赤外線リモコン受信機（ USB接続）

・いまさら赤外線リモコン？
・送信機
・受信機（ハードウェア）
・受信機（ファームウェア）

Goji(goji2100@gmail.com)

いまさら赤外線リモコン？

- **プレゼン用にレーザーポインタ付きのワイヤレスマウスが欲しい**
  - 結構高い
  - 手許にレーザーポインタ付きのリモコンマウスがあった
  - このリモコンは、赤外線で送り出ししているみたい
  - 普通のリモコン用の赤外線受光器で受信できそう
- **USBのお勉強のためにコードを書いてみよう！**
- **なにか役に立つの？**
  - スペシャルな入力デバイスを作るベースになるかも
  - Windows、Mac、Android端末などで使える

送信機

http://ibmpc.jp/item.cgi?item_id=31P6950&ctg_id=41NOPT&page=1

・マイクロポータブル・プロジェクターⅡ(IBM)

附属リモコン

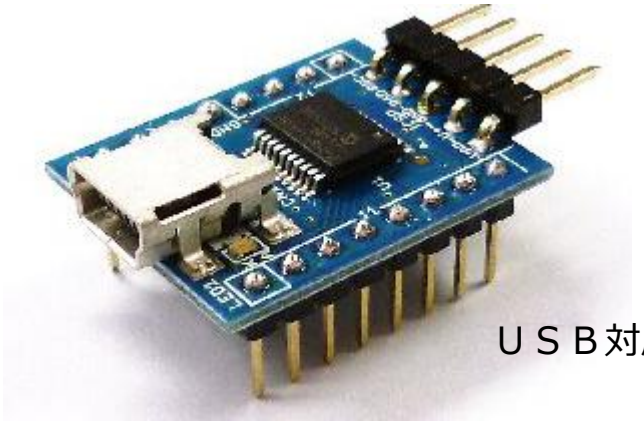
・レーザーポインタ付き

・トラックボール

・ボタン９個

受信機（ハードウェア）

赤外線リモコン受信モジュール
PL-IRM2121(38KHz)



ＵＳＢ対応超小型マイコンボード

http://akizukidenshi.com/download/ds/akizuki/AE-PIC18F14K50.pdf
http://akizukidenshi.com/catalog/g/gI-01570/

・PIC18F14K50使用 USB対応超小型マイコンボード
（秋月電子通商、¥800）

＋赤外線受光器
＋ファームウェア（USB Keyboard+Mouse Combo）

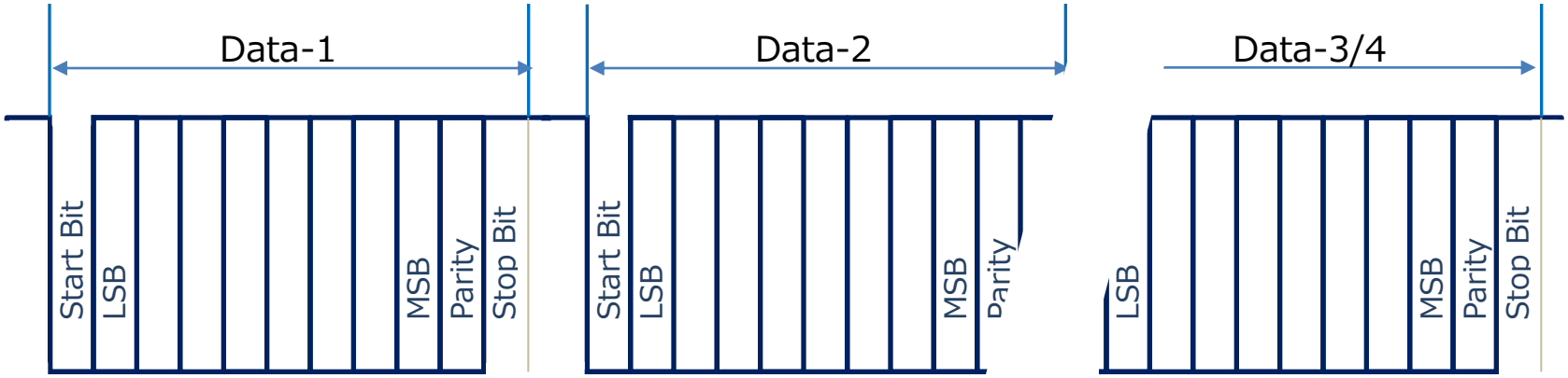
受信機（ファームウェア）

- **Microchip Solutionsには、デバイスの色々なサンプルコードが豊富に用意されている**
  - でも、KeyboardとMouseを複合デバイスとして使えるサンプルは無い
  - **後閑先生**の「**PICで楽しむUSB機器**」に「**複合インターフェースの構成の仕方**」の例がありますが、難しくて・・・
  - ネットで検索して、参考となるコードがありました
    この例は、MicrochipのサンプルのHID KeyboardとHID Mouseを組み合わせたもので、リモコンのベースに使えそうです
    このコードをベースに、IRで受信したデータをUSBパケットで送り出せば OK！

