Megawin

USB EasyPOD

User Manual

Index

1. Introduction....3
2. Advantages....3
3. Use Megawin USB EasyPOD....4
  3.1. System Handle Block....4
  3.2. Hardware Installation....4
  3.3. Software Developing Resource....5
  3.4. Including Megawin USB EasyPOD DLL....5
  3.5. Descriptions for Host DLL API....6
  3.6. Firmware Developing Resource....7
  3.7. Firmware Library Install....7
  3.8. Firmware Control Flow....9
  3.9. Used MCU resource....10
  3.10. Using Device Firmware Upgrade....11
4. Example Application....12
5. Obtaining The Parts....13
6. Revision History....13

1. Introduction

Nowadays, USB is accepted as the new standard for connecting computer peripheral devices, and most of these devices don't even require installation CD's or driver file. Simply just plug it into the USB port of the PC, and the system wizard will find and install the driver for the USB device. But unfortunately, it's not easy to develop an USB firmware to fit the USB classes, such as: HID, mass storage devices, and etc…

To solve this awkwardness, Megawin has provided an effortless USB library solution named "EasyPOD". This document will show how to apply "EasyPOD" to communicate with PC, without requiring the user to have sufficient knowledge on USB.

In addition, Megawin not only just provides the simplest device, but it also integrates some true and powerful peripherals into the same chip, such as: SPI, TWSI, and General IO. It helps user easily to control any devices on the system.

2. Advantages

2.1. USB is on every computer.

2.2. Extra flexibility on the USB: data buffering, no data lost, and etc...

2.3. USB provides the power source for the application.

2.4. Window Build-In driver on Win2K, WinXP, Vista32, and Vista64.

3. Use Megawin USB EasyPOD

3.1. System Handle Block

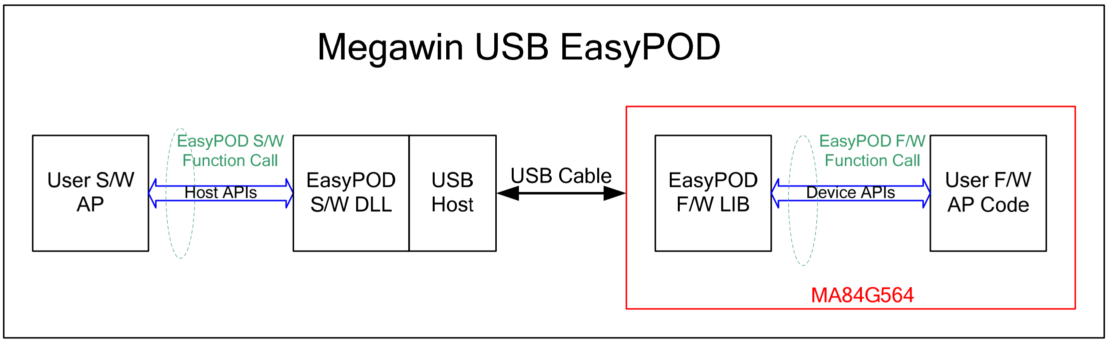
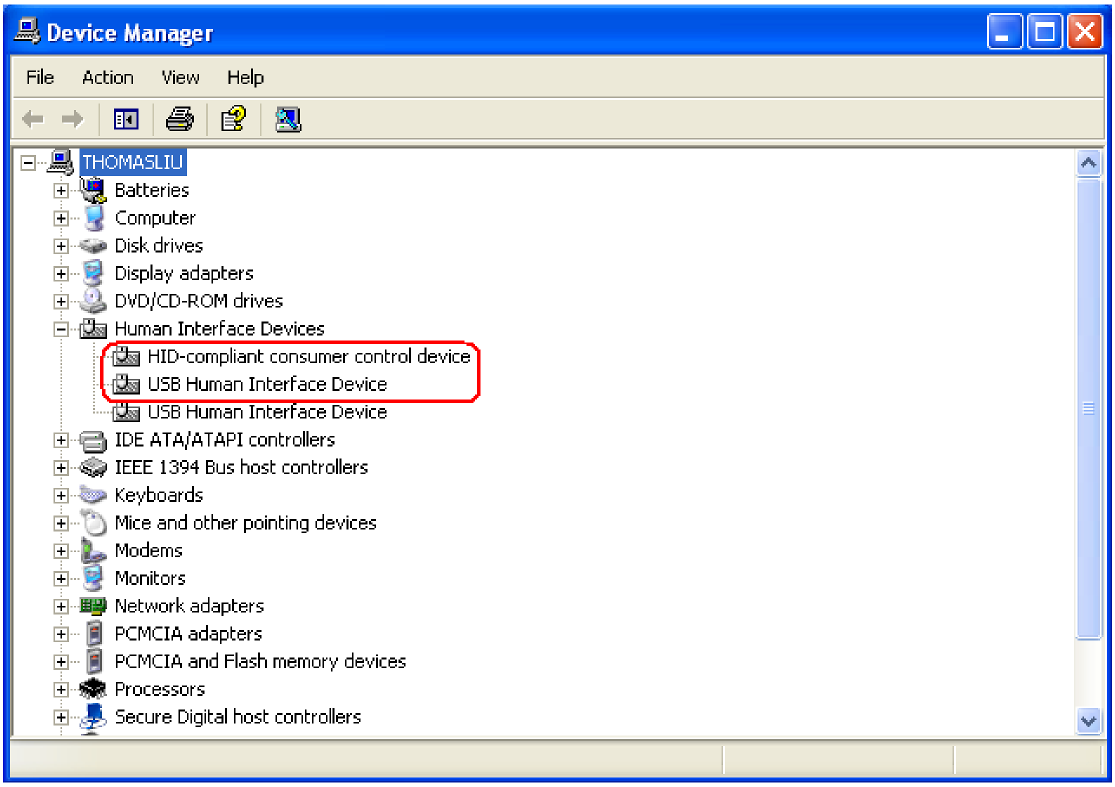


