```
.
```

## I2C

### **OS.I2C.Init**

Initialize the I2C.

```
OS.I2C.Init()
```

### **OS.I2C.Start**

Start the I2C.

```
OS.I2C.Start()
```

### **OS.I2C.ReStart**

ReStart the I2C.

```
OS.I2C.ReStart()
```

### **OS.I2C.Stop**

Stop the I2C.

```
OS.I2C.Stop()
```

### **OS.I2C.Read**

Read from the I2C.

```
OS.I2C.Read(NAK)
```

### **OS.I2C.Write**

Write to the I2C.

```
OS.I2C.Write(value)
```

## SPI

### **OS.SPI.Read**

Read from the Serial Peripheral Interface.

```
OS.SPI.Read()
```

### **OS.SPI.Write**

Write to the Serial Peripheral Interface.

```
OS.SPI.Write(value)
```

## COMM1

Note: COMM1 has a fixed baud rate of 115 200 bps.

### OS.COMM1.TX

Transmit specified data through COMM1 serial port.

```
OS.COMM1.TX(Data)
```

### OS.COMM1.RX

Receive data through COMM1 serial port.

```
OS.COMM1.RX()
```

### OS.COMM1.HasData

Get whether COMM1 has data in buffer as Boolean.

```
OS.COMM1.HasData()
```

#### Example:

```
'The following program constantly checks if there is data to be read in  
'the buffer. If HasData = true, the buffer is read and the data is  
'printed out in the debug window.
```

```
Dim tDat As Integer  
Dim rDat As Integer  
  
If OS.COMM1.HasData() Then  
    rDat = COMM1.RX  
    Debug.Print CSTR(rDat)  
End If
```

## COMM2

### Baud No. Baud Rate(bps)

Baud No.	Baud Rate(bps)
1	9600
2	19 200
3	56 000
4	57 600
5	115 200 (default)

### OS.COMM2.Init

Initialize COMM2 with specified Baud Rate No. (bps)

```
OS.COMM2.Init(Baud No. Range 1-5)
```

## **OS.COMM2.TX**

Transmit specified data through COMM2 serial port.

```
OS.COMM2.TX(Data)
```

## **OS.COMM2.RX**

Receive data through COMM2 serial port.

```
OS.COMM2.RX()
```

## **OS.COMM2.HasData**

Returns whether COMM2 has data in buffer as Boolean.

```
OS.COMM2.HasData()
```

# **PWMSERVO**

## **OS.PWMSERVO.Init**

Initialize PWM for servo connected at the specified pin.

```
OS.PWMSERVO.Init(Pin)
```

## **OS.PWMSERVO.Position**

Set position of the servo at the specified pin.

```
OS.PWMSERVO.Position(Pin, Position)
```

## **OS.PWMSERVO.State**

Toggle state of servo 'On' or 'Off'.

```
OS.PWMSERVO.State(State)
```

State: Where '0' is OFF and '1' is ON.

# **PWM**

## **OS.PWM.Init**

Initialize pulse width modulation.

```
OS.PWM.Init(channel, frequency, multiplier, prescale)
```

## OS.PWM.DutyCycle

Set duty-cycle of the pulse width modulator.

```
OS.PWM.DutyCycle(channel, duty cycle)
```

### Example:

```
OS.PWM.Init(2, 1000, 1, 1)
OS.PWM.DutyCycle(2, 50) '0.5 millisecond square wave
OS.PWM.Start(2)
```

## OS.PWM.Start

Start the PWM at specified channel.

```
OS.PWM.Start(channel)
```

## OS.PWM.Stop

Stop the PWM at specified channel.

```
OS.PWM.Stop(channel)
```

### Example:

```
Public Sub Main()
  Delay(800)

  Debug.Print "PWM TEST"

  Do
    Delay(80)
    OS.PWM.Start(1)

    Delay(80)
    OS.PWM.Stop(1)
    Debug.Print "PWM TEST ENDLESS... "; CStr(I)
    I = I + 1
  Loop

End Sub
```

## SD

### OS.SD.Read

Read data from specified cluster and section in the FICE SD, into the OS buffer.  
Buffer size = 512 bytes.

```
OS.SD.Read(Cluster, Sector)
```

**Note:** Data from the SD will overwrite any previous data that is in the buffer.

### OS.SD.Write

Write data into specified cluster and sector of the FICE SD card.

```
OS.SD.Write(Cluster, Sector)
```

## WAVE

### OS.Wave.Play

Start playing wave file from specified cluster and sectors of the FICE SD card.

```
OS.Wave.Play(Cluster, Start Sector, End Sector)
```

**Note:** This information can be obtained when you use the SD tool to store image and sound files into the FICE SD card. Use the 'Copy to Clipboard' function to paste file information into excel or some other text editor for easy reference.