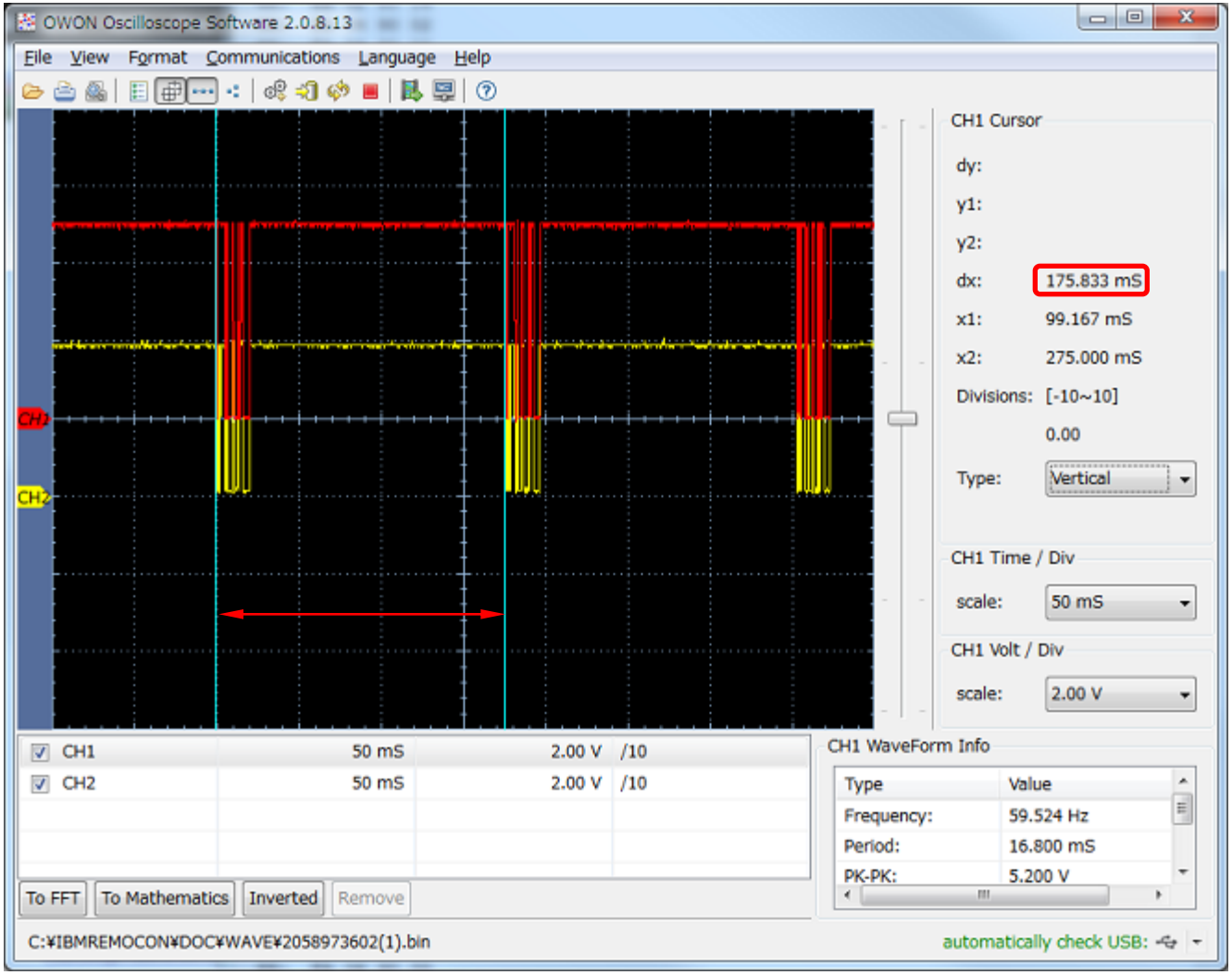
辻見裕史（北海道大学 電子科学研究所 電子材料物性部門 相転移物性研究分野 准教授）
Home：http://phys.sci.hokudai.ac.jp/LABS/yts/pic/4550/4550.html#hid_two
PIC：http://phys.sci.hokudai.ac.jp/LABS/yts/pic/pic.html
PIC18F4550
複合HIDデバイス (マウス+キーボード)
http://phys.sci.hokudai.ac.jp/LABS/yts/pic/4550/4550.html
4550_Mouse_Key_COM_YTS.zip

受信機（ファームウェア）- 赤外線受信コードのフォーマット

- **赤外線受信コードのフォーマットを確認**
  - 赤外線リモコンって、NECフォーマットか家電協フォーマットでしょ！
    ×どちらでもなかった・・
    波形を見ながら悩んだけれど、普通に
    調歩同期(2400bps、データ:8bit、偶数パリティ、Stop bit:1)らしい・・



