法政大学工学部システム制御工学科小林一行研究室　資料
http://www.gerox.com
Gerox(c) 5/30/2003

汎用 USB-IO for MATLAB

ここでは、HID デバイスとして認識するモルフィー企画の USB-IO を MATLAB 上で使うための手順について説明する。
モルフィー企画の USB-IO は、HID デバイスとして認識するため、特別なドライバソフトはいらない。
そのため、USB－IO とそれの入出力を行うための usbg.dll（このドキュメント内で作成する。）さえあれば、８ビット＋４ビットのデータの入出力が可能になる。

MATLAB **用の** MEX **プログラミング手順**

STEP.1
外部ハードウェアとの通信を確実にしておかないとどちらが間違えているのかわからないため、まず、DOS 環境で実行できる C プログラムを作成する。（main 関数を含む形で作成する。）
STEP.2
初期化関数、入力関数、出力関数の３つの機能に分け関数化する。
STEP.3
main 関数をコメントアウトし mexFunction 関数から引数に応じ適切に入出力関数を呼び出せるようにプログラミングしていく。

USB-IO **を使うためのプログラミング環境設定**

汎用 USB のドライバを作成するためには VisualC++ 6.0 に Win2000DDK をインストールする必要がある。また、ライブラリとして setupapi.lib,hid.lib を使用するため、環境変数は以下のものを追加する。

PATH	C:¥Program Files¥Microsoft Visual Studio¥Common¥Tools¥WinNT; C:¥Program Files¥Microsoft Visual Studio¥Common¥MSDev98¥Bin; C:¥Program Files¥Microsoft Visual Studio¥Common¥Tools; C:¥Program Files¥Microsoft Visual Studio¥VC98¥bin
LIB	C:¥Program Files¥Microsoft Visual Studio¥VC98¥mfc¥lib; C:¥Program Files¥Microsoft Visual Studio¥VC98¥lib; c:¥ntddk¥lib; c:¥ntddk¥libchk¥i386
INCLUDE	C:¥Program Files¥Microsoft Visual Studio¥VC98¥atl¥include; C:¥Program Files¥Microsoft Visual Studio¥VC98¥mfc¥include; C:¥Program Files¥Microsoft Visual Studio¥VC98¥include; c:¥ntddk¥inc

DOS プログラムの作成
Visual　C++での環境が整ったら、usbtest1.c をコンパイル実行してみよう。
コンパイル実行は、
cl usbtest1.c
とすればよい。このプログラムでは、HID インターフェースの GUID を取り出すプログラムである。
なお、ここで使っている pragma ディレクティブは、コンパイルする際の外部ライブラリをリンクするためのコードである。usbtest1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <windows.h>
#include <setupapi.h>
#include <hidsdi.h>
#pragma comment(lib, "setupapi.lib")
#pragma comment(lib, "hid.lib")
void main(void){
        GUID hidGuid;
        HidD_GetHidGuid(&hidGuid);
        printf("%x %x %x¥n",
          (unsigned long)hidGuid.Data1,(unsigned long)hidGuid.Data2,
          (unsigned long)hidGuid.Data3);
}
```

コンパイル実行結果