Fig-1

3.2. Hardware Installation

Plug the “**MA84G564 EV Stick**” (Please reference the “ ~Megawin Easy USB\MA84G564\Hardware\Evaluation Stick\MA84G564 Evaluation Stick for more details to confirm the EasyPOD firmware sample code stored in MA84G564 Evaluation Stick) into a PC’s USB port, and the Device Wizard will install driver automatically. After the driver is successfully installed, user will notice the following page in the “***System\Hardware\Device Manager***”, and see a new USB Human Interface Device added to the list of Human Interface Devices. (Fig-2)



- THOMASLIU
  - Batteries
  - Computer
  - Disk drives
  - Display adapters
  - DVD/CD-ROM drives
  - Human Interface Devices
    - HID-compliant consumer control device
    - USB Human Interface Device
    - USB Human Interface Device
  - IDE ATA/ATAPI controllers
  - IEEE 1394 Bus host controllers
  - Keyboards
  - Mice and other pointing devices
  - Modems
  - Monitors
  - Network adapters
  - PCMCIA adapters
  - PCMCIA and Flash memory devices
  - Processors
  - Secure Digital host controllers

Fig-2

3.3. Software Developing Resource

3.3.1. EasyPOD.DLL

3.3.2. EasyPOD.LIB

3.3.3. EasyPODDLL.H

3.4. Including Megawin USB EasyPOD DLL

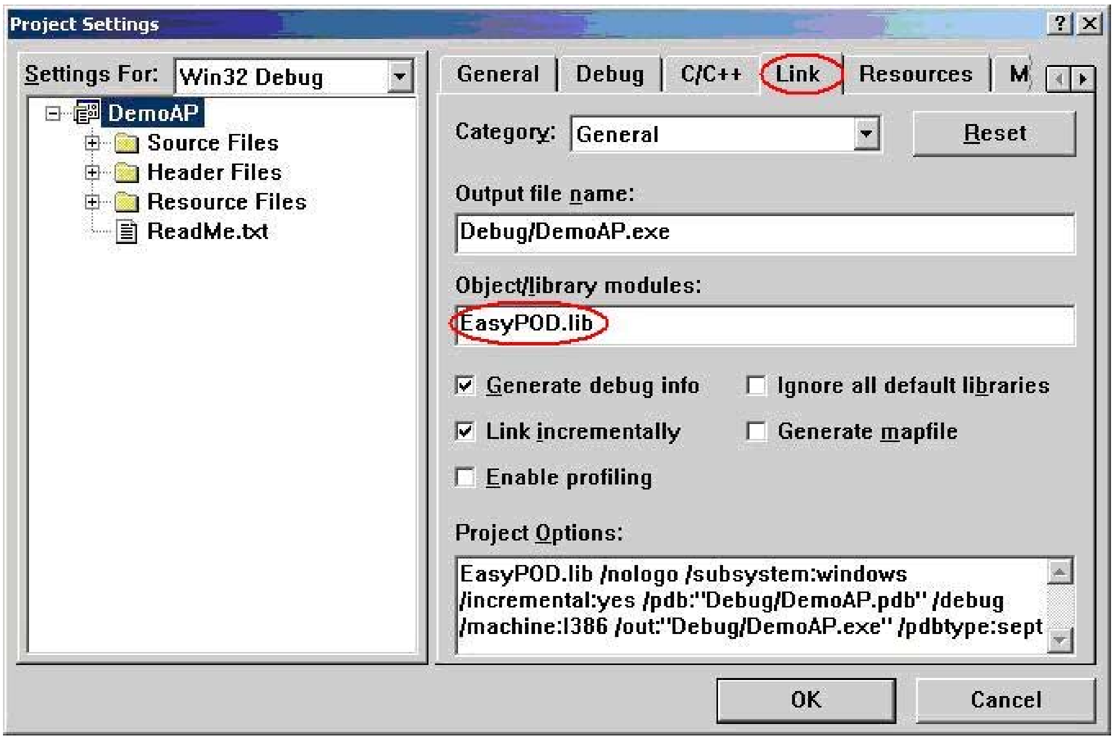
There is an example for Visual C++ 6.0 as following:

3.4.1. Copy previous three files to the directory of user developing project

3.4.2. Add “Head File” in user program

#include “EasyPODDLL.H”

3.4.3. Add “Export Library” (as Fig-3)



Settings For: Win32 Debug

- DemoAP
  - Source Files
  - Header Files
  - Resource Files
  - ReadMe.txt

Category: General

Output file name: Debug/DemoAP.exe

Object/library modules: EasyPOD.lib

☑ Generate debug info ☐ Ignore all default libraries

☑ Link incrementally ☐ Generate mapfile

☐ Enable profiling

Project Options:

```
EasyPOD.lib /nologo /subsystem:windows
/incremental:yes /pdb:"Debug/DemoAP.pdb" /debug
/machine:I386 /out:"Debug/DemoAP.exe" /pdbtype:sept
```

Fig-3

3.5. Descriptions for Host DLL API

3.5.1. ConnectPOD:

Function: **DWORD ConnectPOD(MW_EasyPOD * pEasyPOD, DWORD Index);**

Return: 0(ERROR_SUCCESS), Device connection **success**.

else, Device connection **fail**。Error message is defined in EasyPODDLL.H。

Parameter: pEasyPOD, a pointer of struct MW_EasyPOD.

Index, device index of EasyPOD, Index start at 1.

3.5.2. DisconnectPOD:

Function: **DWORD DisconnectPOD(MW_EasyPOD * pEasyPOD);**

Return: 0(ERROR_SUCCESS)，Device disconnection **success**。

else，Device disconnection **fail**。Error message is defined in EasyPODDLL.H。

Parameter: pEasyPOD, a pointer of struct MW_EasyPOD.

3.5.3. WriteData:

Function: **WriteData(MW_EasyPOD * pEasyPOD, LPBYTE lpBuffer , DWORD nNumberOfBytesToWrite , LPDWORD lpNumberOfBytesWritten);**

Return: 0(ERROR_SUCCESS)，Writes data to device **success**。

else，Writes data to device **fail**。Error message is defined in EasyPODDLL.H。

Parameter: pEasyPOD, a pointer of struct MW_EasyPOD。

lpBuffer, Pointer to the buffer containing the data to write to the device

nNumberOfBytesToWrite，Number of bytes to write to the device。

lpNumberOfBytesWritten，Number of bytes written。

3.5.4. ReadData:

Function: **ReadData(MW_EasyPOD * pEasyPOD, LPBYTE lpBuffer , DWORD nNumberOfBytesToRead , LPDWORD lpNumberOfBytesRead);**

Return: 0(ERROR_SUCCESS)，Reads data from device **success**。

else，Reads data from device **fail**。Error message is defined in EasyPODDLL.H。

Parameter: pEasyPOD, a pointer of struct MW_EasyPOD。

lpBuffer，Pointer to the buffer that receives the data read from the device。

nNumberOfBytesToRead，Number of bytes to be read from the device。

lpNumberOfBytesWritten ，number of bytes read。

3.5.5. ClearPODBuffer:

Function:**ClearPODBuffer(MW_EasyPOD * pEasyPOD);**

Return: 0(ERROR_SUCCESS)，Clears the buffer allocated for the device。

Else,Clears the buffer failed。Error messge is defined in EasyPODDLL.H。

Parameter: pEasyPOD, a pointer to struct MW_EasyPOD。

Software example code

```
BYTE Buffer[2] = {0xA1,0x02} ;
BYTE buf[32];
DWORD dwResult, dwSize, dwRtSize;
MW_EasyPOD dd;
dd.VID = 0x0E6A;
dd.PID = 0x0317;
dwSize = 2;
dwResult = ConnectPOD(&dd, 1);
if (dwResult == ERROR_SUCCESS)
{
  dd. ReadTimeOut = 50;     // Must be set,before read from device
  dd.WriteTimeOut = 100;    // Must be set before write to device
  ClearPODBuffer(&dd);
  WriteData(&dd, Buffer,dwSize,&dwRtSize);;
  ReadData(&dd, buf, dwSize, &dwRtSize) ;
  DisconnectPOD();
}
```

