SMART BOX

Operating System Serial Protocols

ECONOMATICS

1 Version

Read the operating system version number.

Send:	Byte 0 =1
Returns:	Byte 0 = version number - low byte Byte 1 = version number - high byte
Comments:	Divide by 1000 to get the version number, eg. version 1.023 would be returned as 1023.

2 Reset

Reset Smart Box.

Send:	Byte 0 = 2 Byte 1 = 255
Returns:	Nothing
Comments:	Reset Smart Box from the host micro. This performs the same function as pressing the reset switch on the rear panel.

3 NameCode

To obtain the operating system call number where the name is known.

Send:	Byte 0 = 3 Byte 1 - n = ASCI characters of OS call name Byte n+1 = 13
Returns:	Byte 0 = Operating system call number.

4 CodeName

To obtain the name associated with an operating system call.

Send:	Byte 0 = 4 Byte 1 = OS call number
Returns:	String of characters (terminated by character13) specifying OS call name.

5 MultipleSetup

To set up the values that will be returned by MultipleRead.

Send :	Byte 0 = 5 Byte 1 = Analogue sensor 1 Byte 2 = Analogue sensor 2 Byte 3 = Analogue sensor 3 Byte 4 = Analogue sensor 4 Byte 5 = Digital Outputs Byte 6 = Motor Outputs Byte 7 = Digital sensors
Returns:	Nothing
Comments:	This sets up the readings which will be returned when the call MultipleRead is made. If a byte= 1 the corresponding port will be returned, 0 = value not returned.

6 MultipleRead

Returns multiple readings as defined using MultipleSetup.

Send:	Byte 0 = 6
Returns:	Bytes as defined by MultipleSetup
Comments:	This call returns readings from a number of ports as defined by MultipleSetup.

7 MultipleServer

Constantly returns multiple readings as defined using MultipleSetup.

Send:	Byte 0 = 7
Returns:	Bytes as defined by MultipleSetup
Comments:	This call is similar to call 6 but continues to return readings until Smart Box receives the byte 123.

9 Copyright

Returns the copyright string.

Send:	Byte 0 = 9
Returns:	String terminated by NUL
Comments:	This call returns the copyright message and OS details.

10 WriteMotors

Writes a byte to the motor drivers.

Send:	Byte 0 =10
Returns:	Nothing

11 ReadMotors

Read the state of the motor drivers.

Send:	Byte 0 =11
Returns:	Byte 0 = byte read from the motor drivers.

12 MotorForward

Switch a motor /motors on

Send:

Byte 0 =12
Byte 1 = byte determining which motors are on:
1 = motor 1
4 = motor 2
16 = motor 3
64 = motor 4

Returns: Nothing

13 MotorReverse

Switch a motor /motors with reverse polarity

Send:

Byte 0 =13
Byte 1 = byte determining which motors are on:
2 = motor 1
8 = motor 2
32 = motor 3
128 = motor 4

Returns: Nothing

14 MotorHalt

Switch a motor /motors off

Send:

Byte 0 =13
Byte 1 = byte determining which motors are off:
2 = motor 1
8 = motor 2
32 = motor 3
128 = motor 4

Returns: Nothing

14 MotorPower

Pulse the motor outputs to vary the speed

Send: Byte 0 =14
Other details to be confirmed

Returns: Nothing

20 WriteOutputs

Write an 8 bit value to the digital output port.

Send: Byte 0 = 20
Byte 1 = byte to be written to the port

Returns: Nothing.

21 OutputPower

Vary the power, by pulsing of individual output lines.

Send: Byte 0 = 21
Other details to be confirmed

Returns: Nothing.

28 SetBitHigh

Set individual output line / lines high

Send: Byte 0 = 28
Byte 1 = byte determining state of individual lines, a bit set in this byte will set the corresponding output line high

Returns: Nothing.