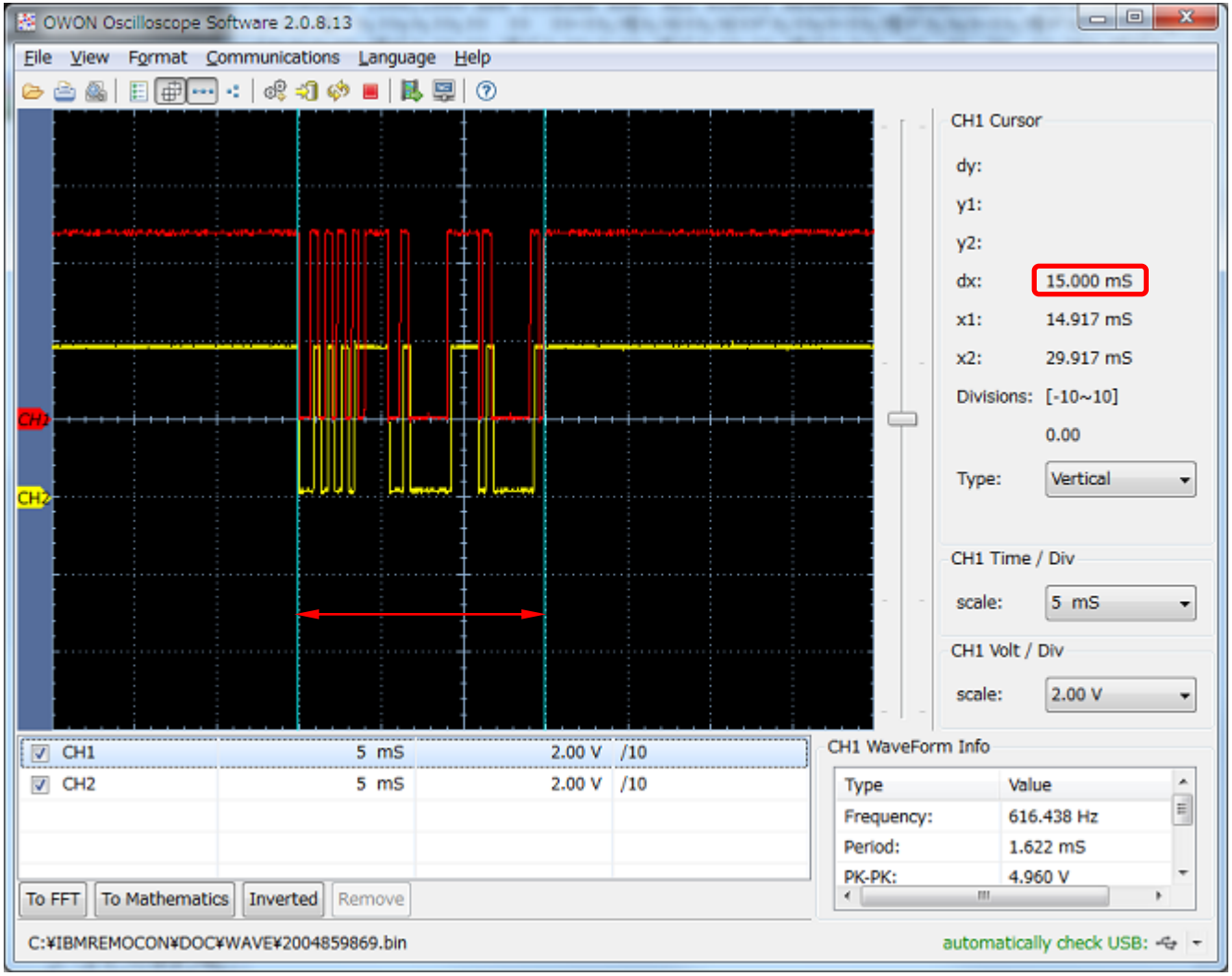
受信機（ファームウェア）- データ受信間隔



データの受信間隔

約175ms間隔で3バイトまたは4バイトのパケットを受信

受信機（ファームウェア） - 3バイトのパケット



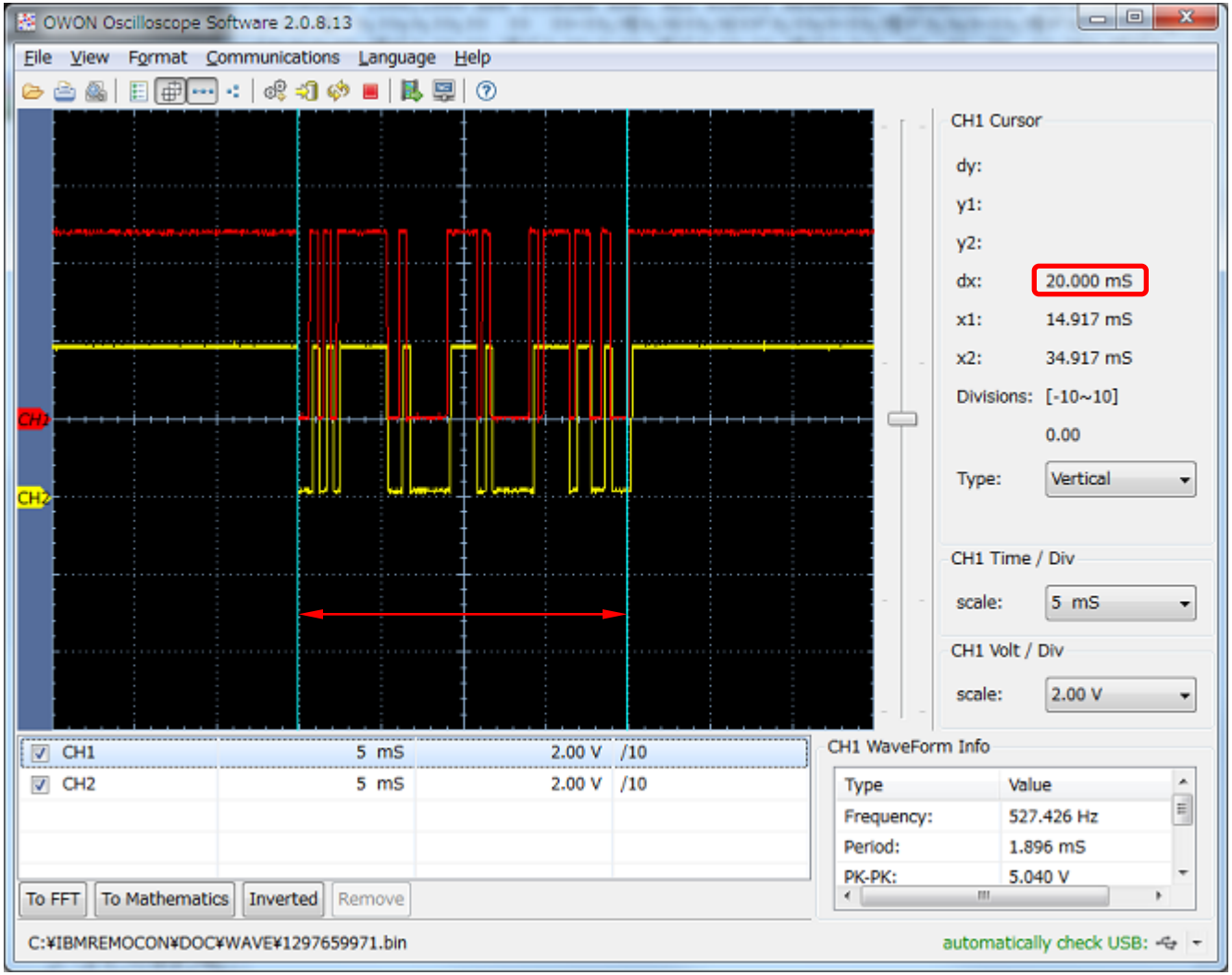
3データのデータ

約15msで3バイトのパケットを受信

1s/2400bps=416.6us/1b