### OS.Wave.Stop

Stop playing the wave file.

```
OS.Wave.Stop()
```

**Note:** Its good programming practice to stop the current wave file before starting to play another one.

### OS.Wave.Volume

Set the volume to play the wave file.  
Range:

```
OS.Wave.Volume(volume)
```

**Example:**

```
'This program plays a wave file and does something else (like  
'displaying images on the GLCD), at the same time, until the sound file  
'has finished playing, (where OS.Wave.State is 0) and continues with  
'the next sound file.
```

```
OS.Wave.Play(10, 410, 438)
```

```
Do
```

```
    'Actions to execute while audio is playing
```

```
Loop UNTIL OS.Wave.State() = 0
```

```
OS.Wave.Play(10, 318, 341)
```

```
Do
```

```
    'Actions to execute while audio is playing
```

```
Loop UNTIL OS.Wave.State() = 0
```

## Internal EEPROM

### EEDATA.READ

Read data from the internal EEPROM at the stated address.

```
EEDATA.READ(Address)
```

### EEDATA.WRITE

Write data to the internal EEPROM at the stated address.

```
EEDATA.WRITE(Address, Data)
```

### OS\_BUFFER

Command for using calling/using the OS.BUFFER as an active storage space or just like an array. Buffer size = 512 bytes.

```
OS_BUFFER
```

#### Example:

```
'This program displays the use of the OS_BUFFER in conjunction with  
'MEM_COPY and its capabilities. It also shows how you can retrieve the  
'data stored in the array.
```

```
Dim I, J As Integer  
Dim ARR1(20) As Integer  
Dim ARR2(20) As Integer  
  
'=====
```

```
Public Sub Main()  
  
For I = 0 To 10  
    OS_Buffer(I) = I * 1000 + 1  
    ARR1(I) = I  
    ARR1(I) = I*2  
Next  
  
DISP()  
  
Mem_Copy(OS_Buffer, ARR1, 0, 5)  
DISP()  
Mem_Copy(ARR2, OS_Buffer, 0, 5)  
DISP()  
Mem_Copy(ARR1, ARR2, 0, 5)  
DISP()  
  
Do  
    Delay(1000)  
    Debug.Print "OS_BUFFER ENDLESS"  
Loop  
  
End Sub
```

```

'continued...
'=====
Public Sub DISP()

Debug.Print "OS_BUFFER : ";
For I = 0 To 10
    J = OS_Buffer(I)
    Debug.Print CStr(J); " ";
Next
Debug.Print
Debug.Print "ARR1      : ";
For I = 0 To 10
    Debug.Print CStr(ARR1(I)); " ";
Next
Debug.Print
Debug.Print "ARR2      : ";
For I = 0 To 10
    Debug.Print CStr(ARR2(I)); " ";
Next
Debug.Print
Debug.Print

End Sub
'=====

```

READ and WRITE from the EEPROM just like how you would an array.

## MEM\_COPY

Copy data from one array to another array.

**MEM\_COPY**(Source, Destination, Start, Length)

### Example:

```

Dim ARRAY1, ARRAY2 As String

'Copy data from array to array.
MEM_COPY(ARRAY1, ARRAY2, Start, Length)
'Copy data from OS buffer into an array.
MEM_COPY(OS.BUFFER, ARRAY2, Start, Length)
'Copy data from an array into the OS buffer.
MEM_COPY(ARRAY1, OS.BUFFER, Start, Length)

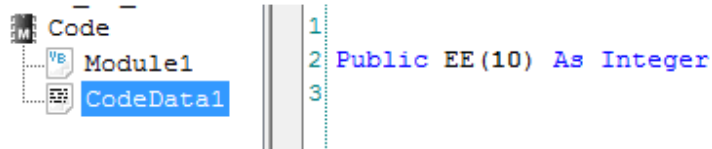
```

## External EEPROM

### Declaring External EEPROM Variable

```
Public [EEPROM Variable Name] ([]) As Integer
```

#### Under CodeData:



#### Example:

```
'Declaring External EEPROM Variable in CodeData1  
'This acts exactly like an array except writing to the EEPROM is not  
allowed.
```

```
Public EE(10) As Integer
```

### Reading External EEPROM Variable

External EEPROM variable can be read using similar method as you would an Array.

#### Example:

```
Dim I, DAT As Integer  
'=====
```

```
Public Sub Main()  
  
For I = 0 To 5  
  
    DAT = EE(I)  
  
    Debug.Print CStr(DAT); " ";  
Next  
  
End Sub
```

### Writing values to External EEPROM

Write data to the External EEPROM at the specified address.

```
OS.ExEEProm.Write(address, data)
```

## **RNDSeed**

Seed/Initiate the random generator. You should always seed the random generator with any value, before you use it.

```
OS.RNDSeed(SEED)
```

## **RND**

Sets a range for the random generator.

```
OS.RND(RANGE)
```

### **Example:**

```
OS.RND(5)
```

#### **Output :**

Output values ranging from 0-4 will be randomly generated.