```
C:¥>cl usbtest1.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8804 for 80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.
usbtest1.c
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
/out:usbtest1.exe
usbtest1.obj
C:¥>usbtest1
4d1e55b2 f16f 11cf
```

この番号が GUID の一部である、実は、hidGuid.Data4[8]もあるが、これは MAC アドレスなどマシンの特定に使われるものらしい。（詳しいことは未確認）なお、GUID はマシンによりユニークな番号が割り当てられるためここで示した番号とは異なるかもしれない。GUID が取得できることがわかったのでさらにプログラムを拡張していく。

usbtest2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <windows.h>
#include <setupapi.h>
#include <hidsdi.h>
#pragma comment(lib, "setupapi.lib")
#pragma comment(lib, "hid.lib")
void main(void){
	int i,j;
	int count;
	GUID hidGuid;
	HDEVINFO dis;
	BOOL r;
	DWORD size;
	SP_DEVICE_INTERFACE_DATA sdid;
	PSP_DEVICE_INTERFACE_DETAIL_DATA_A pdidd = NULL;
	size_t didd_size = 0;
	HANDLE dh,fd;
	HIDD_ATTRIBUTES attr;
	size_t len;
	char buffer[255];
	HidD_GetHidGuid(&hidGuid);
	dis = SetupDiGetClassDevsA(&hidGuid, NULL, 0, DIGCF_PRESENT | DIGCF_DEVICEINTERFACE );
	count = 0;
	for (i = 0;;i++){
		memset(&sdid,0,sizeof(sdid));
		sdid.cbSize = sizeof( SP_DEVICE_INTERFACE_DATA );
		r = SetupDiEnumDeviceInterfaces(dis,NULL,&hidGuid,i,&sdid);
		if (!r) break;
		size = 0;
		r = SetupDiGetDeviceInterfaceDetailA(dis,&sdid,NULL,0,&size,NULL);
		if ( r || GetLastError() != ERROR_INSUFFICIENT_BUFFER ) continue;
		if ( size > didd_size ) {
			didd_size = ( size + 15 ) & ~15;
			pdidd = malloc(didd_size);
		}
		memset(pdidd,0,didd_size);
		pdidd->cbSize = sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA_A);
		r = SetupDiGetDeviceInterfaceDetailA(dis,&sdid,pdidd,size,NULL,NULL);
		if ( !r ) continue;
		dh = CreateFileA( ( LPCSTR )pdidd->DevicePath,GENERIC_READ |
GENERIC_WRITE,FILE_SHARE_READ |
FILE_SHARE_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
		if (dh == INVALID_HANDLE_VALUE ) continue;
		memset(&attr,0,sizeof(HIDD_ATTRIBUTES));
		attr.Size = sizeof(HIDD_ATTRIBUTES);
		HidD_GetAttributes(dh,&attr);
		CloseHandle(dh);
		printf("%s¥n",(LPCSTR)pdidd->DevicePath);
		printf("%x %x %x",attr.VendorID,attr.ProductID,attr.VersionNumber);
		count++;
	}
	SetupDiDestroyDeviceInfoList( dis );
	}
}
```

このプログラムでは、USB-IO へのファイルハンドル名とベンダ ID、プロダクト ID を検出、表示している。ここで、HID タイプの USB デバイスが複数接続されていた場合、これらの出力も複数出るようにプログラミングしてある。（プログラム途中のため malloc したままで free はしていない。）

```
C:¥>cl usbtest2.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8804 for 80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.
usbtest2.c
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
```

```
/out:usbtest2.exe
usbtest2.obj
C:¥ >usbtest
4d1e55b2 f16f 11cf
¥¥?¥hid#vid_0bfe&pid_1003#6&99bfe71&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030
}
bfe 1003 1
C:¥>
```