Start+Data+Parity+Stop=
1+8+1+2=12bit
=>416.6×12=4.99ms

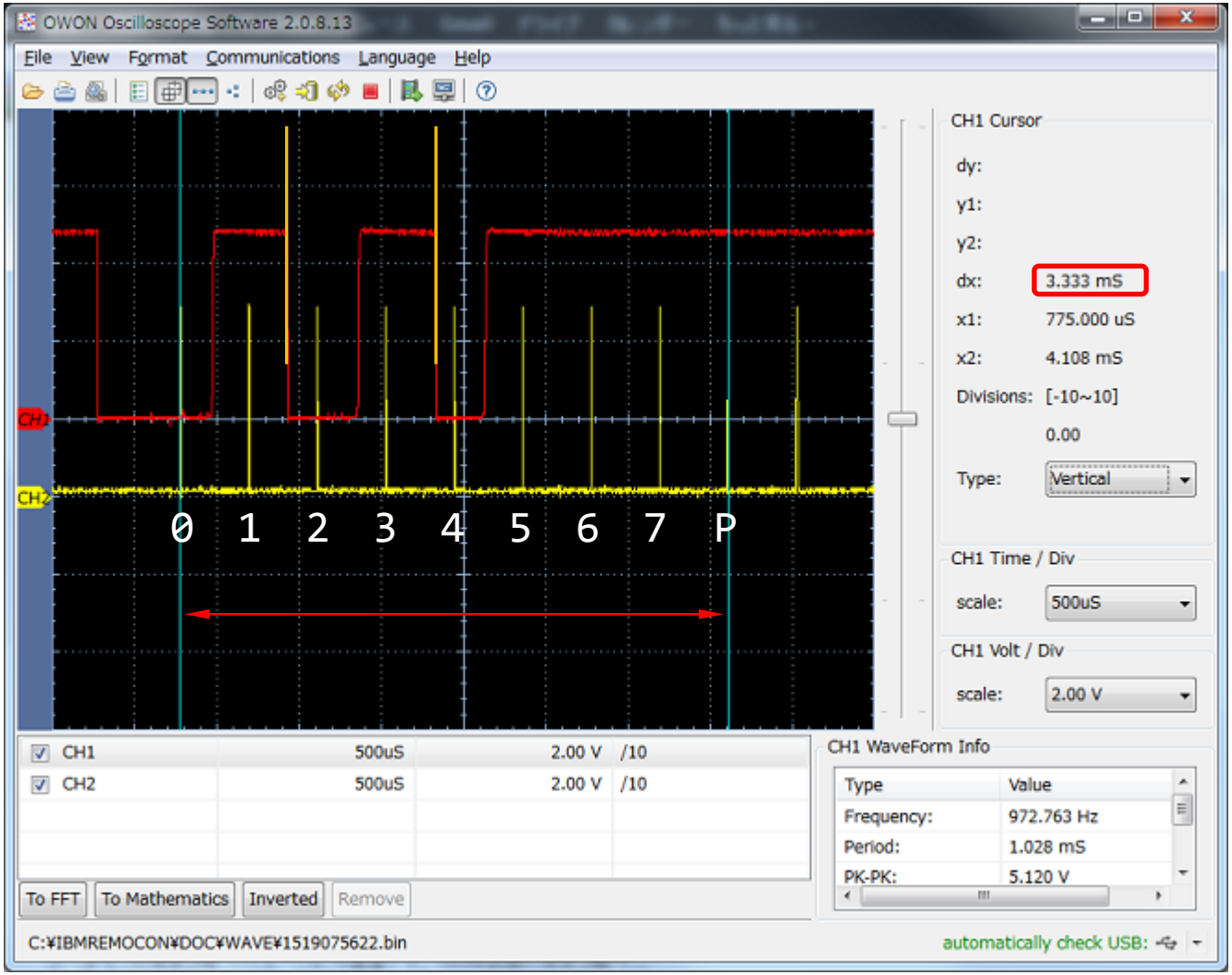
受信機（ファームウェア） - ４バイトのパケット



4データのデータ

約20msで4バイトのパケットを受信

受信機（ファームウェア）- 受信データサンプリング



サンプリングの様子-1

Ch1(赤)：
　受光器受信波形

