Communication

Low-level USB communication

The Infinity USB Smart is a pure HID device with 64byte report packets.

EP0	IN	Control/HID 64 Bytes/1ms - Status from host to device.
EP1	OUT	HID 64 Bytes/2ms - Commands / data from host to device.
EP1	IN	HID 64 Bytes/2ms - Commands /data from device to host.

Each 64B packet on EP1 (OUT+IN) has a header of 1 bytes specifying the total amount of bytes actually filled into the buffer.

The 64byte package will look like this: <Length><Command/Data>

Example:
To get the productname of the device, the following shows the buffers transmitted.

Host->Device : 0x01,0x02,{62bytes unused}
Device->Host: 0x10,'I', 'n', 'f', 'S', 'm', 'a', 'r', 't', 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 , 0x00, 0x00, {47bytes unused}

Furthermore each HID report contains the report ID as the first byte in its buffer. The report ID is always 0.

Please see below for a list of commands to use.

To get started, below is a sequence of commands to set the Infinity USB Smart to communicate with a card running 3.579Mhz, at 9600baud (Which is actually 8064 baud at 3.00Mhz)

Enable phoenix mode at 3Mhz, 8064baud
Send (hex): 49 43 3C FD17 01

Reset card (positive)
Send (hex): 52
Delay 100-200ms
Send (hex): 53

Read ATR
Send (hex): 56
Read (hex): <LL><PP><data>
LL = Length of data (ATR)
PP = Parity error (normally 0)

Write data
Send (hex): 55 <LL><data>
LL = Length of data

Read echo of written data
Send (hex): 56
Read (hex): <LL><PP><data>
LL = Length of data
PP = Parity error (normally 0)

1.0 General commands

Command	Bytes		FW	Description
	InfSmart	PC		
d'00' – h'00'	1	0	0.xx	*NOP* No operation
d'01' – h'01'	1	4	0.xx	*Get firmware version* Returns firmware version as ASCII in the format: "x.xx"
d'02' – h'02'	1	16	0.xx	*Get productname* Returns productname as ASCII in the format: "xxxxxxxxxxxxxxx"
d'03' – h'03'	1	1	0.xx	*Get status* Returns statusregister Smartcardstatus (CARD): b'xxxxxxx0' = no card inserted b'xxxxxxx1' = card inserted Programmingerror (PROGERROR): b'xxxx000x' = No error since last read of Get status b'xxxxxx1x' = Verifyerror(s) (AVR programming) The PROGERROR bit gets reset after reading the statusregister.
d'04' – h'04'	9	0	0.xx	*Set LED* Turns on, off or flashes the dual-color (R+G) LED. Only Red and Green led available with 8bit resolution *[cRRGGBBDF]* RR = PWM dutycycle Red (0x0000 = off, 0xFF00 = on) GG = PWM dutycycle Green (0x0000 = off, 0xFF00 = on) BB = PWM dutycycle Blue (unused) D = Blink duty cycle (0x00 = Mostly On, 0xFF = Mostly Off) F = PWM frequency (0x00 = slow, 0xFF = fast) USB not initialized (Slow blink, short red) {0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0A, 0x80} Idle (Fully green) {0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0xFF}
d'10' – h'0A'	2	0	0.xx	*Set LED Activity* Enables LED activity measured on RX/TX pin [cA] A = Activity (0x00 = Off, 0x01 = On) Led for phoenix mode with activity: {0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xA0, 0xF0}

4.0 Synchronous / I2C commands

Command	Bytes		FW	Description
	InfSmart	PC		
d'50' – h'32'	1	0	0.xx	*ClockOutIICStart* Clocks out I2C start condition
d'51' – h'33'	1	0	0.xx	*ClockOutIICStop* Clocks out I2C stop condition
d'52' – h'34'	2	0	0.xx	*ClockOutIICByte* *Clocks out I2C byte* *[cX]* *X = Byte to clockout*
d'53' – h'35'	1	1	0.xx	*ClockInIICByteAck* *Clocks in I2C byte with ACK*
d'54' – h'36'	1	1	0.xx	*ClockInIICByteNoAck* *Clocks in I2C byte without ACK*
d'55' – h'37'	1	0	0.xx	*ClockIIC* *Clocks one cycle on I2C clock*
d'56' – h'38'	2	0	0.xx	*SyncProcess* *Clocks CLK until IO changes state to X (0 or 1)* *[cX]* *X = Stop when IO changes to this state (0 or 1)*
d'57' – h'39'	2	0	0.xx	*Sync_WriteOBLSB* *Synchronously clocks out one byte LSB first*
d'58' – h'3A'	1	0	0.xx	*Sync_Begin* *Powers on and resets the card ready for clocking out ATR*
d'59' – h'3B'	1	0	0.xx	*Sync_End* *Powers down card*
d'60' – h'3C'	4	0	0.xx	*Sync_WriteCommand* *Writes command, address and data to the card* *[cCAD]* *C = Command* *A = Address* *D = Data*
d'61' – h'3D'	1	0	0.xx	*Sync_Clock* *Clocks one cycle on CLK*
d'62' – h'3E'	2	X	0.xx	*Sync_Read8* *Reads X number of 8bit data* *[cX]* *X = Number of bytes to read*
d'63' – h'3F'	3	X*2	0.xx	*Sync_ReadX* *Reads X number of words consisting of Ybit data each. Y should be 1-16bits of data. This returns X*2 bytes of data, as each word is 16bit.* *[cXY]* *X = Number of words to read* *Y = Number of bits to clock into each word*
d'64' – h'40'	2	0	0.xx	*Sync_WriteOBMSB* *Synchronously clocks out one byte MSB first* *[cX]* *X = Byte to clock out*