ハンドルのファイル名は、
¥¥?¥hid#vid_0bfe&pid_1003#6&99bfe71&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030
}
VenderID は 0xbfe,ProductID が 0x1003,VersionNumber が 1
であることがわかる。ファイルのハンドル名がわかれば HID デバイス情報はいらないので SetupDiDestroyDeviceInfoList(dis);により処理する。
このプログラムでは、最後に検出した HID デバイスを使うようになっている。もししっかり検出、処理したいのであれば、VenderID、ProductID などをチェックしてやると良い。（ここでは、表示をしているだけ）
ここまでくれば、後は、ファイルの入出力関数によりデータをやり取りすることができる。
以下のプログラムでは、CreateFile,ReadFile,WriteFile によりデータのやり取りを行っている。この USB-IO では、1 回に送るデータは９バイトであり、その中に命令、ポート指定 Read,Write、そのパラメータを入力し読み書きするようになっている。
usbtest3.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <windows.h>
#include <setupapi.h>
#include <hidsdi.h>
#pragma comment(lib, "setupapi.lib")
#pragma comment(lib, "hid.lib")
void main(void){
	int i,j;
	int count;
	GUID hidGuid;
	HDEVINFO dis;
	BOOL r;
	DWORD size;
	SP_DEVICE_INTERFACE_DATA sdid;
	PSP_DEVICE_INTERFACE_DETAIL_DATA_A pdidd = NULL;
	size_t didd_size = 0;
	HANDLE dh,fd;
	HIDD_ATTRIBUTES attr;
	size_t len;
	char buffer[255];
	HidD_GetHidGuid(&hidGuid);
	printf("%x %x %x¥n",(unsigned long)hidGuid.Data1,(unsigned long)hidGuid.Data2,(unsigned long)hidGuid.Data3);
	dis = SetupDiGetClassDevsA(&hidGuid, NULL, 0, DIGCF_PRESENT | DIGCF_DEVICEINTERFACE );
	count = 0;
	for (i = 0;;i++){
		memset(&sdid,0,sizeof(sdid));
		sdid.cbSize = sizeof( SP_DEVICE_INTERFACE_DATA );
		r = SetupDiEnumDeviceInterfaces(dis,NULL,&hidGuid,i,&sdid);
		if (!r) break;
		size = 0;
		r = SetupDiGetDeviceInterfaceDetailA(dis,&sdid,NULL,0,&size,NULL);
		if ( r || GetLastError() != ERROR_INSUFFICIENT_BUFFER ) continue;
		if ( size > didd_size ) {
			didd_size = ( size + 15 ) & ~15;
			pdidd = malloc(didd_size);
		}
		memset(pdidd,0,didd_size);
		pdidd->cbSize = sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA_A);
		r = SetupDiGetDeviceInterfaceDetailA(dis,&sdid,pdidd,size,NULL,NULL);
		if ( !r ) continue;
		dh = CreateFileA( (LPCSTR)pdidd->DevicePath,GENERIC_READ |
GENERIC_WRITE,FILE_SHARE_READ |
FILE_SHARE_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
		if (dh == INVALID_HANDLE_VALUE ) continue;
		memset(&attr,0,sizeof(HIDD_ATTRIBUTES));
```