Ch2(黄)：
　サンプリングポイント

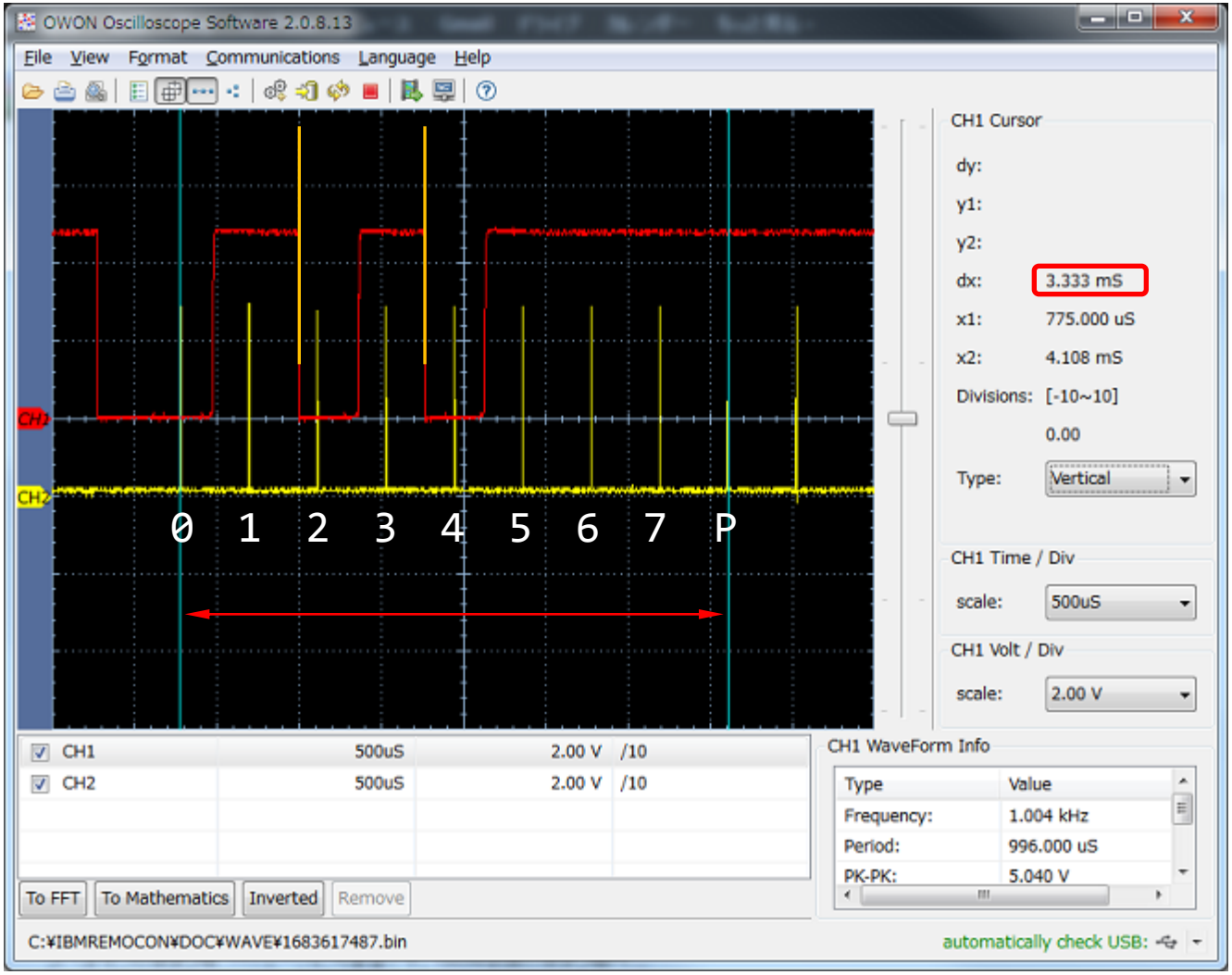
ボタン押下のデータ
=0b11101010
=0xEA

2400bps：
1,000,000/2400=**416.6us**

実測：
3333/8=**416.6us**

カウンター値：
Timer500=**375**
Timer400=**312**
(48MHz/4/16=1.333us)

