H-ITT SDK Software Development Kit Ver 2.0.3

The H-ITT system has been designed with an open architecture to enable independent software vendors, customers, and computer enthusiasts the ability to construct custom software solutions and extensions to the H-ITT system. H-ITT's freely available software development kit (SDK) enables this integration and is described in this document.
The SDK may be used with Windows, Mac, and Linux operating systems.

Published 2008
H-ITT, LLC
420 Shearer Blvd
Cocoa, Florida 32922

A Hyper-Interactive
Teaching Technology
Company

Contents

H-ITT SDK Software Development Kit Ver 2.0.2 1
1 Introduction and Overview 3
2 Base unit compatibility of the SDK (IR and RF): 3
3 Using the SDK 3
3.1 hitt_inspect 4
3.2 hitt_drid_inspect 5
3.3 hitt_action_packet 6
3.4 hitt_base_unit_mode 7

1 Introduction and Overview

The Software Development Kit (SDK) offers low-level access to the byte stream the H-ITT receivers use to communicate with the computer over the COM link. The SDK will work with serial and USB COM ports. The base units have a USB to COM adaptor built into them from FTDI and with the drivers create a virtual COM port. From a programming perspective the devices act just like a COM port. This enables construction of fully customized software around the H-ITT technology platform, or seamless integration of the H-ITT system into an existing application or information technology infrastructure. In addition, H-ITT extends a no royalty license for use of the software development kit to both academic and commercial users.
The H-ITT software development kit comes in one archive for all three operating systems: Windows, Mac-OSX, and Linux.
It contains functions for interfacing with the H-ITT remotes. These functions are written in C and can be called from any programming language capable of calling a dynamic link library for Windows platforms or a shared object for Linux and Mac OSX platforms.
Currently the H-ITT SDK comes with header files and function prototypes for these programming languages on the following operating systems:

- Windows: C/C++, Visual C/C++, Visual Basic 6 and .NET, via dynamic link library (.DLL).
- Mac OSX: C/C++, via shared object (.so)
- Linux: C/C++, via shared object (.so)

2 Base unit compatibility of the SDK (IR and RF):

The SDK works with both the infrared (IR) and radio frequency (RF) systems in the same exact way. No changes are needed to your source code or linking to the SDK to enable use of either the IR or RF system. The base units have an adjustable baud rate which can be set to one of the following by the user by manually adjusting dip switches on the base unit.

- 19.2 Kbps default for base units
- 38.4 Kbps.
- 57.6 Kbps.
- 115.2 Kbps

Your software should ask the user for the baud rate of base unit and default to the proper baud rate. In addition, you can allow the user to select one of the four baud rates.
The SDK does not work with the old low speed base units (model #2000) and/or remotes (ID# < 85000). H-ITT has not sold these low speed units since 2002 so it is unlikely that there will be any need to support them.

3 Using the SDK

This section will assume the reader has some familiarity with programming languages and writing code that calls functions in DLLs or shared objects. In addition, H-ITT recommends that the reader familiarize themselves with reading and writing to COM ports for their respective operating systems. To utilize the SDK The developer must provide software for reading and writing to the COM port on the computer. H-ITT's SDK then provides the needed functions to translate these bytes read into useful information such as the remote ID and key pressed or action codes. To utilize the SDK connect the base units to the COM port and open the COM port for reading and writing using the following settings:

- Baud rate = 19200, 38400, 57600 or 115200
- Parity = N

- Data Bits = 8
- Stop Bits = 1

After the COM port has been successfully opened there are two types of communication with the base units. The first is reading the bytes that it sends to the computer when responses from the remotes are detected and translating them into the remote ID number and key pressed ‘hitt_inspect’ function or response data ‘hitt_drid_inspect’ function. The second is sending individualized action codes to the remotes ‘hitt_action_packet’ function or sending commands to the base unit ‘hitt_base_unit_mode’ function.