28 SetBitLow

Set individual output line / lines low

Send: Byte 0 = 29
Byte 1 = byte determining state of individual lines, a bit set in this byte will set the corresponding output line low

Returns: Nothing.

40 ReadADC

	Take a reading from a specific ADC channel.
Send:	Byte 0 = 40 Byte 1 = channel number
Returns:	For an 8 bit reading Byte 0 = reading from ADC For a 10 bit reading Byte 0 = Low byte Byte 1 = High byte
Comments:	The value returned will be at the resolution specified by OS call numbers 44, 45, 46.

41 ReadADC's

Read all the ADC channels.

Send: Byte 0 = 41

Returns: For 8 bit readings
Byte 0 reading from channel 1
Byte 1 reading from channel 2
Byte 2 reading from channel 3
Byte 3 reading from channel 4

For 12 or 16 bit readings
Byte 0 reading from channel 1 (low byte)
Byte 1 reading from channel 1 (high byte)
Byte 2 reading from channel 2 (low byte)
Byte 3 reading from channel 2 (high byte)
Byte 4 reading from channel 3 (low byte)
Byte 5 reading from channel 3 (high byte)
Byte 6 reading from channel 4 (low byte)
Byte 7 reading from channel 4 (high byte)

42 ForcedADCRead

Force the A to D convertor to make a conversion and return the result.

Send: Byte 0 = 42
Byte 1 = Channel number in the range 1 to 4

Returns: For 8 bit readings
Byte 0 = Reading from the ADC
For 10 bit readings
Byte 0 = Reading (low byte)
Byte 1 = Reading (High byte)

Comments: Refer to calls 44 - 46 for the resolution setting.

43 ReadSensor

Read currently connected sensors

Send: Byte 0 = 43

Returns:
Byte 0 = Sensor on channel A
Byte 1 = Sensor on channel B
Byte 2 = Sensor on channel C
Byte 3 = Sensor on channel D

Returns sensor as ID code in range 0 -16

44 HighResADC

Sets the A to D convertor to make 12 bit conversions. Subsequent readings will return 2 byte values.

Send: Byte 0 = 44

Returns: Nothing.

45 LowResADC

Sets the A to D convertor to make 8 bit conversions. Subsequent readings will return single byte values.

Send: Byte 0 = 45

Returns: Nothing

47 ReadResolution

Read the current ADC resolution setting.

Send: Byte 0 = 47

Returns: Byte 0 = resolution setting where:
0 = 8 bit
1 = 12 bit

50 DownLoadData

This call is used to download data *into* Smart Box's memory.

Send: Byte 0 = 50
Byte 1 = start address for write (least significant byte)
Byte 2 = start address for write (most significant byte)
Byte 3 = length of data(least significant byte)
Byte 4 = length of data(most significant byte)
Bytes 5 - n = data

Returns: Nothing.

51 DownLoadDataX

Download data into Smart Box using Xmodem (CRC) protocol.

Send: Byte 0 = 51
Byte 1 = start address for write (least significant byte)
Byte 2 = start address for write (most significant byte)

Returns: Nothing.

Comments: Xmodem and Xmodem CRC supported.

52 UpLoadData

Upload data *from* Smart Box.

Send: Byte 0 = 52

Byte 1 = start address for read (least significant bit)
Byte 2 = start address for read (most significant bit)
Byte 3 = length of data (least significant bit)
Byte 4 = length of data (most significant bit)

Returns: Bytes 0 - n = data.

53 UpLoadDataX

Upload data from Smart Box using Xmodem (CRC) protocol.

Send: Byte 0 = 51
Byte 1 = start address for write (least significant bit)
Byte 2 = start address for write (most significant bit)

Returns: Nothing.

Comments: Xmodem and Xmodem CRC supported.

54 ExecuteCode

Execute machine code held at a specific address.

