

SPI Port

This embedded C function can be used to add additional SPI ports to the Micromite.

It has the following features:

- It is always the master and can run at speeds up to 4MHz.
- The operating mode (mode 0 to 3) and the number of bits to send/receive can be specified.
- It can use any three I/O pins and these I/O pins can be changed from call to call so an unlimited number of SPI interfaces can be created.

Adding the Function to MMBasic

To add the SPIPort function to MMBasic you must insert the following code somewhere in your BASIC program (you can use copy and paste from this document). The exact spot is not important.

```
CFunction SPIPort
00000008
40024800 00442021 40024800 0044102B 1440FFFFD 00000000 03E00008 00000000
27BDFFB8 AFBF0044 AFBE0040 AFB7003C AFB60038 AFB50034 AFB40030 AFB3002C
AFB20028 AFB10024 AFB00020 00809021 00A08821 00C09821 8C840000 3C029D00
8C420088 00041880 00621021 8C430000 24020002 10620006 00E0B821 3C029D00
8C420010 24050002 0040F809 00003021 8E240000 3C029D00 8C420088 00041880
00621021 8C430000 24020008 10620005 3C029D00 8C420010 24050008 0040F809
00003021 8E640000 3C029D00 8C420088 00041880 00621021 8C430000 24020008
10620005 3C029D00 8C420010 24050008 0040F809 00003021 3C109D00 8E020024
8E440000 0040F809 00002821 AFA20010 8E020028 0040F809 8E440000 0040F021
8E020024 8E240000 0040F809 24050006 AFA20014 8E020024 8E240000 0040F809
24050005 AFA20018 8E020028 0040F809 8E240000 24150001 0055A804 8FA3005C
8C620000 10400003 24030003 1443000D 3C109D00 3C109D00 8E020024 8E640000
0040F809 24050006 0040B021 8E020024 8E640000 0040F809 24050005 1000000B
0040A021 8E020024 8E640000 0040F809 24050005 0040B021 8E020024 8E640000
0040F809 24050006 0040A021 3C029D00 8C420028 0040F809 8E640000 24120001
00529004 8FA20060 8C500000 12000044 8EF70000 2610FFFF 24020001 02028004
8FA20058 8C530000 16000040 00008821 10000027 8FA3005C 02171024 10400004
8FA20018 8FA30014 10000002 AC750000 AC550000 AE920000 00000000 00000000
00000000 00000000 00000000 1260000E 00118840 02602021 0411FF69 00000000
8FA30010 8C620000 AED20000 03C21006 30420001 02228825 0411FF61 02602021
10000008 00108042 8FA30010 8C620000 AED20000 03C21006 30420001 02228825
00108042 1600FFDD 02171024 8FA3005C 8C620000 10400003 24030002 14430003
02201021 AE920000 02201021 00001821 8FBF0044 8FBE0040 8FB7003C 8FB60038
8FB50034 8FB40030 8FB3002C 8FB20028 8FB10024 8FB00020 03E00008 27BD0048
8FA20058 8C530000 24100080 1000FFC2 00008821
End CFunction
```

Usage

```
rd = SPIPort( rx, tx, clk, data_to_send, speed, mode, bits )
```

Where:

- 'rd' is the value returned by the function and is the data received during the transaction.
- 'rx' is the pin number for the data input (MISO)
- 'tx' is the pin number for the data output (MOSI)
- 'clk' is the pin number for the clock generated by this function (CLK)

The following parameters are optional. If not required they can be left off the end of the list.

- 'data_to_send' is optional and is an integer representing the data to send over the output pin. If it is not specified the 'tx' pin will be held low.
- 'speed' is the speed of the clock (see below for an explanation). The default is 0.
- 'mode' is a single numeric digit representing the transmission mode (see below for an explanation). It is optional and if not specified the default is 0.
- 'bits' is the number of bits to send/receive. Range is 1 to 64. The default is 8.

The SPI function will return the data received during the transaction as an integer. Note that a single SPI transaction will send data while simultaneously receiving data from the slave.

Clock Speed

The 'speed' parameter is a number which can range from zero to some arbitrary number, the greater the number the slower the resultant SPI clock speed. The following table lists the various SPI clock speeds that result from typical CPU speeds and values used for the 'speed' parameter:

CPU speed	'speed' parameter and the resultant SPI clock speed		
	0	4	20
100 MHz	4.4 MHz	1.7 MHz	780 KHz
80 MHz	3.8 MHz	1.4 MHz	615 KHz
60 MHz	2.7 MHz	1.0 MHz	470 KHz
40 MHz	1.8 MHz	700 KHz	315 KHz
20 MHz	900 KHz	350 KHz	160 KHz

Transmission Format

The most significant bit is sent and received first. The format of the transmission can be specified by the 'mode' parameter as follows:

Mode	Description	CPOL	CPHA
0	Clock is active high, data is captured on the rising edge and output on the falling edge	0	0
1	Clock is active high, data is captured on the falling edge and output on the rising edge	0	1
2	Clock is active low, data is captured on the falling edge and output on the rising edge	1	0
3	Clock is active low, data is captured on the rising edge and output on the falling edge	1	1

For a more complete explanation see: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

I/O Pins

Before invoking this function the 'rx' pin must be configured as an input using the SETPIN command and the 'tx' and 'clk' pins must be configured as outputs (either normal or open collector) again using the SETPIN command. The clock pin should also be set to the correct polarity (using the PIN function) before the SETPIN command so that it starts as inactive.

The SPI enable signal is often used to select a slave and "prime" it for data transfer. This signal is not generated by this function and if required should be generated using the PIN function.

The SPIPort function does not "take control" of the I/O pins and the PIN command will continue to operate as normal on them. Also, because the I/O pins can be changed between function calls it is possible to communicate with many different SPI slaves on different I/O pins.

Example

The following example will send the command 80 (hex) and receive two bytes from the slave SPI device. Because the mode, speed and number of bits are not specified the defaults are used.

```
SETPIN 21, 2           ' set rx pin as a digital input
SETPIN 22, 8           ' set tx pin as an output
PIN(23) = 0 : SETPIN 23, 8 ' set clk pin low then set it as an output
PIN(24) = 1 : SETPIN 24, 8 ' pin 24 will be used as the enable signal

PIN(11) = 0           ' assert the enable line (active low)
junk = SPIPort(21, 22, 23, &H80) ' send the command and ignore the return
byte1 = SPIPort (21, 22, 23)    ' get the first byte from the slave
byte2 = SPIPort (21, 22, 23)    ' get the second byte from the slave
PIN(24) = 1           ' deselect the slave
```