3.6. Firmware Developing Resource

3.6.1. **EasyPOD.LIB** (Firmware Library File)

3.6.2. **EasyPOD.H** (Firmware Header File)

3.6.3. **DFU.EXE** (Device Firmware Upgrade Software)

3.7. Firmware Library Install

3.7.1. Add the “**EasyPOD.LIB**“ into your own project. (example in Fig-4)

3.7.1.1. **InFlag, in “ Extern.h “** (This flag indicates the data has been received in **InBuffer**)

3.7.1.2. **InLen, in “ Extern.h “** (Indicate the data size in **InBuffer**)

3.7.1.3. **InBuffer[64] , in “ Extern.h “** (Data Buffer)

3.7.1.4. **Initial();, in “ Extern.h “** (This function enables USB and will be called in user project)

3.7.1.5. **USB_Read_Data_Complete();, in “ Extern.h “** (Execute this function will release data buffer for next data transfer from PC)

3.7.1.6. **USB_Send_Data_To_PC(Len, Buffer); , in “ Extern.h “** (This function will send ” **Len** ” size Data to PC)

3.7.1.7. **USB_Event(); in “ Extern.h “** (Notice the USB power event from Host)

3.7.1.7.1. **Suspend :** Suspend event from the Host

3.7.1.7.2. Wakeup : Wakeup event from the Host

3.7.1.7.3. Reset : Reset event from the Host

3.7.1.7.4. EmuOK: USB enumeration OK

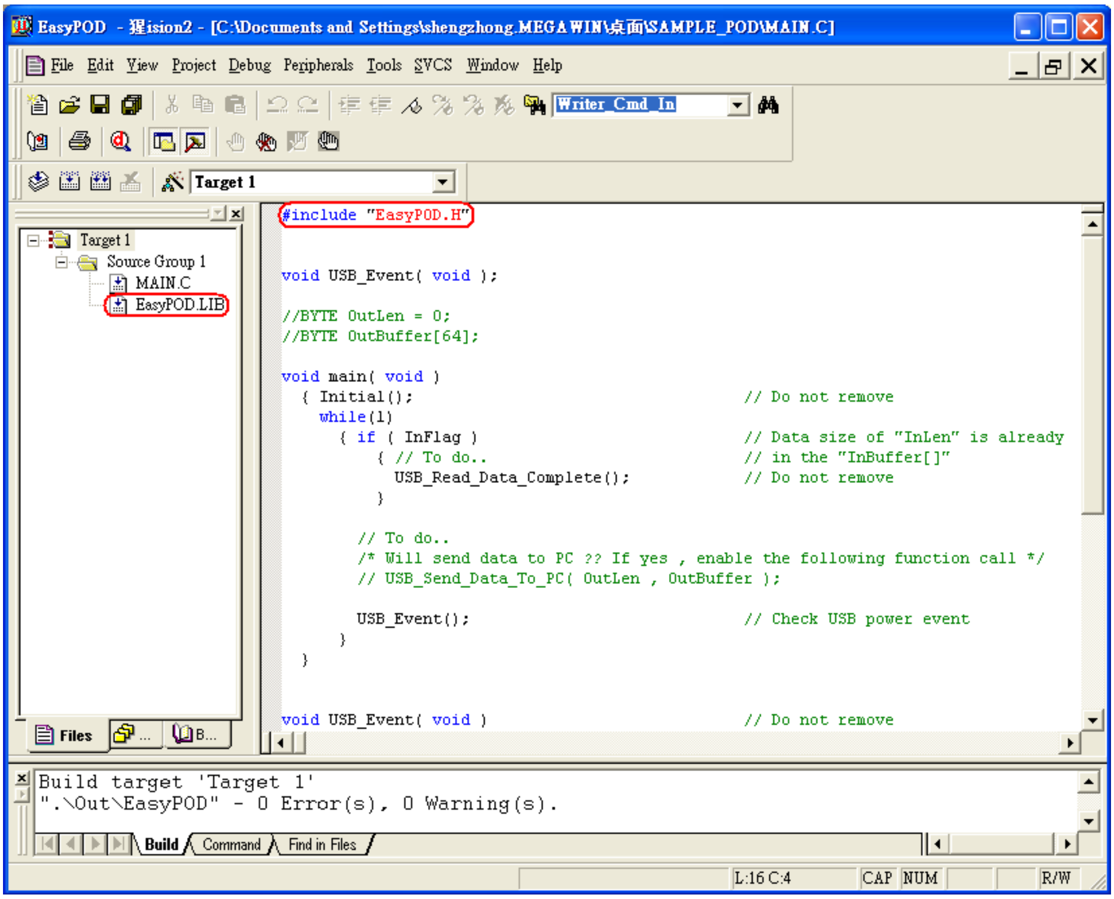
3.7.2. Include the Header file “**EasyPOD.H**“ in the source modules which will use parameter or function call (Fig-4). , The following items could be modified by user application.

