USB-ISS Multifunction USB Communications Module

I2C Mode - Technical Specification

The USB-ISS always operates as an I2C bus master.
I2C mode has a number of commands for accessing I2C devices with 0, 1 or 2 internal address bytes, and for building your own custom I2C sequences. These are:

Command	Value	Description
I2C_SGL	0x53	Read/Write single byte for non-registered devices, such as the Philips PCF8574 I/O chip.
I2C_AD0	0x54	Read/Write multiple bytes for devices without internal address or where address does not require resetting.
I2C_AD1	0x55	Read/Write 1 byte addressed devices (the majority of devices will use this one)
I2C_AD2	0x56	Read/Write 2 byte addressed devices, eeproms from 32kbit (4kx8) and up.
I2C_DIRECT	0x57	Used to build your own custom I2C sequences.
I2C_TEST	0x58	Used to check for the existence of an I2C device on the bus. (V5 or later firmware only)

The USB-ISS module takes care of all the I2C bus requirements such as start/restart/stop sequencing and handles the acknowledge cycles. You only need supply a string of bytes to tell the module what to do.

Writing a single byte to I2C devices without internally addressable registers
These include devices such as the Philips PCF8574 I/O expander. Following the I2C_SGL you send the devices I2C address and the data byte.

	Primary USB-ISS command	Device Address + R/W bit	The data byte
Byte Type	I2C_SGL	Addr+R/W	Data
Example	0x53	0x40	0x00
Meaning	Direct Read/Write command	PCF8574 I2C address	Set all bits low

This 3 byte sequence sets all bits of a PCF8574 I/O expander chip low. All 3 bytes should be sent to the USB-ISS in one sequence. A gap will result in the USB-ISS re-starting its internal command synchronization loop and ignoring the message. After all bytes have been received the USB-ISS performs the IC2 write operation out to the PCF8574 and sends a single byte back to the PC. This returned byte will be 0x00 (zero) if the write command failed and non-zero if the write succeeded. The PC should wait for this byte to be returned (timing out after 500mS) before proceeding with the next transaction.

Reading a single byte from I2C devices without internally addressable registers
This is similar to writing, except that you should add 1 to the device address to make it an odd number. To read from a PCF8574 at address 0x40, you would use 0x41 as the address. (When the address goes out on the I2C bus, its the 1 in the lowest bit position that indicates a read cycle is happening). Here is an example of reading the inputs on a PCF8574 I/O expander:

I2C_SGL	PCF8574 I2C address + Read bit
0x53	0x41

The USB-ISS module will perform the read operation on the I2C bus and send a single byte (the PCF8574 inputs) back to the PC. The PC should wait for the byte to be returned (timing out after 500mS) before proceeding with the next transaction.

Writing multiple bytes to devices that do not have an internal address register.
There are very few, if any, real devices that operate this way, however some people have programmed uControllers to work like this. Here is an example of sending four bytes to a device at I2C address 0x30.

I2C_AD0	Device I2C address + Write bit	Number of bytes to write	Data 1	Data 2	Data 3	Data 4
0x54	0x30	0x04	0x12	0x34	0x56	0x78

After all bytes have been received the USB-ISS performs the IC2 write operation on the I2C bus and sends a single byte back to the PC. This returned byte will be 0x00 (zero) if the write command failed and non-zero if the write succeeded. The PC should wait for this byte to be returned (timing out after 500mS) before proceeding with the next transaction.

Reading multiple bytes from I2C devices without setting a new address
This is used for devices that do not have an internal register address but returns multiple bytes. Examples of such devices are the Honeywell ASDX DO series pressure sensors. This command can also be used for devices that do have an internal address which it increments automatically between reads and doesn't need to be set each time, such as eeproms. In this case you would use command I2C_AD1 or I2C_AD2 for the first read, then I2C_AD0 for subsequent reads. Here is an example of reading the two byte pressure from the Honeywell sensor.

I2C_AD0	ASDX I2C address + Read bit	Number of bytes to read
0x54	0xF1	0x02

The USB-ISS will perform the read operation on the I2C bus and send two bytes back to the PC - high byte first in this example for the ASDX sensor. The PC should wait for both bytes to be returned (timing out after 500mS) before proceeding with the next transaction.