5.0 Phoenix/Smartmouse commands

Command	Bytes		FW	Description
	InfSmart	PC		
d'73' – h'49'	6	0	0.xx	*Enable / Change Phoenix-mode* Sets the hardware in Phoenix mode, with the specified parameters [cABCDE]: **A = SBCON0 (Set to 0xC3)** Bit7:SB0CLK: Baud Rate Generator Clock Source. 0: SYSCLK is used as Baud Rate Generator Clock Source. 1: USBCLK is used as Baud Rate Generator Clock Source. Bit6:SB0RUN: Baud Rate Generator Enable. 0: Baud Rate Generator is disabled. UART0 will not function. 1: Baud Rate Generator is enabled.Bits5–2:Reserved: Read = 0000b. Must write 0000b. Bits1–0:SB0PS[1:0]: Baud Rate Prescaler Select. 00: Prescaler = 12 01: Prescaler = 4 10: Prescaler = 48 11: Prescaler = 1R/ **B = SMOD0** *Bit7:MCE0: Multiprocessor Communication Enable.* 0: RI will be activated if stop bit(s) are '1'. 1: RI will be activated if stop bit(s) and extra bit are '1' (extra bit must be enabled using XBE0). *Bits6–5:S0PT[1:0]: Parity* 00: Odd 01: Even 10: Mark 11: Space *Bit4:PE0: Parity Enable.* 0: Hardware parity is disabled. 1: Hardware parity is enabled. *Bits3–2:S0DL[1:0]: Data Length.* 00: 5-bit data 01: 6-bit data 10: 7-bit data 11: 8-bit data *Bit1:XBE0: Extra Bit Enable* 0: Extra Bit Disabled. 1: Extra Bit Enabled. *Bit0:SBL0: Stop Bit Length* 0: Short 1: Long **C = SBRLH0 (8bit MSB)** **D = SBRLL0 (8bit LSB)** 16bit Baudrate generator value, calculated from this formula:

				SBRL = (65536 – (48000000/baudrate)*0.5) A typical card running 9600baud at 3.579Mhz should run at 8064baud, at 3.00Mhz (or 16129baud at 6.00Mhz) E = 0 = Sysclock = 24MHz (Card frequency = 6.00Mhz) E = 1 = Sysclock = 12MHz (Card frequency = 3.00Mhz) Selects the clock for the card Example: 49 C3 3C F45F 00 = 8064 8E1 (Actual 9600baud) 49 C3 3D F45F 00 = 8064 8E2 (Actual 9600baud)
d'74' – h'4A'	1	0	0.xx	*Disable Phoenix-mode*
d'76' – h'4C'	6	0	0.xx	*Set UART (Same as command 0x49, but only changes baudrate)*
d'82' – h'52'	1	0	0.xx	*Set RTS* *Sets CReset low (card in reset)*
d'83' – h'53'	1	0	0.xx	*Clear RTS* *Sets CReset high (card out of reset)*
d'84' – h'54'	3	0	0.xx	*Trap* *Toggles CReset, waits X*10 [cXx] microseconds (0-2550us) and sends one byte to the card [cxX].*
d'91' – h'5B'	3	0	0.xx	*TrapBreak* *Sets I/O pins low (break), toggles CReset, waits X*10 [cXx] milliseconds (0-2550ms) and sends one byte to the card [cxX].*
d'85' – h'55'	1+	1	0.xx	*TXByte* *Sends (0-255) bytes to the card. The number of bytes to send [cXxx...] are followed by the actual data.*
d'86' – h'56'	1	2+	0.xx	*RXByte* *Receives the current buffer of data (0-255 bytes) from the card. The number of bytes are returned as one byte, then following 1 byte that indicates the number of parityerrors (returns 0 if none), and then the actual data (0-255 bytes)*