```
		attr.Size = sizeof(HIDD_ATTRIBUTES);
		HidD_GetAttributes(dh,&attr);
		CloseHandle(dh);
		printf("%s¥n",(LPCSTR)pdidd->DevicePath);
		printf("%x %x %x",attr.VendorID,attr.ProductID,attr.VersionNumber);
		count++;
	}
	SetupDiDestroyDeviceInfoList( dis );
////////////////////////////
	fd = CreateFile((LPCSTR)pdidd->DevicePath,GENERIC_READ | GENERIC_WRITE,FILE_SHARE_READ |
FILE_SHARE_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
	printf("[%x]",fd);
getch();
////////////////////////////////////
	printf("¥n");
	for (j=0;j <255;j++) {
		buffer[0]=0;
		buffer[1]=0x01;
		buffer[2]=j;
		buffer[3]=0;
		buffer[4]=0;
		buffer[5]=0;
		buffer[6]=0;
		buffer[7]=0;
		buffer[8]=0;
		count = 9;
		r=WriteFile((HANDLE)fd,buffer,9,&count,0);
		for (i = 0;i < 8;i++) printf("%x ",buffer[i]);
		printf("¥n");
		r=ReadFile((HANDLE)fd,buffer,9,&count,0);
		for (i = 0;i < 8;i++) printf("%x ",buffer[i]);
		printf("¥n");
	}
	CloseHandle( ( HANDLE )fd );
	free(pdidd);
}
```

以下に簡単な表を示す。buffer[1]の内容によりモードが変わる。

buffer[1]=01	buffer[1]=02	buffer[1]=03	buffer[1]=04
buffer[2]をポート 0 に書き込み	buffer[2]をポート 1 に書き込み(ただし下位４ビットのみ)	buffer[2]をポート 0 に読み込み ただし、読み込む前に対応するビットを１に書き込んでおく必要あり。	buffer[2]をポート 1 に読み込み(ただし下位４ビットのみ) ただし、読み込む前に対応するビットを１に書き込んでおく必要あり。

以上の注意点を元に MEX 関数化を考えていけばよい。
この場合、プログラム名は usbg.c とするとし、仕様は、以下のように定義する。
usbg(0,255);% ポート 0 に 255 を書き込み
data=usbg(0) % 変数 data にポート 0 の内容を読み込む。
まず、このようなプログラムを作成するために、
int usb_write(int port,int data),
int usb_read(int port)
int usb_init(void);
int usb_close(void);
の 4 つの機能に分けてプログラムの作成を行ってみる。
usbtest4.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <windows.h>
#include <setupapi.h>
#include <hidsdi.h>
#pragma comment(lib, "setupapi.lib")
#pragma comment(lib, "hid.lib")
int usb_write(int port,int data,HANDLE fd){
	char buffer[9];
	int ch = 2;
	int count = 9;
	if (port == 0) ch = 1;
```

```
	buffer[0]=0;
	buffer[1]=(char)ch;
	buffer[2]=(char)data;
	buffer[3]=0xff;
	buffer[4]=0xff;
	buffer[5]=0xff;
	buffer[6]=0xff;
	buffer[7]=0xff;
	buffer[8]=0xff;
	return WriteFile((HANDLE)fd,buffer,9,&count,0);
}
int usb_read(int port,HANDLE fd){
	char buffer[9];
	int ch = 4;
	int count = 9;
	if (port == 0) ch = 3;
	usb_write(port,0xff,fd);
	buffer[0]=0;
	buffer[1]=(char)ch;
	buffer[2]=0xff;
	buffer[3]=0xff;
	buffer[4]=0xff;
	buffer[5]=0xff;
	buffer[6]=0xff;
	buffer[7]=0xff;
	buffer[8]=0xff;
