System Parameter Sample

October 6, 2006

SkyeTek, Inc.
11030 Circle Point Road, Suite 300
Westminster, CO 80020

720-565-0441 phone
720-565-8989 fax

Sales
sales@skyetek.com

Technical Support
techsupport@skyetek.com

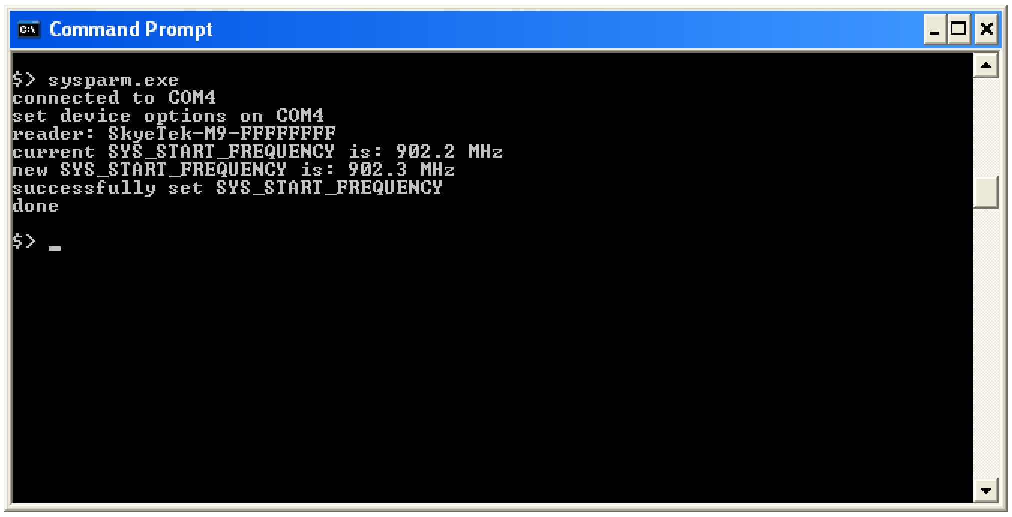
Table of Contents

1 Introduction

The SkyeTek System Parameter Example is a Visual C++ Windows application that demonstrates a host application getting and setting a system parameter on an SkyeModule M9. The example also shows how to create a device and reader manually instead of using discovery. Applications that know the device port a reader is attached to can use this method to get a handle to a reader. The example creates the device and reader and then gets, increments and sets the start frequency system parameter. This document describes the example application and the code beneath it.

2 Overview

The System Parameter Example displays output of its activities to standard out. It is hard-coded to connect to a reader on the COM4 port. It sets the serial communications parameters and then gets the handle to the reader. It gets the start frequency, increments it by 0.1 MHz and then sets it. The screenshot below shows the debug output.



```
$> sysparm.exe
connected to COM4
set device options on COM4
reader: SkyeTek-M9-FFFFFFFF
current SYS_START_FREQUENCY is: 902.2 MHz
new SYS_START_FREQUENCY is: 902.3 MHz
successfully set SYS_START_FREQUENCY
done

$> _
```

Figure 1: Loop Example Main Window

3 Source Code Walkthrough

3.1 The main Function

The main application entry point in the example code is the `main()` function. The `main()` function all the major functions functions:

- Creates a device handle by passing in “COM4” to the `SkyeTek_CreateDevice()` function.
- Opens the device handle using the `SkyeTek_OpenDevice()` function.
- Sets the baud rate to 38400, data bits to 8, parity to NONE and stop bits to ONE; these are the default serial communication parameters.
- Creates a reader handle for the device using `SkyeTek_CreateReader().`

- Gets the `SYS_START_FREQUECY` parameter using the `SkyeTek_GetSystemParameter()` function.
- Converts the data returned to floating point number.
- Increments the number by 0.1.
- Converts the floating point number back to data.
- Sets the `SYS_START_FREQUENCY` parameter using the `SkyeTek_SetSystemParameter()` function.
- Frees the handles using `SkyeTek_FreeData()`, `SkyeTek_FreeDevice()`, and `SkyeTek_FreeReader()`.

3.2 Device Addresses

Serial port device addresses are straight forward. They are strings that begin with "COM" and end with the port number, such as 1 or 2. Some examples are: "COM1", "COM2", … "COMN". USB port device addresses are more complex. To obtain a USB address, open the Device Manager for the computer and go to the HID-compliant Human Interface Devices. Open the device and view its Details page to get its Device Instance Id.

3.3 Working with Data

The `SkyeTek_GetSystemParameter()` and `SkyeTek_SetSystemParameter()` functions work with SKYETEK_DATA structures. These structures encapsulate the data and length information for data passed back and forth to the API. The SkyeTek_GetSystemParameter() allocates a data object to return. The calling function passes a pointer (set to NULL) to accept the newly allocated data. Once the calling function has finished manipulating the data, it should call SkyeTek_FreeData() to free it. The actual data is stored in the data variable in the structure. The following code shows how to set the 4-byte data for an integer.

```
lpData = SkyeTek_AllocateData(4);
lpData->data[0] = (i & 0xFF000000) >> 24;
lpData->data[1] = (i & 0x00FF0000) >> 16;
lpData->data[2] = (i & 0x0000FF00) >> 8;
lpData->data[3] = (i & 0x000000FF);
```

4 Building and Running the Example

The example code can be built using Visual C++ 6.0. The SkyeTek API is included in the **`stapi`** subfolder. The Visual C++ project includes the **`stapi`** directory in both the include file path and library path. The Visual C++ project also adds the **`stapi.lib`** in the link line. The **`stapi.lib`** is just a shell for the stapi.dll. To build the example, open Visual C++ 6.0 on the **`sysparm.dsw`** file and select "Rebuild All" from the Build menu. The post-build step copies the **`sysparm.exe`** to the **`bin`** directory.

