



Select Loop 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

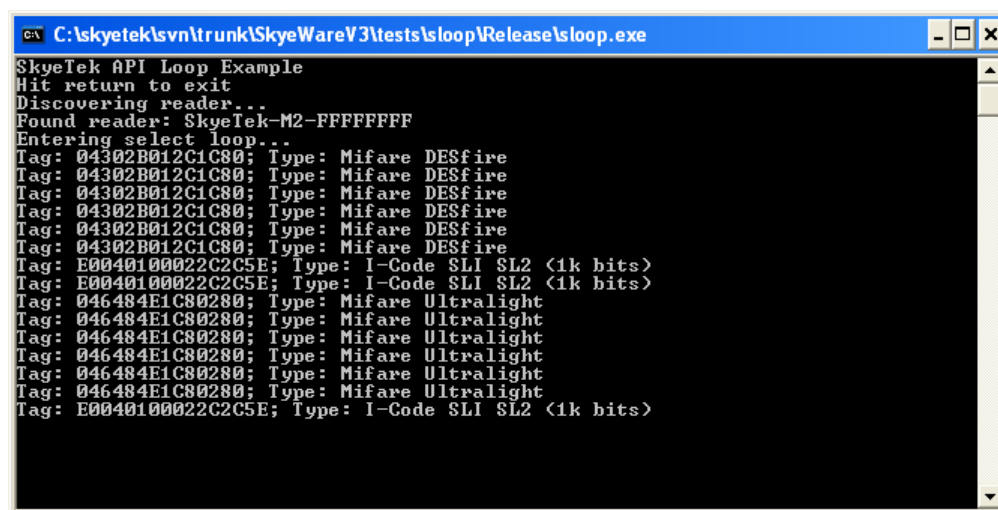
1	INTRODUCTION.....	3
2	OVERVIEW.....	3
3	SOURCE CODE WALKTHROUGH.....	3
3.1	The main Function.....	3
3.2	The ThreadProc Function	3
3.3	The SelectLoopCallback Function	4
3.4	Loop Control.....	4
4	BUILDING AND RUNNING THE EXAMPLE.....	4
5	CONCLUSION.....	5
APPENDIX A.	CODE LISTING	6

1 Introduction

The Skyetek Loop Example is a Visual C++ Windows application that demonstrates a host application performing a select loop for tags. It displays the tags identified during the select loop called by a separate thread. The code shows how to call the `Skyetek_SelectTags()` function, how to implement the callback and how to stop the select loop. This document describes the example application and the code beneath it.

2 Overview

The Loop Example displays the results of the select loop to standard out. It spawns a thread to communicate with the SkyeModule reader. As the thread discovers the reader, enters the select loop and discovers tags in its field, it reports the information to standard out. The screenshot below shows the Loop Example displaying results for three tags in the field.



```

C:\skyetek\svn\trunk\SkyeWareV3\tests\loop\Release\loop.exe
Skyetek API Loop Example
Hit return to exit
Discovering reader...
Found reader: Skyetek-M2-FFFFFFFF
Entering select loop...
Tag: 04302B012C1C80; Type: Mifare DESfire
Tag: 04302B012C1C80; Type: Mifare DESfire
Tag: 04302B012C1C80; Type: Mifare DESfire
Tag: 04302B012C1C80; Type: Mifare DESfire
Tag: 04302B012C1C80; Type: Mifare DESfire
Tag: 04302B012C1C80; Type: Mifare DESfire
Tag: E0040100022C2C5E; Type: I-Code SLI SL2 <1k bits>
Tag: E0040100022C2C5E; Type: I-Code SLI SL2 <1k bits>
Tag: 046484E1C80280; Type: Mifare Ultralight
Tag: 046484E1C80280; Type: Mifare Ultralight
Tag: 046484E1C80280; Type: Mifare Ultralight
Tag: 046484E1C80280; Type: Mifare Ultralight
Tag: 046484E1C80280; Type: Mifare Ultralight
Tag: 046484E1C80280; Type: Mifare Ultralight
Tag: E0040100022C2C5E; Type: I-Code SLI SL2 <1k bits>
  
```

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 performs two important functions:

- Creates the thread that communicates with the SkyeModule reader by calling the `CreateThread()` function.
- Enters a while loop waiting for the user to press the “Return” key.

3.2 The ThreadProc Function

The `ThreadProc()` function is the thread’s primary function. The handle to the function is passed in the `CreateThread()` function in `main()`. The `ThreadProc()` function performs three major tasks:

- It discovers the serial and USB devices connected to the host by calling the `SkyeTek_DiscoverDevices()` function.
- It then discovers the `SkyeModule` readers attached by calling the `SkyeTek_DiscoverReaders()` function, passing in the array of devices returned by `SkyeTek_DiscoverDevices()`.
- Lastly, it calls the `SkyeTek_SelectTags()` function to enter the select loop.

When the thread enters the select loop, it does not return until the loop is stopped. The thread passes in the handle to the select loop callback function when it calls `SkyeTek_SelectTags()`. The select loop will call that callback function every time it completes a read on the reader, either when the reader is reporting a tag in the field or when the read call timed out.

When the select loop ends, the `SkyeTek_SelectTags()` function returns and the thread cleans up the devices and readers it discovered and returns, ending the thread.