受信機（ファームウェア）- 受信データサンプリング



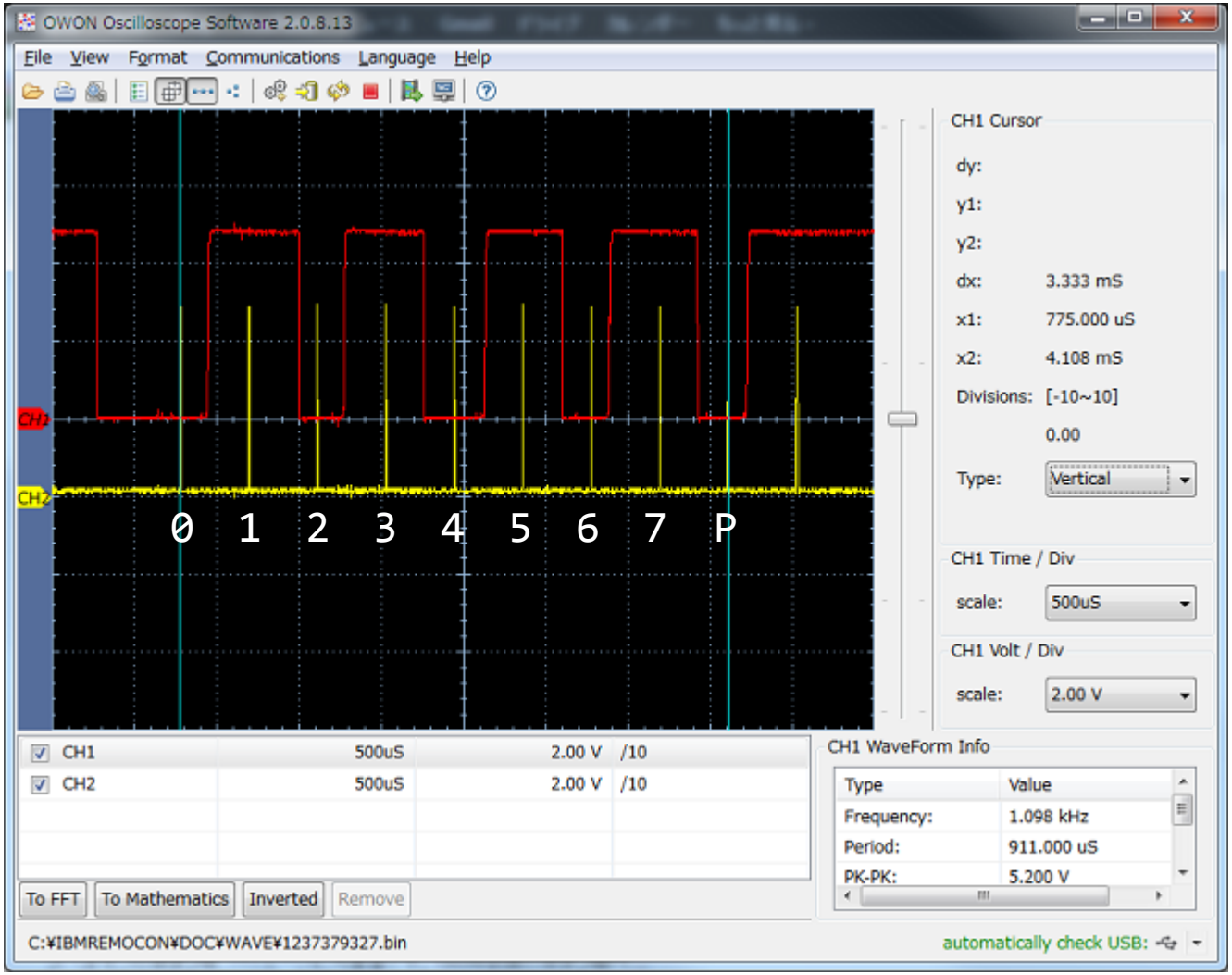
サンプリングの様子-2

Ch1(赤)：
　受光器受信波形

Ch2(黄)：
　サンプリングポイント

かなり、揺らぐ

受信機（ファームウェア）- 受信データサンプリング



サンプリングの様子-3

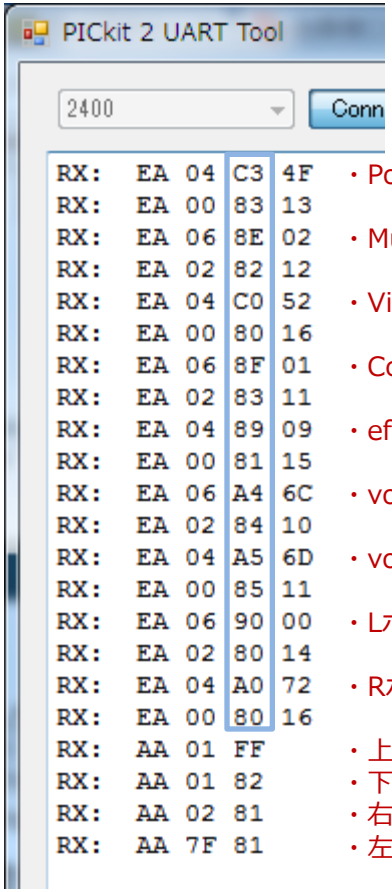
Ch1(赤)：
　受光器受信波形

Ch2(黄)：
　サンプリングポイント

トラックボールの移動データ
=0b10101010
=0xAA

受信機（ファームウェア）- リモコンからの受信データ

・**リモコンからの受信データ**



・Powerボタン

・Muneボタン

・Videoボタン

・Computerボタン

・effectボタン

・volume UPボタン

・volume DOWNボタン

・Lボタン

・Rボタン

・上に移動

・下に移動

・右に移動

・左に移動

受信機（ファームウェア）- 赤外線受信データ形式

・**赤外線受信データ**（リモコンからの受信データ）

・トラックボールとボタン

種別	Byte0	Byte1	Byte2	Byte3	内容
トラックボール	AA	X移動量	Y移動量	-	トラックボールの移動
ボタン	EA	0000 01xx	Key Code	XX	Make(押した)
		0000 00xx	1000 00xx	XX	Break(離した)