3.1 hitt_inspect

When the receiver detects a valid signal it will send 10 bytes of information to the computer.
Pass these 10 bytes to the function called ‘hitt_inspect’ and check the return value.
The SDK include file defines the following self explanatory constants for the key codes.
HITT KEY A
HITT KEY B
HITT KEY C
HITT KEY D
HITT KEY E
HITT KEY F
HITT KEY G
HITT KEY H
HITT KEY I
HITT KEY J
HITT KEY FORWARD
HITT KEY REVERSE

HITT_ERROR
HITT_OK

Return value = HITT_OK: The function call was successful and id will contain the remote ID number and key code will contain the key code.

Return value = HITT_ERROR: There was an error and you should only discard the first byte and save the rest. When more bytes arrive on the COM port append 1 new byte onto the end of the saved ones and reprocess with hitt_inspect. This is an important step to implement properly. Only discard all 10 bytes if the function call was successful. If there was an error just discard the first byte. Discarding all 10 bytes upon an error will cause you programs buffers to get out of sequence with the incoming byte stream. If this happens your program will not properly detect the responses that are coming in.

The following is the function prototype in C:
int hitt_inspect(unsigned char *pBytes,unsigned int *pid, unsigned int *pkey_code);
Where:
pBytes is a pointer to at least 10 bytes.
pid is a pointer to the remote ID number if successful.
pBytes is a pointer to the key code if successful.

The following is the same function prototype but in Visual Basic 6:
Declare Function hitt_inspect Lib "H-ITTSDK.dll" (ByRef Bytes As Byte, ByRef id As Long, ByRef key_code As Long) As Long

Example source code:

The following shows an example on calling the function from C/C++:

```
#include <hittsdk.h>
/*...read in bytes from COM port (settings 19200 N 8 1) into a unsigned char buffer*/
unsigned char bytes[10];
unsigned int id;
unsigned int key_code;
/* consult OS documentation on reading COM port*/
read_com(bytes,10);
if(hitt_inspect(bytes,&id,&key_code) == HITT_OK){
printf("Success! ID=%d key_code=%d\n",id,key_code);
}else{
printf("Error.\n");
}
```

The following shows an example on calling the function from Visual Basic 6:

```
Dim DataBuffer(10) As Byte 'Data received from comm port.
Dim Key As Long 'Decoded remote button pressed.
Dim id As Long 'Decoded remote ID data is from.
Dim Ret As Long 'Status of function call
' read from COM port into DataBuffer
' consult VB documentation for reading from COM port

Ret = hitt_inspect(DataBuffer(0), id, Key)
'inspect Ret for success 'HITT_OK' or failure 'HITT_ERROR'
```

3.2 hitt_drid_inspect

Used to get remote ID and other data from LCD remotes.
You must first put base unit in DRID mode see hitt_base_unit_mode.
Then collect 32 bytes and pass to hitt_drid_inspect .
The SDK include file defines the following additional constants.
HITT_DRID_NORMAL
HITT_DRID_ROSTER_MODE
HITT_DRID_TESTING_MODE
HITT_DATA_BUFFER_LEN

If return is HITT_OK data is valid.
Return value = HITT_ERROR: There was an error and you should only discard the first byte and save the rest. When more bytes arrive on the COM port append 1 new byte onto the end of the saved ones and reprocess with hitt_inspect. This is an important step to implement properly. Only discard all 32 bytes if the function call was successful. If there was an error just discard the first byte. Discarding all 32 bytes upon an error will cause you programs buffers to get out of sequence with the incoming byte stream. If this happens your program will not properly detect the responses that are coming in.

The following is the function prototype in Visual Basic 6:

```
Declare Function hitt_drid_inspect Lib "H-ITTSDK.dll" (ByRef Bytes As Byte, id As Long, drid_type As Long, question_number As Long, assignment_number As Long, ByRef Data As Byte) As Long
```

Example source code:

The following shows an example on calling the function from Visual Basic 6:
Dim DataBuffer(32) As Byte 'Data received from comm port.
Dim id As Long 'Decoded remote ID data is from.
Dim drid_type As Long 'Decoded data type.
Dim question_number As Long 'Decoded question number if drid_type =
HITT_DRID_TESTING_MODE.
Dim assignment_number As Long 'Decoded assignment number if drid_type =
HITT_DRID_TESTING_MODE.
Dim Data(HITT_DATA_BUFFER_LEN) As Byte 'Decoded array of responce characters.
Dim Ret As Long 'Status of function call
' read from COM port into DataBuffer
' consult VB documentation for reading from COM port

Ret = hitt_drid_inspect(DataBuffer(0), id, drid_type, question_number, assignment_number, Data(0))
'inspect Ret for success or failure.

