

USB「空中マウス」の製作

茂木光多

本誌 2008 年 5 月号付属 ARM 基板で「空中マウス」を作る。STM32 の 500 ページを超えるマニュアルは脇に置いて、周辺ハードウェアのアクセスは STMicroelectronics 社のファームウェア FWlib を使用する。USB 制御を含むマウスの骨格は ARM 用 IAR Embedded Workbench - コード・サイズ限定版 (以下 EWARM と略す。2008 年 5 月号本誌付属 CD-ROM に収録。スウェーデン IAR Systems 社の Web サイトよりダウンロード可能) に付属のサンプル・プログラムを改造し、STMicroelectronics 社の 3 軸加速度センサ LIS344 ALH を使うことに集中する。 (筆者)

1. STM32 マイコンを使う

STM32 は豊富な機能を持ったマイクロコントローラです。パラレル・ポートに接続した LED を点灯させる作業一つをとってみても、ポート自身とその方向レジスタを探し出せば何とかなった以前とは大違いです。マニュアルをなんとか読み下して初期化プログラムを書いたとしても、本当に正しいのかどうか不安になると思います。

半導体メーカーも、年々複雑になるマイコンをユーザーに使ってもらうために、いろいろ知恵を絞っています。評価ボードやサンプル・プログラムを提供することはずいぶん前から一般的になっていますが、公開されているプログラムを Web サイトからダウンロードし、理解し、自分自身のターゲットに移植し、自分の開発環境に合わせてプロジェクトを作成して動かすには、手の早い人でも 2、3 日仕事になります。

このような現実の中で、STMicroelectronics 社の取り組みは独特です。開発ツール・メーカーと協力し、自社の評価ボード用のサンプル・プログラムだけでなく、すべての内部のハードウェアの初期化、制御関数をツールに付属する形で供給していることです。このことにより、ユーザはプロジェクトの作成や移植作業を行うことなくすぐに動作確認を行えます。また、上で述べた「ほんとにこれで良いのか」の疑問から解放されます。

リスト 1 に GPIO の初期化関数の最初の部分を示します。コメントや空行を含めて 100 行を超えていて、いわゆるパラレル・ポートの初期化としては最大級のプログラムといえます。やはり STMicroelectronics 社の敷いたレールに乗り、このまま使用するのが簡単で確実です。

2. 加速度センサと「空中マウス」の仕様

今回製作する「空中マウス」では、STMicroelectronics 社の 3 軸加速度センサ LIS344ALH を単なる姿勢センサとして使用します。

加速度センサの出力は、図 1 のように USB コネクタを前にして持ち、左に傾けると X 出力電圧が増加し垂直になったところで最大、右に傾けると減少し垂直になると電圧が最小になります。さらにケーブルが上になるように傾けると Y 出力電圧が増加し垂直になったところで最大、下に傾けると減少し垂直になると電圧が最小になります。Z 出力は上記の傾ける方向とは無関係に、基板の実装面が上を向いていて最大、下を向いて最小になります。

Keyword

STM32, STMicroelectronics, GPIO の初期化, LIS344ALH, EWARM, 空中マウス, GPIO_int 関数, DfuSe, リンカ・コマンド・ファイル, HID 準拠マウス, 加速度センサ

リスト1

GPIO の初期化関数 (抜粋)

```

/*****
 * Function Name : GPIO_Init
 * Description   : Initializes the GPIOx peripheral according to the specified
 *                 parameters in the GPIO_InitStruct.
 * Input        : - GPIOx: where x can be (A..E) to select the GPIO peripheral.
 *                 - GPIO_InitStruct: pointer to a GPIO_InitTypeDef structure that
 *                 contains the configuration information for the specified GPIO
 *                 peripheral.
 * Output       : None
 * Return       : None
 *****/
void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct)
{
    u32 currentmode = 0x00, currentpin = 0x00, pinpos = 0x00, pos = 0x00;
    u32 tmpreg = 0x00, pinmask = 0x00;

    /* Check the parameters */
    assert_param(IS_GPIO_MODE(GPIO_InitStruct->GPIO_Mode));
    assert_param(IS_GPIO_PIN(GPIO_InitStruct->GPIO_Pin));

    /*----- GPIO Mode Configuration -----*/
    currentmode = ((u32)GPIO_InitStruct->GPIO_Mode) & ((u32)0x0F);

    if (((u32)GPIO_InitStruct->GPIO_Mode) & ((u32)0x10)) != 0x00)
    {
        /* Check the parameters */
        assert_param(IS_GPIO_SPEED(GPIO_InitStruct->GPIO_Speed));
        /* Output mode */
        currentmode |= (u32)GPIO_InitStruct->GPIO_Speed;
    }

    /*----- GPIO CRL Configuration -----*/
    /* Configure the eight low port pins */
    if (((u32)GPIO_InitStruct->GPIO_Pin & ((u32)0x00FF)) != 0x00)
    {
        tmpreg = GPIOx->CRL;

        for (pinpos = 0x00; pinpos < 0x08; pinpos++)
        {
            pos = ((u32)0x01) << pinpos;
            /* Get the port pins position */
            currentpin = (GPIO_InitStruct->GPIO_Pin) & pos;

            if (currentpin == pos)
            {
                pos = pinpos << 2;
                /* Clear the corresponding low control register bits */
                pinmask = ((u32)0x0F) << pos;
                tmpreg &= ~pinmask;

                /* Write the mode configuration in the corresponding bits */
                tmpreg |= (currentmode << pos);

                /* Reset the corresponding ODR bit */
                if (GPIO_InitStruct->GPIO_Mode == GPIO_Mode_IPD)
                {
                    GPIOx->BRR = (((u32)0x01) << pinpos);
                }
                /* Set the corresponding ODR bit */
                if (GPIO_InitStruct->GPIO_Mode == GPIO_Mode_IPU)
                {
                    GPIOx->BSRR = (((u32)0x01) << pinpos);
                }
            }
        }
        GPIOx->CRL = tmpreg;
        tmpreg = 0;
    }
}

```

以下略

以上のことを踏まえてマウスの仕様を決めます。通常のマウスに比べて操作性は劣りますが、マウスとしての最低限のことはできるようにしました。

[移動中]

水平 : 停止

左に傾ける : マウス・カーソルが左に移動

右に傾ける : マウス・カーソルが右に移動

前に傾ける : マウス・カーソルが上に移動