UHF 帯 RFID リーダ・ライタを
FTDI でリセットする方法

2020 年 10 月 2 日　　第 1.0.0 版

株式会社アートファイネックス

変更履歴

日付	版数	変更内容
2020/10/02	1.0.0	初版発行

はじめに

本書は、アートファイネックス製 UHF 帯 RFID リーダ・ライタ（以下、リーダ・ライタとします）を FTDI デバイス経由でリセットする方法を説明したものです。

目次

1. はじめに

リーダ・ライタを制御しているときに、何らかのミスやトラブルにより、コマンドを送信しても返答がこなくなる場合があります。そのときはリーダ・ライタをリセットする必要があります。その方法は主に下の 3 つあります。

（1） USB ケーブルを抜き差しします。

（2） USB バスパワーを OFF/ON 制御します。

（3） FTDI デバイスを介してリーダ・ライタにリセット信号を入れます。

このドキュメントでは上の（3）の方法について記します。

2. リセットの原理

リーダ・ライタは USB ポートから受信したデータを［USB⇒シリアル］変換してリーダ・ライタ内部に取り込んでいます。 また、リーダ・ライタ内部から送信したデータを［シリアル⇒USB］変換して USB ポートから送信しています。 リーダ・ライタはこの［USB⇔シリアル］変換のために FTDI 社製のデバイスを使用しています。 ホストからその FTDI デバイスを経由してリーダ・ライタにリセット信号を入れることでリセットします。

※USB 接続されていないリーダ・ライタに対してはリセットできません。

※FTDI デバイスとリーダ・ライタの間にリセット信号線が配線されていないリーダ・ライタはリセットできません。
お使いのリーダ・ライタがリセットに対応しているかどうかは、機種名とシリアル番号でお問い合わせください。

3. Windows の場合

弊社提供の Windows アプリ用 API 関数を利用してリーダ・ライタを制御しているときは、次のように対処します。

・API 関数 so_CommSetup でホスト PC と RFID リーダ・ライタを接続しているときは
API 関数 so_ResetReader や so_ResetReaderAnyTime をお使いください。

・まだ API 関数 so_CommSetup でホスト PC と RFID リーダ・ライタを接続していないときは
API 関数 so_ResetReaderBeforeConnect をお使いください。