X移動量：0x01=移動無し、x0nn nnn0=正方向、x1nn nnn1=負方向
Y移動量：0x81=移動無し、x0nn nnn0=正方向、x1nn nnn1=負方向

Key Code(ボタン)

Code	Key	Code	Key	Code	Key
89	Effect	90	L	A5	Vol Down
8E	Menu	A0	R	C0	Video
8F	Computer	A4	Vol Up	C3	Power

受信機（ファームウェア）- USB送出データ

・**USB送出データ**(マウス:トラックボールとL/Rボタン)

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	BTN 8	BTN 7	BTN 6	BTN 5	BTN 4	BTN 3	BTN 2	BTN 1
1	X							
2	Y							

BTNn：0=押されていない、1=押されている、(n:8..1、Right=2 、Left=1)
X、Y：相対移動量

・**USB送出データ**(キーボード)

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	GUI	Alt	Shift	Ctrl	GUI	Alt	Shift	Ctrl
1	-							
2	Key data(1)							
..	..							
7	Key data(6)							

出典：Keyboard/Keypad Page (0x07) http://www.usb.org/developers/devclass_docs/Hut1_11.pdf

受信機（ファームウェア） - 処理概要

- **処理概要**
  - **受信処理**

    赤外線受信データをボタン、トラックボールデータとして認識

    ボタン：押した、離した、押し続けている

    ボール：上下、左右の相対移動量

  - **USB送信処理**

    トラックボールの移動情報＋L/Rボタン情報をマウスデータとして送出

    その他のボタンは、適当なキーに変換してキーボードデータとして送出

ボタンの割付け


レーザー
GUI
↑
ESC
↓
Enter
←
トラックボール
→
左クリック
右クリック
laser aperture
power
menu
laser
volume
video
on off
effect
computer
L
R
select

受信機（ファームウェア）- データ受信処理

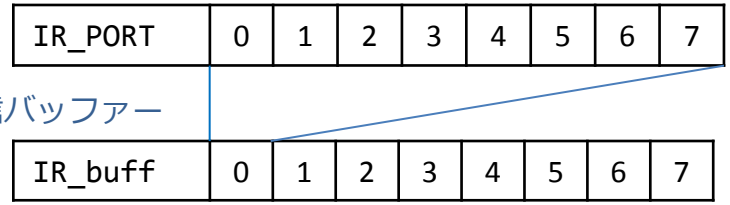
IR_PORTからのパルスデータを割込みで処理する

1) Start bit の立ち下がりで「状態変化割込み」発生
2) Start bit を読み捨てるために 1.5bit 分のインターバルタイマースタート
   「状態変化割込み」を無効、「タイマー割込み」を有効にする

3) インターバルタイマー割込みで、IR_PORTを読出し、バッファーに蓄積
4) 次のデータ待ちのために 1bit 分のインターバルタイマースタート

5) Parity bit まで、3)、4)を繰り返す
6) Parity bit を受信したら、パリティチェックを行う

7) Stop bit で 1 データの受信完了とし、ボタンデータ（0xEAで始まる4byte）、
   トラックボールデータ（0xAAで始まる 3 byte）の単位で送信処理用バッファに移す
   受信バッファーは、次のデータを受信できるようにクリアする
   「状態変化割込み」を有効に、「タイマー割込み」を無効にする　・・



受信機（ファームウェア）- 初期化

初期化

```
void UserInit(void)
{
    mInitAllLEDs();
    mLED_1_On();

    mInitTimer();                                   // タイマー初期化
    mSetTimer0();
    IR_TMR_IF = 0;
    IR_TMR_IE = 0;

    mInitIR();                                      // 赤外線入力初期化
    IR_Dummy = IRR_PORT;
    IR_IOC_IF = 0;
    IR_IOC_IE = 1;

    IR_didx = 0;                                    // 受信バッファ初期化
    IR_buff[0] = IR_data[0] = 0;

    INTCONbits.GIEH = 1;                            // 割込み許可
    INTCONbits.GIEL = 1;

    //Initialize all of the mouse data to 0,0,0 (no movement)
    buffer[0]=buffer[1]=buffer[2]=0;

    //initialize the variable holding the handle for the last transmission
    lastTransmission2 =
    lastTransmission1 = 0;
}
```

受信機（ファームウェア）- 割込みルーチン(1)

割込みルーチン(1)

```
//These are your actual interrupt handling routines.
#pragma interrupt YourHighPriorityISRCode
void YourHighPriorityISRCode()
{
    #if defined(USB_INTERRUPT)
        USBDeviceTasks();
    #endif

    if (IR_IOC_IE == 1 && IR_IOC_IF == 1) {             // 状態変化割込み
        mLED_1_Off();
        IR_bpos = 0;
        IR_stat = 0;
        IR_bitc = 0;
        IR_bmsk = 0x01;
        IR_buff[IR_didx] = 0;
        mSetTimer500();                                 // スタートビットをスキップ
        IR_TMR_IF = 0;
        IR_TMR_IE = 1;                                  // タイマー割込み有効
        mSetTimerON();                                  // タイマースタート
        IR_Dummy = IRR_PORT;
        IR_IOC_IF = 0;                                  // 状態割込みフラグクリア
        IR_IOC_IE = 0;                                  // 状態変化割込み禁止
    }
```

受信機（ファームウェア）- 割込みルーチン(2)