Writing to I2C devices with a 1 byte internal address register
This includes almost all I2C devices. Following the I2C_AD1 command you send the device I2C address, then the devices internal register address you want to write to and the number of bytes you're writing. The maximum number of data bytes should not exceed 60 so as not to overflow the USB-ISS's internal buffer.

	Primary USB-ISS command	Device Address + R/W bit	Device internal register	Number of data bytes	The data bytes
Byte Type	I2C_AD1	Addr+R/W	Reg	Byte Count	Data
Example	0x55	0xE0	0x00	0x01	0x51
Meaning	Primary USB-ISS command	SRF08 I2C address	SRF08 command Reg	One command byte follows	Start ranging in cm

This 5 byte sequence starts an SRF08 at address 0xE0 ranging. All 5 bytes should be sent to the USB-ISS in one sequence. A gap will result in the USB-ISS re-starting its internal command synchronization loop and ignoring the message. After all bytes have been received the USB-ISS performs the IC2 write operation out to the SRF08 and sends a single byte back to the PC. This returned byte will be 0x00 (zero) if the write command failed and non-zero if the write succeeded. The PC should wait for this byte to be returned (timing out after 500mS) before proceeding with the next transaction.
Here is another write example - this time an 8 byte sequence to initialize the MD22 motor driver:

I2C_AD1	MD22 Addr+R/W	Mode Reg	Data byte count	MD22 mode 1	Left Motor Stopped	Right Motor Stopped	Fast acceleration
0x55	0xB0	0x00	0x04	0x01	0x00	0x00	0x02

Again the USB-ISS will respond with non-zero if the write succeeded and zero if it failed. A failure means that no acknowledge was received from the I2C device.

Reading from I2C devices with a 1 byte internal address register
This is similar to writing, except that you should add 1 to the device address to make it an odd number. To read from an SRF08 at address 0xE0, you would use 0xE1 as the address. (When the address goes out on the I2C bus, its the 1 in the lowest bit position that indicates a read cycle is happening). The maximum number of data bytes requested should not exceed 60 so as not to overflow the USB-ISS's internal buffer. Here is an example of reading the two byte bearing from the CMPS03 compass module:

I2C_AD1	CPMS03 I2C address + Read bit	CMPS03 bearing register	Number of bytes to read
0x55	0xC1	0x02	0x02

The USB-ISS will perform the read operation on the I2C bus and send two bytes back to the PC - high byte first. The PC should wait for both bytes to be returned (timing out after 500mS) before proceeding with the next transaction.

Writing to I2C devices with a 2 byte internal address register
This is primarily for eeprom's from 24LC32 (4k x 8) to 24LC1024 (2 * 64k x 8). Following the I2C_AD2 you send the device I2C address, then the devices internal register address (2 bytes, high byte first for eeprom's) and then the number of bytes you're writing. The maximum number of data bytes should not exceed 59 so as not to overflow the USB-ISS's 64 byte internal buffer.

	Primary USB-ISS command	Device Address + R/W bit	High byte of internal Address	Low byte of internal Address	Number of data bytes	The data bytes
Byte Type	I2C_AD2	Addr+R/W	Address High	Address Low	Byte Count	Data
Example	0x56	0xA0	0x00	0x00	0x20	0xnn
Meaning	Primary USB-ISS command	24LC32 I2C address	Address 0x0000	Address 0x0000	One command byte follows	32 (0x20) data bytes

This 37 byte sequence writes the last 32 bytes to address 0x0000 in the eeprom. All 37 bytes should be sent to the USB-ISS in one sequence. A gap will result in the USB-ISS re-starting its internal command synchronization loop and ignoring the message. After all bytes have been received the USB-ISS performs the IC2 write operation out to the eeprom and sends a single byte back to the PC. This returned byte will be 0x00 (zero) if the write command failed and non-zero if the write succeeded. The PC should wait for this byte to be returned (timing out after 500mS) before proceeding with the next transaction.

