The Antec Kuhler 920 uses the USBXPRESS CP2101 interface to send and receive data.
The programmer's guide is available at
( http://www.silabs.com/Support%20Documents/TechnicalDocs/an169.pdf )

Data is received by making DLL calls to SiUSBXp.dll

Most of the information is covered in the programmer's guide, but to make it simpler I will list the steps here. (Generic since I do not know which language you are using)

All calls to SIUSBXP.dll return a DWORD value indicating success or several errors.
The values are listed in Appendix D of the an169.pdf document.

1) The first step is to determine how many (if any) coolers are connected

   This is done by calling the SI_GetNumDevices() function

```
SI_STATUS SI_GetNumDevices(LPDWORD NumDevices)
```

   SI_STATUS is the return DWORD indicating success or failuire (This is identical for all functions)
   NumDevices is a return value which indicates how many devices are present

2) The device must be opened and a handle is assigned

```
SI_STATUS SI_Open (DWORD DeviceNum, HANDLE *Handle)
```

   DeviceNum is the number of the device which must be opened (Device 1 = 0, Device 2=1 etc.)
   Handle is a return value which gives the handle of the opened device

3) In order to poll the device a command must be written to the device

```
SI_STATUS SI_Write (HANDLE Handle, LPVOID Buffer, DWORD NumBytesToWrite,
                    DWORD *NumBytesWritten, OVERLAPPED* o = NULL)
```

   Handle is the handle value as obtained in 2).
   Buffer is a pointer to a buffer which will be written
   NumBytesToWrite is the amount of bytes which should be written (This should be 32)
   NumBytesWritten is the return to indicate actual bytes written
   OverLapped is used when very long writes are done. (Leave this blank – "")

   The value in the buffer should be 4 bytes. The controller expects 10 00 00 00 , however it appears that the driver reverses the data before sending it, thus the buffer should be 0x00000010. The rest of the 32 bytes can be 00

4) Once the controller receives the "write" it will immediately send back a 32 byte pointer. This must then be read.

```
SI_STATUS SI_Read (HANDLE Handle, LPVOID Buffer, DWORD NumBytesToRead,
                   DWORD *NumBytesReturned, OVERLAPPED* o = NULL)
```

   Buffer is a pointer to a 32 byte read buffer,
   NumBytesToRead is 32
   NumBytesReturned is a return value which indicates how many bytes the device sent
   Overlapper is as above, Blank ("")

5) The buffer contains 32 bytes which for clarity I will break into 8x 4bytes

```
AB CD EF 12|AB CD EF 12|AB 28 EF 12|AB 05 EF 12|AB CD EF 12|AB CD EF 12|AB CD EF 12|
AB CD EF 12
```

The only “groups” of interest in this case are 3 and 4

In Group 3, byte 2 you find the integer of the temperature

```
AB 28 EF 12
```

In this case 28 which is 40 (dec).

In Group 4, byte 2 you find the decimal of the temp, this goes from 1 – A (No 0)

```
AB 05 EF 12
```

In this case 05 which is 5. This obviously need to be divided by 10 = 0.5

Thus the temperature is 40.5

(For A you would have 10/10 = 1 thus 40+1 = 41)

By repeating steps 3 – 5 you continuously get temperature updates.