If ret is HITT_OK data is valid. If drid_type = HITT_DRID_TESTING_MODE then question_number contains the valid question number and assignment_number contains the valid assignment number. Otherwise question_number and assignment_number are ignored.

3.3 hitt_action_packet

Used for forming packets to send to the remotes.
H-ITT remotes can be sent commands to make their lights perform different actions for multiple choice remotes and display various text in the multi-digit remotes display.
The following is a list of action code constants:

H-ITT Constant	Multiple choice remotes	Multi-digit remotes
HITT AC RED = 0	Turns on remotes red LED	GRADE A
HITT AC GREEN = 1	Turns on remotes green LED	GRADE B
HITT AC YELLOW = 2	Turns on remotes red and green LEDs	GRADE C
HITT AC BLINK RED = 3	Blinks remotes red LED	GRADE D
HITT AC BLINK GREEN = 4	Blinks remotes green LED	GRADE F
HITT AC BLINK YELLOW = 5	Blinks remotes red and green LEDs	CORRECT
HITT AC TOGGLE RED GREEN = 6	Alternates remotes red and green LEDs	INCORRECT
HITT AC FAST BLINK RED = 7	Rapidly blinks remotes red LED	ACTION CODE 7
HITT AC FAST BLINK GREEN = 8	Rapidly blinks remotes green LED	ACTION CODE 8
HITT AC FAST BLINK YELLOW = 9	Rapidly blinks remotes red and green LEDs	ACTION CODE 9
HITT AC FAST TOGGLE RED GREEN = 10	Rapidly alternates remotes red and green LEDs	ACTION CODE 10
HITT AC OFF = 11	Turns off remotes LEDs	Turns off remotes LEDs

Call 'hitt_action_packet' by passing a pointer to an allocated chunk of memory of 5 bytes and the remote ID number of the remote and the action code desired and the function will fill form the packet. Then take this packet and send it to the receiver that will send along to the remotes.
The remote user must press the receive key on the remote (down arrow) for it to receive the packet and perform the action. When the receive key is pressed the green light stays on for 5 seconds and during this five seconds the remote is ready to receive an action packet.

The C prototype for the function call:
‘int hitt_action_packet(unsigned char *pBytes,unsigned int id, unsigned int action_code);’
The following is the function prototype in Visual Basic 6:
Declare Function hitt_action_packet Lib "H-ITTSDK.dll" (ByRef Bytes As Byte, ByVal id As Long, ByVal action_code As Long) As Long

Example source code:

The following shows an example on calling the function from Visual Basic 6:
Dim Ret As Long 'Status of function call
Dim Pkt(5) As Byte 'Encoded array of action packet bytes to be sent to comm port.
Dim id As Long 'Remote ID action packet is for.
id = 700001

Ret = hitt_action_packet(Pkt(0), id, HITT_AC_TOGGLE_RED_GREEN)
‘If return is HITT_OK data is valid. Send Pkt bytes to comm port.

3.4 hitt_base_unit_mode

Used for forming packets to command the RX-4100 and newer base units.
The RX-4100 and newer base units can be sent commands to allow them to receive the extended ‘DRID’ data packets.
The following is a list of base unit mode constants:
HITT_BASE_UNIT_NORMAL
HITT_BASE_UNIT_DRID

The following is the function prototype in Visual Basic 6:
Declare Function hitt_base_unit_mode Lib "H-ITTSDK.dll" (ByRef Bytes As Byte, ByVal mode As Long) As Long

Example source code:

The following shows an example on calling the function from Visual Basic 6:
Dim Ret As Long 'Status of function call
Dim Pkt(5) As Byte 'Encoded array of command packet bytes to be sent to comm port.

Ret = hitt_base_unit_mode(Pkt(0), HITT_BASE_UNIT_NORMAL) 'Set base unit for standard packets.
‘If return is HITT_OK data is valid. Send Pkt bytes to comm port.