Reading from I2C devices with a 2 byte internal address register
This is similar to writing, except that you should add 1 to the device address to make it an odd number. To read from an eeprom at address 0xA0, you would use 0xA1 as the address. (When the address goes out on the I2C bus, its the 1 in the lowest bit position that indicates a read cycle is happening). The maximum number of data bytes requested should not exceed 64 so as not to overflow the USB-ISS's internal buffer. Here is an example of reading 64 (0x40) bytes from internal address 0x0000 of an eeprom at I2C address 0xA0.

I2C_AD2	Device I2C address + Read bit	High byte of internal Address	Low byte of internal Address	Number of bytes to read
0x56	0xA1	0x00	0x00	0x40

The USB-ISS will perform the read operation on the I2C bus and send 64 bytes back to the PC. The PC should wait for all 64 bytes to be returned (timing out after 500mS) before proceeding with the next transaction.

I2C_DIRECT commands
The I2C_DIRECT commands are used to build custom I2C sequences.

You send the I2C_DIRECT command (0x57) followed by as many sub-commands as required. These sub-commands are:

I2C Start	0x01	send start sequence
I2C Restart	0x02	send restart sequence
I2C Stop	0x03	send stop sequence
I2C Nack	0x04	send NACK after next read
I2C Read	0x20 - 0x2F	reads 1-16 bytes
I2C Write	0x30 - 0x3F	writes next 1-16 bytes

The I2C_Read sub-commands can read up to 16 bytes. 0x20 will read 1 byte, 0x21 will read 2 bytes, 0x2F will read 16 bytes. All bytes read are buffered and sent after all sub-commands are processed.
The I2C_Write sub-commands can write up to 16 bytes. 0x30 will write 1 byte, 0x35 will write 6 bytes, 0x3F will write 16 bytes. The bytes to be written immediately follow the write sub-command. It is up to you to make sure you supply the correct number of bytes as specified in the sub-command.

Here is the sequence to write 0x55 to the PCF8574 I/O expander.

I2C_DIRECT	I2C_START	I2C_WRITE2	PCF8574 I2C write address	Data	I2C_STOP
0x57	0x01	0x31	0x40	0x55	0x03

The following will write 4 bytes to a 24LC512 EEPROM. This uses 2 internal address bytes.

Direct	Start	Write7	I2C Address	Address High	Address low	Data1	Data2	Data3	Data4	Stop
0x57	0x01	0x36	0xA0	0x00	0x00	0x11	0x22	0x33	0x44	0x03

And this will read 4 bytes from a 24LC512 EEPROM.

Direct	Start	Write3	I2C Address	Address High	Address low	Restart	Write1	I2C address+RD	Read3	Nack	Read1	Stop
0x57	0x01	0x32	0xA0	0x00	0x00	0x02	0x30	0xA1	0x22	0x04	0x20	0x03

Note that as part of the read sequence we first have to write the address that we want to read from. We read 3of the 4 bytes then set the NACK flag before reading the last byte. This is according to I2C specifications, however its not really necessary and the following will work just as well:

Direct	Start	Write3	I2C Address	Address High	Address low	Restart	Write1	I2C address+RD	Read4	Stop
0x57	0x01	0x32	0xA0	0x00	0x00	0x02	0x30	0xA1	0x23	0x03

The response to the I2C_DIRECT command is a variable number of bytes, but it will always be at least two. You should read these two bytes first as they are the status bytes. The fist byte is the ACK (0xFF) or NACK (0x00) status.
If you get an ACK the command was successful and the second byte contains the number of bytes to read. If you have not requested any reads then it will be zero, otherwise you should now read the number of bytes available.
If you get a NACK the command failed and the second byte contains the reason, as follows:

Error Type	Error Code	Comment
Device Error	0x01	No ACK from device
Buffer Overflow	0x02	You must limit the frame to < 60 bytes

Buffer Underflow	0x03	More write data was expected than sent
Unknown command	0x04	Probably your write count is wrong

Checking for the existence of an I2C device (V5 and later firmware only)

I2C_TEST	Device I2C Address
0x58	0xA0

This 2 byte command was added to V5 following customer requests. It checks for the ACK response to a devices address. A single byte is returned, zero if no device is detected or non-zero if the device was detected.