	WriteFile((HANDLE)fd,buffer,9,&count,0);
	ReadFile((HANDLE)fd,buffer,9,&count,0);
	return buffer[2];
}
HANDLE usb_init(void){
	GUID hidGuid;
	HDEVINFO dis;
	int count,i;
	BOOL r;
	DWORD size;
	SP_DEVICE_INTERFACE_DATA sdid;
	PSP_DEVICE_INTERFACE_DETAIL_DATA_A pdidd = NULL;
	size_t didd_size = 0;
	HANDLE dh,fd;
	HIDD_ATTRIBUTES attr;
	size_t len;
	char buffer[255];
	HidD_GetHidGuid(&hidGuid);
//	printf("%x %x %x¥n",(unsigned long)hidGuid.Data1,(unsigned long)hidGuid.Data2,(unsigned long)hidGuid.Data3);
	dis = SetupDiGetClassDevsA(&hidGuid, NULL, 0, DIGCF_PRESENT | DIGCF_DEVICEINTERFACE );
	count = 0;
	for (i = 0;;i++){
		memset(&sdid,0,sizeof(sdid));
		sdid.cbSize = sizeof( SP_DEVICE_INTERFACE_DATA );
		r = SetupDiEnumDeviceInterfaces(dis,NULL,&hidGuid,i,&sdid);
		if (!r) break;
		size = 0;
		r = SetupDiGetDeviceInterfaceDetailA(dis,&sdid,NULL,0,&size,NULL);
		if ( r || GetLastError() != ERROR_INSUFFICIENT_BUFFER ) continue;
		if ( size > didd_size ) {
			didd_size = ( size + 15 ) & ~15;
			pdidd = malloc(didd_size);
		}
		memset(pdidd,0,didd_size);
		pdidd->cbSize = sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA_A);
		r = SetupDiGetDeviceInterfaceDetailA(dis,&sdid,pdidd,size,NULL,NULL);
		if ( !r ) continue;
		dh	=	CreateFileA(	(	LPCSTR	)pdidd->DevicePath,GENERIC_READ	|
GENERIC_WRITE,FILE_SHARE_READ	|
FILE_SHARE_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
		if (dh == INVALID_HANDLE_VALUE ) continue;
		memset(&attr,0,sizeof(HIDD_ATTRIBUTES));
		attr.Size = sizeof(HIDD_ATTRIBUTES);
		HidD_GetAttributes(dh,&attr);
		CloseHandle(dh);
```

```
//              printf("%s¥n",(LPCSTR)pdidd->DevicePath);
//              printf("%x %x %x",attr.VendorID,attr.ProductID,attr.VersionNumber);
                count++;
        }
        SetupDiDestroyDeviceInfoList( dis );
        fd = CreateFile((LPCSTR)pdidd->DevicePath,GENERIC_READ | GENERIC_WRITE,FILE_SHARE_READ |
FILE_SHARE_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
//      printf("[%x]",fd);
        free(pdidd);
        return (HANDLE)fd;
}
void usb_close(HANDLE fd) {
        CloseHandle(( HANDLE )fd);
}

void main(void){
        int i;
        HANDLE fd = usb_init();
        for(i = 0;i < 20;i++) {
                printf("%x",usb_read(0,fd));
                usb_write(0,i,fd);
                getch();
        }
        usb_close(fd);
}
```