4. Windows 以外の場合

FTDI 社から下の OS に対応した関数が提供されています。

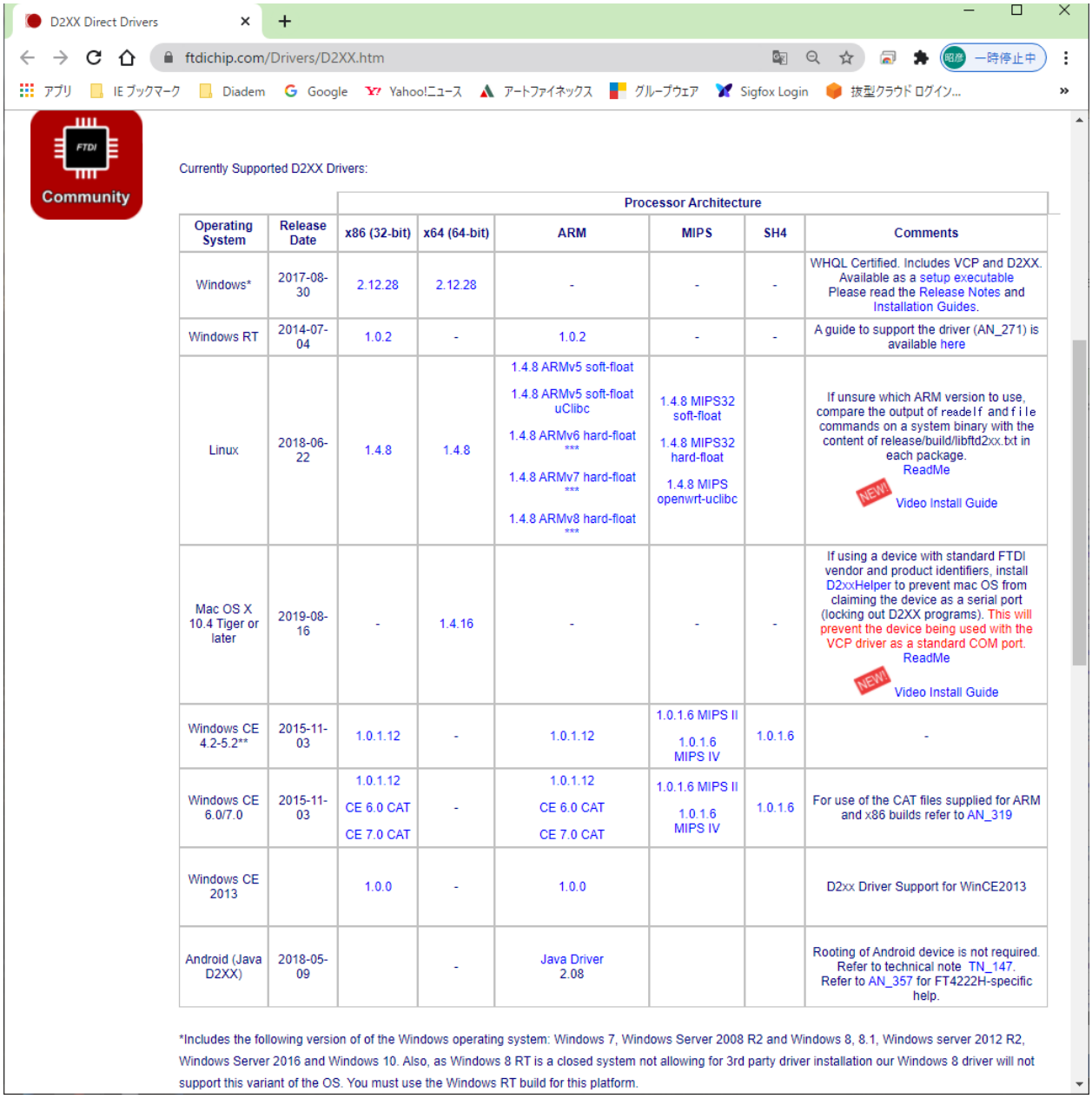
Linux
Mac OS X（10.4 以降）
Windows（2000 以降）
Windows CE（4.2 以降）

この章では、FTDI 社提供の関数を使用して RFID リーダ・ライタをリセットする方法を記します。

4.1. 事前に用意しておくもの

FTDI 社の下の URL からドライバを入手します。その中にある ftd2xx.h と ftd2xx.lib を使用します。

https://www.ftdichip.com/Drivers/D2XX.htm



Operating System	Release Date	Processor Architecture: x86 (32-bit)	x64 (64-bit)	ARM	MIPS	SH4	Comments
Windows*	2017-08-30	2.12.28	2.12.28	-	-	-	WHQL Certified. Includes VCP and D2XX. Available as a setup executable Please read the Release Notes and Installation Guides.
Windows RT	2014-07-04	1.0.2	-	1.0.2	-	-	A guide to support the driver (AN_271) is available here
Linux	2018-06-22	1.4.8	1.4.8	1.4.8 ARMv5 soft-float 1.4.8 ARMv5 soft-float uClibc 1.4.8 ARMv6 hard-float *** 1.4.8 ARMv7 hard-float *** 1.4.8 ARMv8 hard-float ***	1.4.8 MIPS32 soft-float 1.4.8 MIPS32 hard-float 1.4.8 MIPS openwrt-uclibc		If unsure which ARM version to use, compare the output of `readelf` and `file` commands on a system binary with the content of release/build/libftd2xx.txt in each package. ReadMe NEW! Video Install Guide
Mac OS X 10.4 Tiger or later	2019-08-16	-	1.4.16	-	-	-	If using a device with standard FTDI vendor and product identifiers, install D2xxHelper to prevent mac OS from claiming the device as a serial port (locking out D2XX programs). This will prevent the device being used with the VCP driver as a standard COM port. ReadMe NEW! Video Install Guide
Windows CE 4.2-5.2**	2015-11-03	1.0.1.12	-	1.0.1.12	1.0.1.6 MIPS II 1.0.1.6 MIPS IV	1.0.1.6	-
Windows CE 6.0/7.0	2015-11-03	1.0.1.12 CE 6.0 CAT CE 7.0 CAT	-	1.0.1.12 CE 6.0 CAT CE 7.0 CAT	1.0.1.6 MIPS II 1.0.1.6 MIPS IV	1.0.1.6	For use of the CAT files supplied for ARM and x86 builds refer to AN_319
Windows CE 2013		1.0.0	-	1.0.0			D2xx Driver Support for WinCE2013
Android (Java D2XX)	2018-05-09		-	Java Driver 2.08			Rooting of Android device is not required. Refer to technical note TN_147. Refer to AN_357 for FT4222H-specific help.