3.7.2.1. USB_VID, in “ Define.h” (The VID, 0x0E6Ah, is registered under Megawin Technology Co., Ltd. at **USB-IF**, Any third party needs the written approval from Megawin in order to use this VID)

3.7.2.2. MF_STRING, in “ Define.h” (Define for Manufacture String are supported)

3.7.2.3. PD_STRING, in “ Define.h” (Define for Product String are supported)

3.7.2.4. SN_STRING, in “ Define.h” (Define for SerialNumber String are supported)


EasyPOD - 礙ision2 - [C:\Documents and Settings\shengzhong.MEGAWIN\桌面\SAMPLE_POD\MAIN.C]
File Edit View Project Debug Peripherals Tools SVCS Window Help
Writer_Cmd_In
Target 1
Target 1
Source Group 1
MAIN.C
EasyPOD.LIB
Files
B...
Build
Command
Find in Files
L:16 C:4
CAP NUM
R/W

```
#include "EasyPOD.H"


void USB_Event( void );

//BYTE OutLen = 0;
//BYTE OutBuffer[64];

void main( void )
  { Initial();                                    // Do not remove
    while(1)
      { if ( InFlag )                             // Data size of "InLen" is already
          { // To do..                            // in the "InBuffer[]"
            USB_Read_Data_Complete();             // Do not remove
          }

        // To do..
        /* Will send data to PC ?? If yes , enable the following function call */
        // USB_Send_Data_To_PC( OutLen , OutBuffer );

        USB_Event();                              // Check USB power event
      }
  }


void USB_Event( void )                            // Do not remove
```

```
Build target 'Target 1'
".\Out\EasyPOD" - 0 Error(s), 0 Warning(s).
```