Send: Byte 0 = 54
Byte 1 = execution address (low byte)
Byte 2 = execution address (high byte)
Byte 3 = contents of accumulator on entry to the code
Byte 4 = contents of X register on entry to the code
Byte 5 = contents of Y register on entry to the code

Returns: Nothing.

55 StoreByte

Store a byte into Smart Box's RAM.

Send: Byte 0 = 55
Byte 1 = memory address (low byte)
Byte 2 = memory address (high byte)
Byte 3 = byte to be stored

Returns: Nothing.

Comments: Attempts to write to the data direction register of the printer port will be ignored.

56 ReadByte

Read a byte from Smart Box's memory.

Send: Byte 0 = 56
Byte 1 = memory address (low byte)
Byte 2 = memory address (high byte)

Returns: Byte 0 = byte read from Smart Box's memory.

57 ReadRAMSize

Read the amount of RAM with which Smart Box is fitted.

Send: Byte 0 = 57

Returns: Byte 0 = Number of Bytes (least significant byte)
Byte 1 = Number of Bytes (most significant byte)

58 ReadModule

Read the names of any modules fitted.

Send: Byte 0 = 58

Returns: Byte0 - n = String terminated by NUL
String = part time NUL terminated
String = part version (4 byte - form x.xx)
byte = 255 parts list end

Comments: A module is intended for permanent installation to turn the Interface into a dedicated Interface or to install a permanent OS call extension. Only one module can be fitted (in a single ROM), because of this it may have many parts in the same ROM. See Chapter 6, Smart Box and machine code, for more details.

59 ExtendCall

Call the extended call vector.

Send:	Byte 0 = 59
Returns:	Byte 0 = extension value

Comments: This call provides the user with the possibility of adding extra calls to Smart Box. See Appendix B, machine code programming, for more details.

60 SetClock

Set the internal clock in Smart Box. This clock only runs while the power is maintained.

Send:	Byte 0 = 60 Byte 1 = 1/10 Seconds (0-9) Byte 2 = Seconds (0-59) Byte 3 = Minutes (0-59) Byte 4 = Hours (0-23)
Returns:	Nothing.

61 ReadClock

Read the internal clock. On power up or reset, the clock will be set to zero.

Send:	Byte 0 = 61
Returns:	Byte 0 = 1/10 Seconds Byte 1 = Seconds Byte 2 = Minutes Byte 3 = Hours Byte 5 = Days

62 ReadTopMem

Read the current value of TopMem.

Send:	Byte 0 = 62
Returns:	Byte 0 = value of TopMem (lsb) Byte 1 = value of TopMem (msb)

63 WriteTopMem

Write a new value for TopMem.

Send:	Byte 0 = 63 Byte 1 = new value of TopMem (lsb) Byte 2 = new value of TopMem (msb)
Returns:	Nothing

64 ReadLoMem

Read the current value of LoMem.

Send:	Byte 0 = 64
Returns:	Byte 0 = value of LoMem (lsb) Byte 1 = value of LoMem (msb)

65 WriteLoMem

Write a new value for LoMem.

Send:	Byte 0 = 65 Byte 1 = new value for LoMem (lsb) byte 2 = new value for LoMem (msb)
Returns:	Nothing

66 ReadHiMem

Read the current value of HiMem.

Send:	Byte 0 = 66
Returns:	Byte 0 = value of HiMem (lsb) Byte 1 = value of HiMem (msb)

67 WriteHiMem

Write a new value for HiMem.

Send:	Byte 0 = 67 Byte 1 = new value for HiMem (lsb) byte 2 = new value for HiMem (msb)
Returns:	Nothing

90 ReadInputs

Reads a byte from the digital sensor port

Send:	Byte 0 = 90
Returns:	Byte read from the digital sensor port

91 ReadBit

Reads a bit from the digital sensor port

Send:	Byte 0 = 90 Byte 1 = bit to read
Returns:	Bit read from the digital sensor port
Comments:	Reads an individual sensor from the digital sensor port