割込みルーチン(2)

```
if (IR_TMR_IE == 1 && IR_TMR_IF == 1) {          // タイマー割込み
   mLED_2_On();
   mSetTimer400();                               // 次のタイマー割込みまでの時間セット

   if (IR_bpos < 8) {                            // データビット処理（MSBからLSBまで）
       if (IRR_PORT) {
           mLED_1_On();
           IR_buff[IR_didx] |= IR_bmsk;
           IR_bitc++;
       } else
           mLED_1_Off();
       IR_bmsk <<= 1;
   }

   if (IR_bpos == 8) {                           // パリティビットの処理
       mLED_1_On();
       if ((IRR_PORT) != (IR_bitc & 0x01))       // パリティチェック
           IR_stat |= IR_ERR_PARITY;
   }
```

受信機（ファームウェア）- 割込みルーチン(3)

割込みルーチン(3)

```
            if (IR_bpos > 8) {                              // ストップビット処理
                mLED_1_On();
                if (IRR_PORT) {
                    if ((IR_buff[0] == 0xAA) || (IR_buff[0] == 0xEA)) { // パケット受信完了
                        if (IR_didx < (IRR_DATAL - 1)) IR_didx++;
                        if (((IR_buff[0] == 0xAA) && (IR_didx >= 3)) || // トラックボールデータ(3byte)
                            ((IR_buff[0] == 0xEA) && (IR_didx >= 4))) { // ボタンデータ(4byte)
                            if (IR_data[0] == 0x00) {
                                IR_data[3] = IR_buff[3];    // データ退避
                                IR_data[2] = IR_buff[2];
                                IR_data[1] = IR_buff[1];
                                IR_data[0] = IR_buff[0];
                            }
                            IR_didx = 0;
                        }
                    }
                    IR_Dummy = IRR_PORT;
                    IR_IOC_IF = 0;
                    IR_IOC_IE = 1;                          // 状態変化割込み有効
                    IR_TMR_IE = 0;                          // タイマー割込み禁止
                    mSetTimer0();
                } else {
                    IR_stat |= IR_ERR_FRAME;                // ストップビットでない（フレーミングエラー）
                }
            }
            IR_bpos++;
            IR_TMR_IF = 0;                                  // タイマー割込みフラグクリア
            mLED_2_Off();
        }
    }
```

受信機（ファームウェア）- USBデータ処理

USBデータ処理

```
void ProcessIO(void)
{
    // User Application USB tasks
    if ((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1)) return;

    // Call the function if received IR signal
    if (IR_data[0]) {
        Emulate_Mouse();
        Emulate_Keyboard();
    }
}
```

受信機（ファームウェア）- マウス処理

マウス処理

```
void Emulate_Mouse(void)
{
    if (IR_data[0] == 0xAA) {                                   // トラックボール移動データ処理
        if (IR_data[1]!=0x01) buffer[1]=(IR_data[1]&0x40)?(IR_data[1]-1|0xE0):(IR_data[1]&0x1F);
        if (IR_data[2]!=0x81) buffer[2]=(IR_data[2]&0x40)?(IR_data[2]-1|0xE0):(IR_data[2]&0x1F);
        IR_data[0] = 0;                                         // 次のデータ受入れ可
    }

    if (IR_data[0] == 0xEA) {                                   // L/Rキーの処理
        buffer[0] = 0;
        if (IR_data[1] & 0x04) {                                // Make Key(押した)
            if (IR_data[2] == 0x90) {                           // L Button
                buffer[0] |= 0x01;
                IR_data[0] = 0;
            }
            if (IR_data[2] == 0xA0) {                           // R Button
                buffer[0] |= 0x02;
                IR_data[0] = 0;
            }
        }
    }

    if (!HIDTxHandleBusy(lastTransmission1)) {                  // USB送信可？
        hid_report_in1[0] = buffer[0];
        hid_report_in1[1] = buffer[1];
        hid_report_in1[2] = buffer[2];
        lastTransmission1 = HIDTxPacket(HID_EP1, (BYTE*)hid_report_in1, 3);
        buffer[1] = buffer[2] = 0;                              // 移動データクリア
    }
}
```

受信機（ファームウェア）- キーボード処理

キーボード処理

```
void Emulate_Keyboard(void)
{
    if (!HIDTxHandleBusy(lastTransmission2)) {
        memset((void *)hid_report_in2, 0x00, sizeof(hid_report_in2));
        if (IR_data[0] == 0xEA) {
            if (IR_data[1] & 0x04) {              // Make Key
                switch (IR_data[2]) {
                  case 0x89:                      // Effect
                    hid_report_in2[2] = 0x50;     // Left Arrow
                    break;
                  // 同様の処理省略
                  case 0x8E: ⇒ 0x29:ESC           // Menu
                  case 0x8F: ⇒ 0x4F:Left Arrow    // Computer
                  case 0xA4: ⇒ 0x52:Up Arrow      // Volume Up
                  case 0xA5: ⇒ 0x51:Down Arrow    // Volume Down
                  case 0xC0: ⇒ 0x28:Enter         // Video
                  // 省略ここまで
                  case 0xC3:                      // Power
                    hid_report_in2[0] = 0x08;     // GUI
                    break;
                  default:
                    break;
                }
            }
            IR_data[0] = 0;
        }
        lastTransmission2 = HIDTxPacket(HID_EP2, (BYTE*)hid_report_in2, 0x08);
    }
}
```

最後に・・

・Amazonで安いリモコン付きレーザーポインターが売っていました・・・

リモコン付レーザーポインター SP-101 ￥ 1,740