3.3 The SelectLoopCallback Function

A handle to the `SelectLoopCallback()` function is passed to the `SkyeTek_SelectTags()` function and is called by the select loop after every read call completes. The read completes either when a tag is in the field or when the read call times out. When the read completes, the callback is called. This gives the callback function an opportunity to terminate the select loop. An application can signal the select loop to stop by returning a value of zero from the callback function.

Looking at the `SelectLoopCallback()` function in the Loop Example, we see it first checks that stop flag to see whether or not it has been signaled to stop. If it is to stop, it returns zero. If it is to continue, it then checks the tag pointer, `lpTag`. The pointer will be `NULL` if the read call timed out and no tags have been detected in the field. Otherwise, it will point to a `SKYETEK_TAG` structure. The `SelectLoopCallback()` function prints out the tag ID (i.e. `lpTag->friendly`) and the tag type (using `SkyeTek_GetTagTypeNameFromType()` to get the string name of the type). Lastly, it calls `SkyeTek_FreeTag()` to free the tag.

3.4 Loop Control

The Loop Example controls the select loop using a stop flag. The stop flag causes the `SelectLoopCallback()` function to return zero the next time it is called by the select loop and, therefore, causes the select loop to stop and the thread to return from the `SkyeTek_SelectTags()` function.

Note that in the Loop Example, the flag is not really needed because the application exits its main loop in `main()` and calls `CloseHandle()` on the thread, thereby killing the thread and forcing the select loop to stop. The stop flag is included to show an example of how an application can cause the select loop to stop.

4 Building and Running the Example

The Example Loop 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 **sloop.dsw** file and select “Rebuild All” from the Build menu. The post-build step copies the **sloop.exe** to the **bin** directory.

To run the **sloop.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 Loop Example should run from the Visual C++ environment when debugger is run and/or the program is executed from within Visual C++.

The Loop Example connects to the first SkyeModule reader attached to the host. The reader may be attached either to the USB or Serial port. If no reader is attached, the program will sit in a loop until one is attached.

5 Conclusion

The Loop Example demonstrates a simple program for selecting tags in a field using the Skyetek API. It shows how a separate thread can be used in the select loop and how the callback function can be used to both report tags found in the field and stop the select loop.

APPENDIX A. Code Listing

```
#include "stdafx.h"
#include "SkyeTekAPI.h"
#include "SkyeTekProtocol.h"

// stop flag
bool isStop = false;

//
// FUNCTION: SelectLoopCallback(LPSKYETEK_TAG, void *)
//
// PURPOSE: Callback called by SkyeTek_SelectTags whenever
//          a tag is selected. This returns 1 to continue
//          and zero to stop.
//
//
unsigned char SelectLoopCallback(LPSKYETEK_TAG lpTag, void *user)
{
    if( !isStop )
    {
        if( lpTag != NULL )
        {
            printf("Tag: %s; Type: %s\n", lpTag->friendly,
                SkyeTek_GetTagNameFromType(lpTag->type));
            SkyeTek_FreeTag(lpTag);
        }
    }
    return( !isStop );
}

//
// FUNCTION: ThreadProc(LPVOID)
//
// PURPOSE: Main thread function. It sits in a loop until the
//          reader is discovered and then it calls the
//          SkyeTek_SelectTags function, which does not return
//          until the loop stops. To stop the loop, the
//          SelectLoopCallback needs to return zero.
//
//
DWORD WINAPI ThreadProc(LPVOID lpParameter)
{
    LPSKYETEK_DEVICE *devices = NULL;
    LPSKYETEK_READER *readers = NULL;
    SKYETEK_STATUS st;
    unsigned int numDevices;
    unsigned int numReaders;

    printf("Discovering reader...\n");
    while( !isStop )
    {
        numDevices = SkyeTek_DiscoverDevices(&devices);
        if( numDevices == 0 )
        {
            Sleep(100);
            continue;
        }
        if( isStop )
            return 1;

        numReaders = SkyeTek_DiscoverReaders(devices, numDevices, &readers);
        if( numReaders == 0 )
        {
            SkyeTek_FreeDevices(devices, numDevices);
        }
    }
}
```

```
        Sleep(100);
        continue;
    }
    break;
}

// set reader info
printf("Found reader: %s\n", readers[0]->friendly);

// the Skyetek_SelectTags function does not return until the loop is done
printf("Entering select loop...\n");
st = Skyetek_SelectTags(readers[0], AUTO_DETECT, SelectLoopCallback, 0, 1, NULL);
if( st != SKYETEK_SUCCESS )
    printf("Select loop failed\n");
printf("Select loop done\n");

// clean up readers
Skyetek_FreeReaders(readers, numReaders);
Skyetek_FreeDevices(devices, numDevices);
return 1;
}

int main(int argc, char* argv[])
{
    DWORD id1;
    HANDLE h;
    char line[128];

    printf("Skyetek API Loop Example\n");
    printf("Hit return to exit\n");

    if( (h=CreateThread(NULL, 0, ThreadProc, NULL, 0, &id1)) == NULL )
        return FALSE;

    gets(line);

    CloseHandle(h);

    printf("Done\n");
    isStop = true;
    return 0;
}
```