To run the **`sysparm.exe`** program, the **`stapi.dll`** must be included in the same directory. The **`bin`** directory contains the **`stapi.dll`**. Also, the Visual C++ project copies the **`stapi.dll`** from the **`API`** directory into the resulting **`Debug`** or **`Release`** directory. Therefore, the System Parameter Example should run from the Visual C++ environment when debugger is run and/or the program is executed from within Visual C++.

5 Conclusion

The System Parameter Example demonstrates a simple program for reading and writing a system parameter using the SkyeTek API. It also shows how to create a device handle and a reader handle.

APPENDIX A. Code Listing

```
int main(int argc, char* argv[])
{
  SKYETEK_STATUS st;
  LPSKYETEK_DEVICE lpDevice = NULL;
  LPSKYETEK_READER lpReader = NULL;
  LPSKYETEK_DATA lpData = NULL;
  SKYETEK_SERIAL_SETTINGS settings;
  unsigned int i = 0;
  double f = 0.0;

  // set debugger -- uncomment this to see debug output
  // SkyeTek_SetDebugger(debug);

  // create device - change port to your COM port if different
  st = SkyeTek_CreateDevice("COM4", &lpDevice);
  if( st != SKYETEK_SUCCESS )
  {
    printf("error: could not create COM4 port device: %s\n", SkyeTek_GetStatusMessage(st));
    goto failure;
  }

  // open device
  st = SkyeTek_OpenDevice(lpDevice);
  if( st != SKYETEK_SUCCESS )
  {
    printf("error: could not open COM4 port: %s\n", SkyeTek_GetStatusMessage(st));
    goto failure;
  }
  printf("connected to COM4\n");

  // set our reader device options
  memset(&settings,0,sizeof(SKYETEK_SERIAL_SETTINGS));
  settings.baudRate = 38400;
  settings.dataBits = 8;
  settings.parity = NONE;
  settings.stopBits = ONE;
  st = SkyeTek_SetSerialOptions(lpDevice, &settings);
  if( st != SKYETEK_SUCCESS )
  {
    printf("error:    could    not    set    device    options    on    COM4    port:    %s\n",
SkyeTek_GetStatusMessage(st));
    goto failure;
  }
  printf("set device options on COM4\n");

  // create reader
  st = SkyeTek_CreateReader(lpDevice, &lpReader);
  if( st != SKYETEK_SUCCESS )
  {
    printf("error: could not find reader on COM4 port: %s\n", SkyeTek_GetStatusMessage(st));
    goto failure;
  }
  printf("reader: %s\n", lpReader->friendly);

  // check that it is a M9 reader
  if( strcmp(lpReader->model,"M9") != 0 )
  {
    printf("error: not an M9 reader: %s\n", SkyeTek_GetStatusMessage(st));
    goto failure;
  }

  // get system parameter
  st = SkyeTek_GetSystemParameter(lpReader,SYS_START_FREQUENCY,&lpData);
  if( st != SKYETEK_SUCCESS )
  {
    printf("error: could not get SYS_START_FREQUENCY: %s\n",
      SkyeTek_GetStatusMessage(st));
    goto failure;
  }

  // check value
  if( lpData == NULL || lpData->size == 0 )
  {
    printf("error: SYS_START_FREQUENCY is NULL or empty\n");
    goto failure;
  }

  // print frequency value
  i = (lpData->data[0] << 24) | (lpData->data[1] << 16) | (lpData->data[2] << 8) | lpData-
>data[3];
  f = (double)((double)i/1000000.0);
  printf("current SYS_START_FREQUENCY is: %.1f MHz\n", f);
  SkyeTek_FreeData(lpData);

  // calculate new value
  f += 0.1;
  i = (unsigned int)(f*1000000);
  lpData = SkyeTek_AllocateData(4);
  lpData->data[0] = (i & 0xFF000000) >> 24;
  lpData->data[1] = (i & 0x00FF0000) >> 16;
  lpData->data[2] = (i & 0x0000FF00) >> 8;
  lpData->data[3] = (i & 0x000000FF);
  printf("new SYS_START_FREQUENCY is: %.1f MHz\n", f);

  // set system parameter
  st = SkyeTek_SetSystemParameter(lpReader,SYS_START_FREQUENCY,lpData);
  if( st != SKYETEK_SUCCESS )
  {
    printf("error: could not set SYS_START_FREQUENCY: %s\n",
      SkyeTek_GetStatusMessage(st));
    goto failure;
  }
  printf("successfully set SYS_START_FREQUENCY\n");

  SkyeTek_FreeReader(lpReader); // do nothing if NULL
  SkyeTek_FreeDevice(lpDevice); // do nothing if NULL
  SkyeTek_FreeData(lpData); // do nothing if NULL
  printf("done\n");
  return 1;

failure:
  SkyeTek_FreeReader(lpReader); // do nothing if NULL
  SkyeTek_FreeDevice(lpDevice); // do nothing if NULL
  SkyeTek_FreeData(lpData); // do nothing if NULL
        return 0;

}
```