Fig-4 ~/Megawin Easy USB\EasyPOD\SampleCode\MAIN.C

3.8. Firmware Control Flow

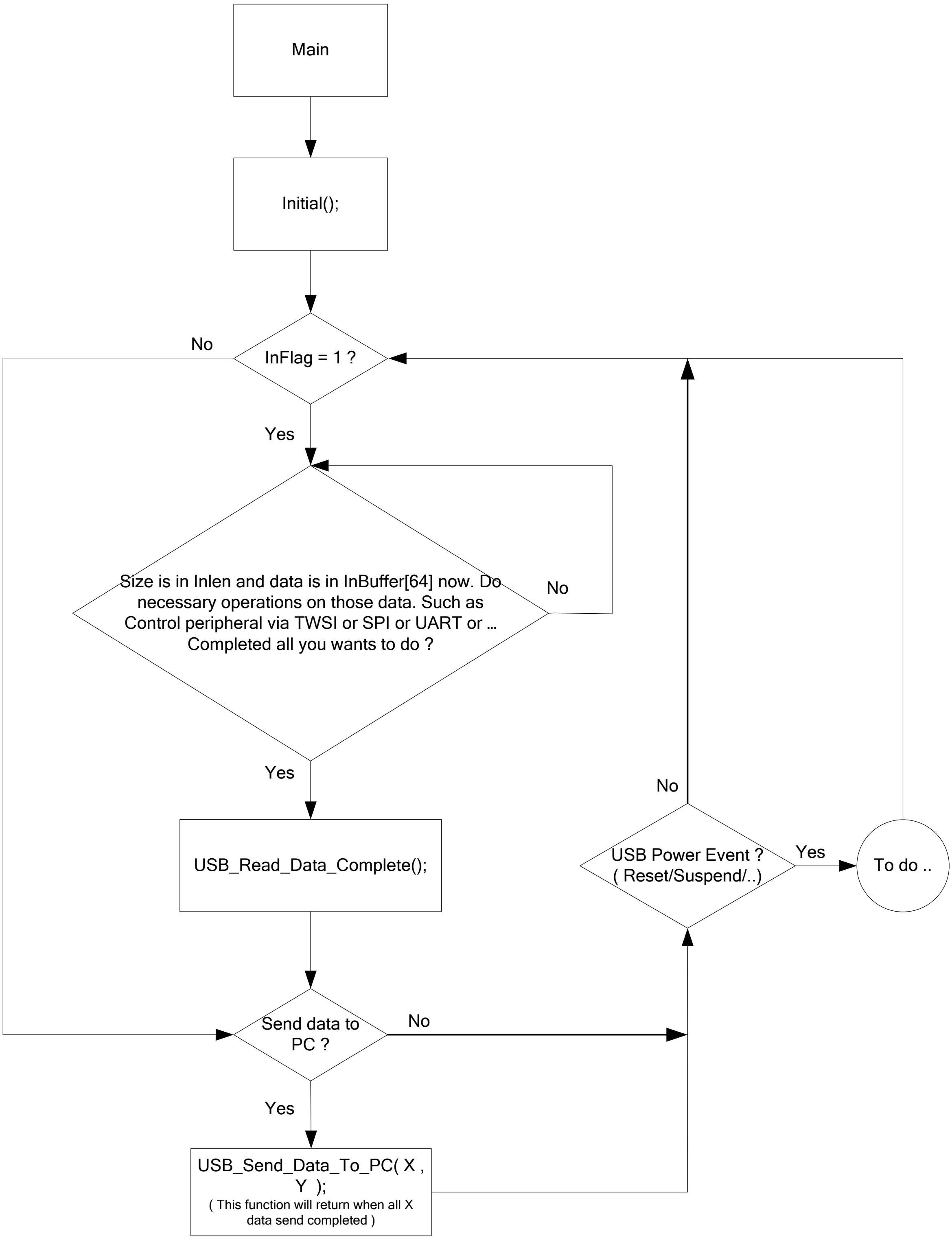

Main
Initial();
InFlag = 1 ?
No
Yes
Size is in Inlen and data is in InBuffer[64] now. Do necessary operations on those data. Such as Control peripheral via TWSI or SPI or UART or ... Completed all you wants to do ?
No
Yes
USB_Read_Data_Complete();
No
USB Power Event ? (Reset/Suspend/..)
Yes
To do ..
Send data to PC ?
No
Yes
USB_Send_Data_To_PC(X , Y);
(This function will return when all X data send completed)