ファイルハンドルｆｄはすべての関数に共通の変数であるため、関数の引数となるように変更した。（グローバル変数として定義してもよいが、変数として明示した方がよい。以下の Mex 関数では、グローバル変数として使っている。）
ここまでできたら後は、main 関数を Mex 関数化すればよい。MEX 関数化するには、include に mex.h を入れて、MATLAB 上では printf 関数は使えないため print 文関連は、すべて mexPrintf に変換する。

usbg.c

```
// USB-IO for MATLAB by Gerox(c) 2003
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <windows.h>
#include <setupapi.h>
#include <hidsdi.h>
#pragma comment(lib, "setupapi.lib")
#pragma comment(lib, "hid.lib")
int usb_write(int port,int data,HANDLE fd){
        char buffer[9];
        int ch = 2;
        int count = 9;
        if (port == 0) ch = 1;
        buffer[0]=(char)0;
        buffer[1]=(char)ch;
        buffer[2]=(char)data;
        buffer[3]=(char)0xff;
        buffer[4]=(char)0xff;
        buffer[5]=(char)0xff;
        buffer[6]=(char)0xff;
        buffer[7]=(char)0xff;
        buffer[8]=(char)0xff;
        return WriteFile((HANDLE)fd,buffer,9,&count,0);
}
int usb_read(int port,HANDLE fd){
        char buffer[9];
        int ch = 4;
        int count = 9;
        if (port == 0) ch = 3;
//      usb_write(port,0xff,fd);
        buffer[0]=(char)0;
        buffer[1]=(char)ch;
        buffer[2]=(char)0xff;
        buffer[3]=(char)0xff;
        buffer[4]=(char)0xff;
        buffer[5]=(char)0xff;
        buffer[6]=(char)0xff;
```

```
	buffer[7]=(char)0xff;
	buffer[8]=(char)0xff;
	WriteFile((HANDLE)fd,buffer,9,&count,0);
	ReadFile((HANDLE)fd,buffer,9,&count,0);
	return buffer[2];
}
HANDLE usb_init(void){
	GUID hidGuid;
	HDEVINFO dis;
	int count,i;
	BOOL r;
	DWORD size;
	SP_DEVICE_INTERFACE_DATA sdid;
	PSP_DEVICE_INTERFACE_DETAIL_DATA_A pdidd = NULL;
	size_t didd_size = 0;
	HANDLE dh,fd;
	HIDD_ATTRIBUTES attr;
	HidD_GetHidGuid(&hidGuid);
//	printf("%x %x %x¥n",(unsigned long)hidGuid.Data1,(unsigned long)hidGuid.Data2,(unsigned long)hidGuid.Data3);
	dis = SetupDiGetClassDevsA(&hidGuid, NULL, 0, DIGCF_PRESENT | DIGCF_DEVICEINTERFACE );
	count = 0;
	for (i = 0;;i++){
		memset(&sdid,0,sizeof(sdid));
		sdid.cbSize = sizeof( SP_DEVICE_INTERFACE_DATA );
		r = SetupDiEnumDeviceInterfaces(dis,NULL,&hidGuid,i,&sdid);
		if (!r) break;
		size = 0;
		r = SetupDiGetDeviceInterfaceDetailA(dis,&sdid,NULL,0,&size,NULL);
		if ( r || GetLastError() != ERROR_INSUFFICIENT_BUFFER ) continue;
		if ( size > didd_size ) {
			didd_size = ( size + 15 ) & ~15;
			pdidd = malloc(didd_size);
		}
		memset(pdidd,0,didd_size);
		pdidd->cbSize = sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA_A);
		r = SetupDiGetDeviceInterfaceDetailA(dis,&sdid,pdidd,size,NULL,NULL);
		if ( !r ) continue;
		dh	=	CreateFileA(	(	LPCSTR	)pdidd->DevicePath,GENERIC_READ	|
GENERIC_WRITE,FILE_SHARE_READ	|
FILE_SHARE_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
		if (dh == INVALID_HANDLE_VALUE ) continue;
		memset(&attr,0,sizeof(HIDD_ATTRIBUTES));
		attr.Size = sizeof(HIDD_ATTRIBUTES);
		HidD_GetAttributes(dh,&attr);
		CloseHandle(dh);
		mexPrintf("%s¥n",(LPCSTR)pdidd->DevicePath);

	mexPrintf("VendorID=%x¥nProductID=%x¥nVersion=%x¥n",attr.VendorID,attr.ProductID,attr.VersionNumber);
		count++;
	}
	SetupDiDestroyDeviceInfoList( dis );
	fd	=	CreateFile((LPCSTR)pdidd->DevicePath,GENERIC_READ	|	GENERIC_WRITE,FILE_SHARE_READ	|
FILE_SHARE_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
//	printf("[%x]",fd);
	mexPrintf("Ready to use¥n");
	free(pdidd);
	return (HANDLE)fd;
}
void usb_close(HANDLE fd) {
	CloseHandle(( HANDLE )fd);
}
#if 0
void main(void){
	int i;
	HANDLE fd = usb_init();
	for(i = 0;i < 20;i++) {
		printf("%x",usb_read(0,fd));
		usb_write(0,i,fd);
		getch();
	}
	usb_close(fd);
```

```
}
#else
#include "mex.h"
static int initialized = 0;
static HANDLE Fd;
void cLeanup(void) {
	if(initialized != 0) {
		usb_close(Fd);
		mexPrintf("MEX-file is safely terminated.¥n");
		initialized = 0;
	}
}
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray * prhs[]) {
	double data,*pldata;
	if(nlhs > 1) {
		mexErrMsgTxt("one output must be required.");return;
	}
	if(initialized == 0) {
		Fd = usb_init();
		mexAtExit(cLeanup);
		initialized++;
	}
//	nrhs==1 read
	if(nrhs == 1) {
		data = (double)usb_read((int)mxGetScalar(prhs[0]),Fd);
	} else {
		if(nrhs == 2) {
			data =(double) usb_write((int)mxGetScalar(prhs[0]),(int)mxGetScalar(prhs[1]),Fd);
		}	else return;
	}
	plhs[0] = mxCreateDoubleMatrix(1,1,mxREAL);
	pldata = mxGetPr(plhs[0]);
	pldata[0] = data;
	return;
}
#endif
```

MATLAB での使用例

初めての起動のとき、以下のメッセージが出る。

```
>> usbg(0,255)
¥¥?¥hid#vid_0bfe&pid_1003#6&99bfe71&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
VendorID=bfe
ProductID=1003
Version=1
Ready to use
ans =
	1
```

ここでは、ポート 0 番地に 255 を書き込んでいる例である。2 回目以降は、ベンダ ID などは出力されない。

```
>> usbg(0,8)

ans =

	1
```

なお、USB コネクタを抜くときは、必ず clear usbg を実行してから抜くこと、そうしないとメモリリークが起こりシステムが不安定となる原因となる。