*Includes the following version of of the Windows operating system: Windows 7, Windows Server 2008 R2 and Windows 8, 8.1, Windows server 2012 R2, Windows Server 2016 and Windows 10. Also, as Windows 8 RT is a closed system not allowing for 3rd party driver installation our Windows 8 driver will not support this variant of the OS. You must use the Windows RT build for this platform.

4.2. FTDI 社提供の関数

FTDI 社の下の URL に関数のドキュメントがあります。この中の関数を使用します。
https://www.ftdichip.com/Support/Documents/ProgramGuides/D2XX_Programmer's_Guide(FT_000071).pdf

4.3. リセットサンプルアプリ

ここでは Visual Studio の Win32 コンソールアプリケーションのときのサンプルソースを記します。

4.3.1. FTDI デバイスが1つだけ接続されている場合

接続されている FTDI デバイスを検出し、検出された 1 つ目の FTDI デバイスに対してリセットします。

```
#include "stdafx.h"
#include <stdlib.h>
#include "ftd2xx.h"

int main(int argc, char *argv[])
{
	FT_STATUS ftStatus;
	DWORD numDevs;
	char Buffer[64];
	HANDLE hc;
	FT_PROGRAM_DATA	ftData;
	char	szManufacturer[32], szManufacturerId[16], szDescription[64], szSerialNumber[16];
	unsigned char	data;

	// 接続されている台数を入手
	ftStatus = FT_ListDevices(&numDevs, NULL, FT_LIST_NUMBER_ONLY);
	if (ftStatus != FT_OK) {
		// エラー処理
		return 0;
	}
	if (numDevs < 1) {	// 1 台も接続されていなかったらエラー
		// エラー処理
```

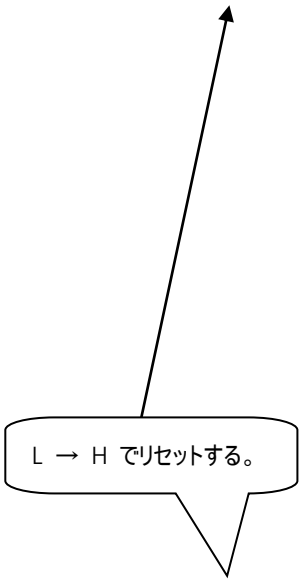
```
    return 0;
}

ftStatus = FT_ListDevices((PVOID)0, Buffer, FT_LIST_BY_INDEX | FT_OPEN_BY_DESCRIPTION);
if (ftStatus != FT_OK) {
    // エラー処理
    return 0;
}
hc = FT_W32_CreateFile((LPCTSTR)Buffer, GENERIC_READ | GENERIC_WRITE, 0,
    NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED |
    FT_OPEN_BY_DESCRIPTION, NULL);
if (hc == INVALID_HANDLE_VALUE) {
    // エラー処理
    return 0;
}

// 以下 リセット処理
memset(&ftData, 0, sizeof(FT_PROGRAM_DATA));
ftData.Manufacturer = szManufacturer;
ftData.ManufacturerId = szManufacturerId;
ftData.Description = szDescription;
ftData.SerialNumber = szSerialNumber;
ftStatus = FT_EE_Read(hc, &ftData);   // 現在の設定値を入手
if (ftStatus != FT_OK) {
    // エラー処理
    goto end;
}
ftData.Signature1 = 0x00000000;
ftData.Signature2 = 0xFFFFFFFF;
ftData.Version = 2;
ftData.Cbus0 = FT_232R_CBUS_IOMODE;   // CBUS0 を I/O モードにする
ftStatus = FT_EE_Program(hc, &ftData);   // 設定値を書込む
if (ftStatus != FT_OK) {
    // エラー処理
    goto end;
}
```