Fig-5

3.9. Used MCU resource

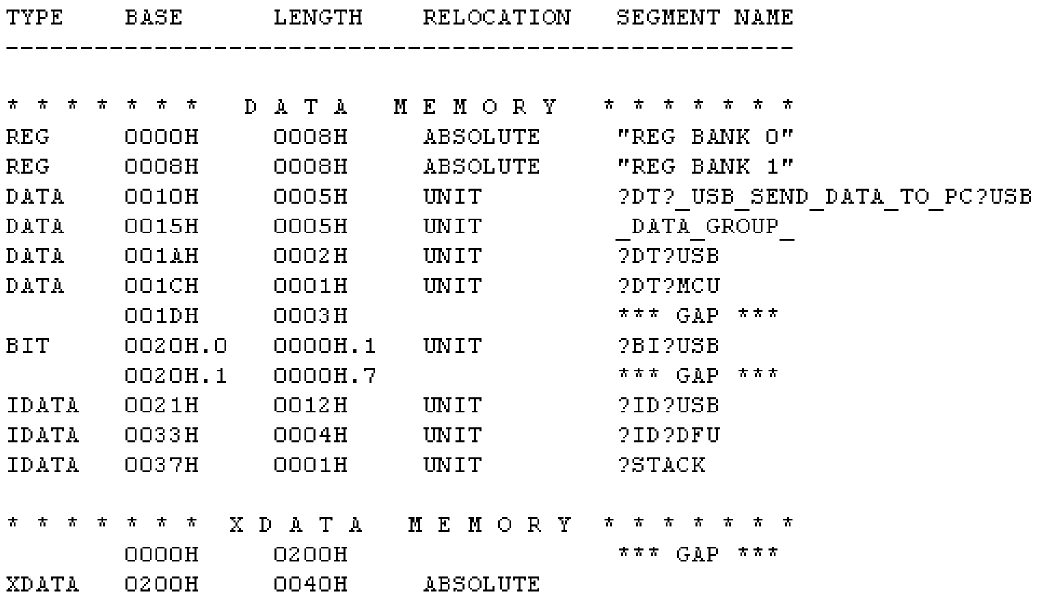
3.9.1. Direct Data Memory : 14 bytes

3.9.2. Indirect Data Memory : 22 bytes

3.9.3. eXternal Data Memory : 64 bytes

3.9.4. Sample Code size : 3651 bytes

3.9.5. USB ISR use “REG BANK 1”



```
TYPE    BASE      LENGTH    RELOCATION   SEGMENT NAME
-----------------------------------------------------

* * * * * * *   D A T A   M E M O R Y   * * * * * * *
REG     0000H     0008H     ABSOLUTE     "REG BANK 0"
REG     0008H     0008H     ABSOLUTE     "REG BANK 1"
DATA    0010H     0005H     UNIT         ?DT?_USB_SEND_DATA_TO_PC?USB
DATA    0015H     0005H     UNIT         _DATA_GROUP_
DATA    001AH     0002H     UNIT         ?DT?USB
DATA    001CH     0001H     UNIT         ?DT?MCU
        001DH     0003H                  *** GAP ***
BIT     0020H.0   0000H.1   UNIT         ?BI?USB
        0020H.1   0000H.7                *** GAP ***
IDATA   0021H     0012H     UNIT         ?ID?USB
IDATA   0033H     0004H     UNIT         ?ID?DFU
IDATA   0037H     0001H     UNIT         ?STACK

* * * * * * *  X D A T A   M E M O R Y  * * * * * * *
        0000H     0200H                  *** GAP ***
XDATA   0200H     0040H     ABSOLUTE
```

Fig-6

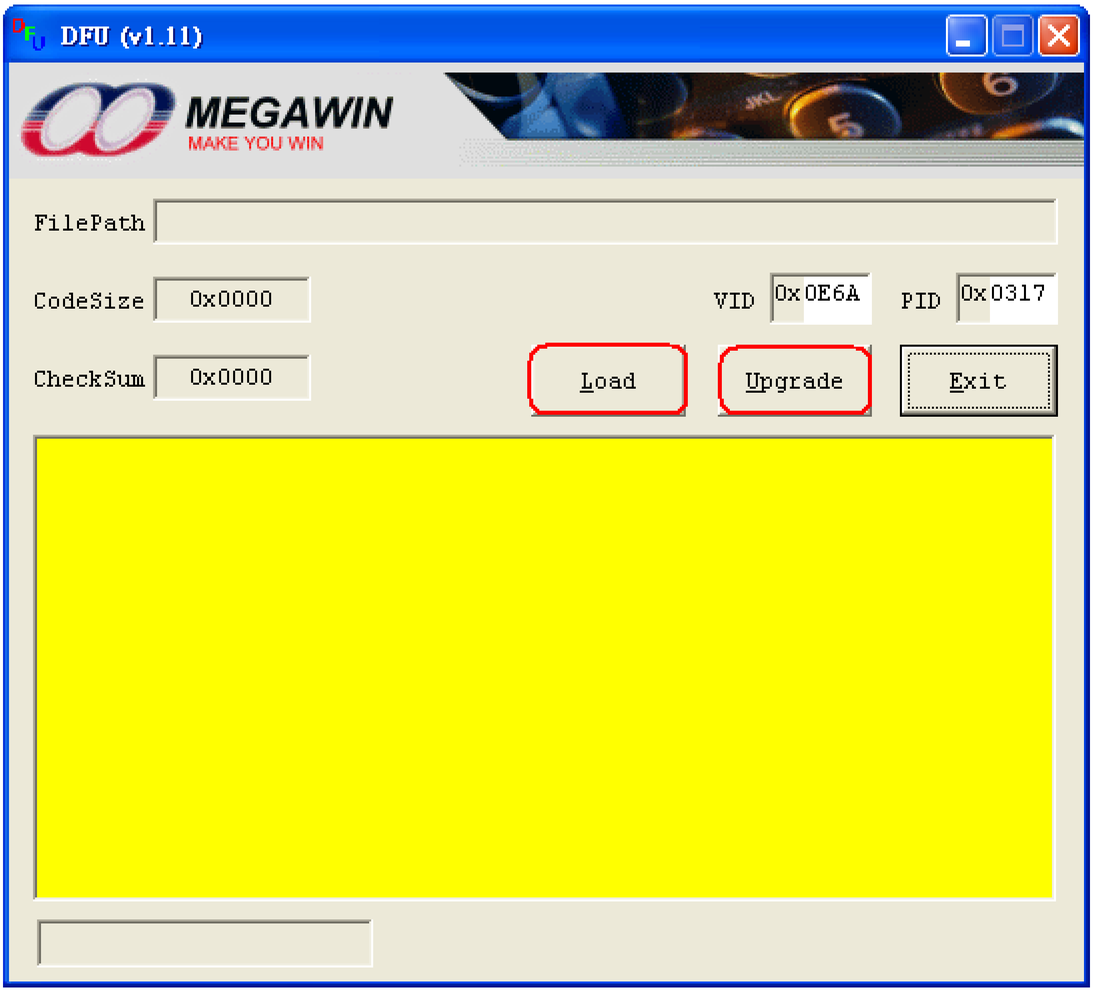
3.10. Using Device Firmware Upgrade

After firmware development, user could update the device firmware through this software tool by following procedures.