## **OS.D2A.Ch1.Set**

Set digital to analog value at channel 1. (12 bit)

```
OS.D2A.Ch1.Set(value)
```

## **OS.D2A.Ch2.Set**

Set digital to analog value at channel 2. (12 bit)

```
OS.D2A.Ch2.Set(value)
```

## **ClrBit**

Clear bit value at specified position.  
Position = bit 0 to bit 15

```
ClrBit(value, position)
```

## **SetBit**

Set bit value at specified position.

```
SetBit(value, position)
```

# Commands

## **Abs**

Returns the absolute value of a variable.

```
Abs (value)
```

## **SetBit**

Set specified bit number of value or variable to 1.

```
SetBit (value, bit no.)
```

## **ClrBit**

Clear specified bit number of value or variable to 0.

```
ClrBit (value, bit no.)
```

## **GetBit**

Returns the bit value of the bit number specified in the value/variable.

```
GetBit (value, bit no.)
```

## **HI**

Returns the high byte of integer variable

```
HI (value)
```

## **LO**

Returns the low byte of integer variable

```
LO (value)
```

### **Example:**

```
Dim i(3) As Integer

Public Sub Main()

    Dim hiByte As byte
    Dim loByte As byte

    Dim a As byte
    Dim b As byte
    Dim byteCnt As Integer
```

```

'continued...
'=====

i(0) = 300
i(1) = 256
i(2) = 128

byteCnt = 3

Dim x As Integer
Dim aBit As byte
Dim bBit As byte

For x = 0 To byteCnt - 1

    loByte = LO(i(x))
    hiByte = HI(i(x))
    Debug.Print "hiByte = "; CStr(hiByte); " loByte = "; CStr(loByte)

    a = getbit(hiByte, 7)
    b = getBit(loByte, 7)
    Debug.Print "a = "; CStr(a); " b = "; CStr(b)

    a = clrbit(hiByte, 7)
    b = clrBit(loByte, 7)

    Debug.Print "final hByte = "; CStr(a); " final lByte = "; CStr(b)
    Debug.Print CStr(hiByte); " "; CStr(loByte)
    Debug.Print "===== "

Next

End Sub

```

**Output:**

```

hiByte = 1 loByte = 44
a = 0 b = 0
final hByte = 1 final lByte = 44
1 44
=====
hiByte = 1 loByte = 0
a = 0 b = 0
final hByte = 1 final lByte = 0
1 0
=====
hiByte = 0 loByte = 128
a = 0 b = 1
final hByte = 0 final lByte = 0
0 128
=====

```

## HIGH

Set logic 1 at the specified pin.

```
HIGH [ pin no. ]
```

### Example:

'Assuming an LED is attached to the specified pin, this code will turn the LED **OFF**.

```
HIGH 3
```

## LOW

Set logic 0 at the specified pin.

```
LOW [ pin no. ]
```

### Example:

'Assuming an LED is attached to the specified pin, this code will turn the LED **ON**.

```
LOW 3
```

**NOTE:** I/O pins are tied low to the ground on the hardware therefore explaining the inverse result as opposed to the logic statements.

## PutPin

Set state of particular pin. Where state '0' is off and '1' is on.

```
PutPin(Pin, State)
```

## GetPin

Returns state of particular pin.

```
GetPin [ pin no. ]
```

### Example:

```
Dim i As Integer  
  
Do  
    i = GetPin(1)  
    Debug.Print CSTR(i)  
    Delay(200)  
Loop
```

## PulseIn

Find the pulse width of a pulse from low to high to low transition at specified pin.

```
PulseIn [ pin no. ]
```

### Example:

```
'Assuming a dual pushbutton module is attached to the specified pin,  
'this code will turn read the pulse from the pushbutton if you press on  
'it fast enough.
```

```
Dim i As Integer
```

```
Do
```

```
    i = PulseIn(1)
```

```
    '0 and 32001 is the maximum range for the pulse width that can be  
    'read.
```

```
    If (i <> 0) or (i <> 32001) then
```

```
        Debug.Print CSTR(i)
```

```
    End If
```

```
Loop
```

**NOTE:** The Microprocessor will only wait for 20ms for the pulse to come in at the pin. Any pulse readings must fall within the 20ms range, if no readings come in by 20ms, it will return a 0.

## PulseOut

Send a pulse from high to low transition at specified pin.

```
PulseOut (Pin, Duration)
```

## GETADC

Retrieves converted digital values from analog device connected at specified pin of the FI board.

```
GETADC [ pin no. ]
```

### Example:

```
'This program infinitely retrieves the values of a device attached to  
'pin 3 of the FI board and displays it in the debug window.
```

```
Dim i As Integer
```

```
Do
```

```
    i = GETADC(3)
```

```
    Debug.Print CSTR(i)
```

```
Loop
```

## **DELAY**

Delay by default is equivalent to DELAYMS where time is delayed in milliseconds.

```
Delay(time in milliseconds)
```

## **DELAYMS**

Sets delay time in milliseconds.

```
DelayMS(time in milliseconds)
```

## **DELAYUS**

Sets delay time in microseconds.

```
DELAYUS(time in microseconds)
```

### **Example:**

```
'This program infinitely blinks an LED attached to pin 3 of the FI  
'board at the rate of 1ms. (500us = 0.5ms) * 2 = 1ms
```

```
Do
```

```
    HIGH 3
```

```
    DELAYUS(500)
```

```
    LOW 3
```

```
    DELAYUS(500)
```

```
Loop
```