検出された FTDI デバイスの 1 つ目（0 番目）を入手

```
    ftStatus = FT_GetBitMode(hc, &data);
    if (ftStatus != FT_OK) {
        // エラー処理
        goto end;
    }
    data |= 0x10;
    data &= 0xFE;
    ftStatus = FT_SetBitMode(hc, data, FT_BITMODE_CBUS_BITBANG);   // CBUS0 を出力&L
    if (ftStatus != FT_OK) {
        // エラー処理
        goto end;
    }
    Sleep(100);
    ftStatus = FT_GetBitMode(hc, &data);
    if (ftStatus != FT_OK) {
        // エラー処理
        goto end;
    }
    data |= 0x10;
    data |= 0x01;
    ftStatus = FT_SetBitMode(hc, data, FT_BITMODE_CBUS_BITBANG);   // CBUS0 を出力&H
    if (ftStatus != FT_OK) {
        // エラー処理
        goto end;
    }
end:
    FT_W32_CloseHandle(hc);
    return 0;
}
```



4.3.2.FTDI デバイスが複数接続されている場合

複数接続されている場合は、リセットしたい FTDI デバイスのシリアル番号を指定してリセットします。
FTDI デバイスのシリアル番号の入手方法は後述します。

```
                    :
                    :
// 接続されている台数を入手
ftStatus = FT_ListDevices(&numDevs, NULL, FT_LIST_NUMBER_ONLY);
if (ftStatus != FT_OK) {
    // エラー処理
    return 0;
}
if (numDevs < 1) {   // 1 台も接続されていなかったらエラー
    // エラー処理
    return 0;
}
memset(Buffer, 0, sizeof(Buffer));
memcpy(Buffer, argv[1], strlen(argv[1]));
hc = FT_W32_CreateFile((LPCTSTR)Buffer, GENERIC_READ | GENERIC_WRITE, 0, NULL,
    OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED |
    FT_OPEN_BY_SERIAL_NUMBER, NULL);
if (hc == INVALID_HANDLE_VALUE) {
    // エラー処理
    return 0;
}

// 以下 リセット処理
memset(&ftData, 0, sizeof(FT_PROGRAM_DATA));
                    :
                    :
```

引数で渡されたシリアル番号

シリアル番号指定でハンドルを入手する

4.4. FTDI デバイスのシリアル番号取得アプリ

```
#include "stdafx.h"
#include <stdlib.h>
#include "ftd2xx.h"

int main(int argc, char *argv[])
{
	FT_STATUS ftStatus;
	DWORD numDevs, Flags, ID, Type, LocId, i;
	char SerialNumber[16], Description[64];
	FT_HANDLE	ftHandle;

	// 接続されている台数を入手
	numDevs = 0;
	ftStatus = FT_CreateDeviceInfoList(&numDevs);
	if (ftStatus == FT_OK)
	{
		printf("¥nNumber of devices is %d¥n¥n", numDevs);	// 台数を画面表示
	}
	else {
		// エラー処理
		return 0;
	}
	// シリアルナンバーを入手する
	for (i = 0; i < numDevs; i++) {
		memset(SerialNumber, 0, sizeof(SerialNumber));
		ftStatus = FT_GetDeviceInfoDetail(i, &Flags, &Type, &ID, &LocId, SerialNumber,
			Description, &ftHandle);
		printf("%d : %s¥n", i + 1, SerialNumber);	// シリアル番号を画面表示
	}
	return 0;
}
```

以上