3.10.1. Run " **DFU.EXE"** (**Fig-7)**

3.10.2. **"Load"** file which you want to upgrade

3.10.3. **"Upgrade"** to process upgrade procedure



FilePath

CodeSize 0x0000

CheckSum 0x0000

VID 0x0E6A

PID 0x0317

Fig-7

4. Example Application

- 4.1. USB to MCU GPIO
- 4.2. USB to UART
- 4.3. USB to Parallel Port
- 4.4. USB to SPI
- 4.5. USB to TWSI
- 4.6. USB to user's interface

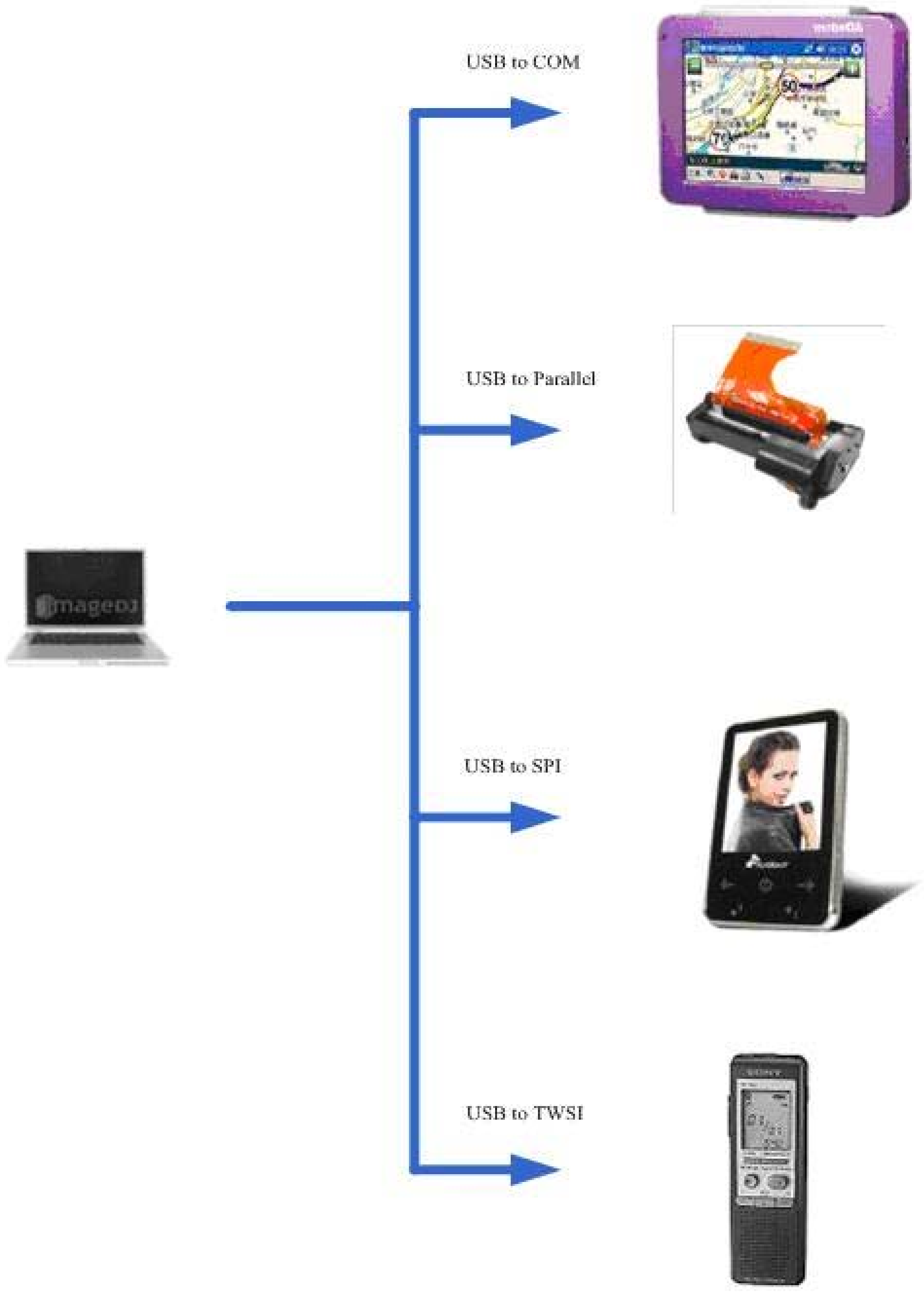


Fig-8

5. Obtaining The Parts

The **MA84G564** device comes in a 64 pin surface mount LQFP-64 package. For this solution, some passives, and the USB socket. Of course, customers could obtain it from Megawin in Taiwan. Please, visit the Megawin's website at ***http://www.megawin.com.tw*** for the latest details on pricing and availability

6. Revision History

Revision	Description	Date
v1.00	Release version	2011/03/01