

Programação de um PIC por USB sem programador

Introdução

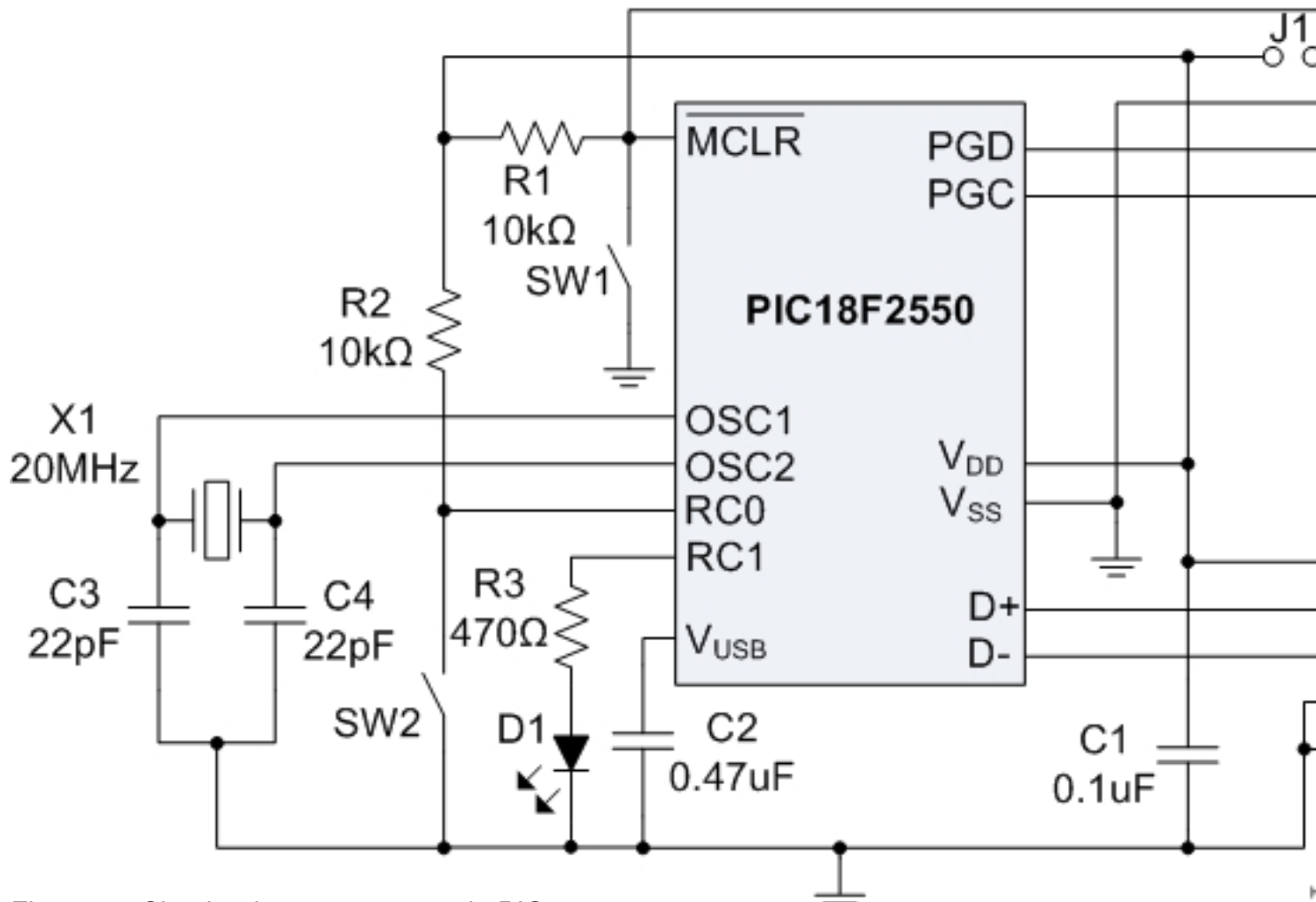
Tipicamente os micro-controladores PIC são programados com recurso a programadores (PICkit, ICD, etc.). A possibilidade de programar os mesmos recorrendo a uma ligação directa pela porta USB não só é interessante em termos de custos, como em termos da liberdade que nos dá. No domínio educativo, se considerarmos dar a possibilidade a todos os alunos de programar um PIC, independentemente do local onde o fazem, esta ideia ganha ainda mais significado.

Nesta página encontra os passos necessários para poder programar um PIC sem ter de recorrer a um programador. Note-se que numa primeira fase este será ainda necessário.

A ideia base será a instalação de um *bootloader* USB do tipo *Human Interface Device* (HID). O circuito utilizado possui um botão que, quando premido, permite ao PIC arrancar o *bootlader*, ficando assim disponível para carregar uma nova aplicação (*firmware*). Caso o botão não seja pressionado durante o arranque, o PIC irá arrancar com o programa actualmente carregado em memória. Este programa tanto poderá ser um programa que utilize a porta USB como um qualquer outro programa.

O Circuito

Na Figura 1 encontra o circuito utilizado para o efeito. Note-se que o circuito permite tanto a programação/utilização da porta USB, como a programação utilizando a funcionalidade ICSP (com recurso a um programador).



Para mais informações sobre a configuração do PIC18F2550 para a programação por USB, consulte o [PICKit 2 Programmer/Debugger User's Guide](#), capítulo 14, seção 14.1.1.

Bootloader

O *bootloader* aqui utilizado é baseado no fornecido na [Microchip Applications Libraries \(MAL\)](#), nomeadamente no exemplo

Microchip SolutionsUSB Device - BootloadersHID - BootloaderHID Bootloader - Firmware for PIC18 Non-J Devices

da biblioteca USB versão 2.8. Nota, caso queira utilizar um dos *bootloaders*

fornecidos previamente compilados, não necessita de descarregar a [MAL](#)

O princípio de funcionamento consiste em testar uma determinada entrada no arranque. Caso esta esteja activa continua o arranque do *bootloader*, caso contrário arranca a aplicação do

utilizador. Para tal, quer durante o desenvolvimento do bootloader, quer no desenvolvimento da aplicação a correr no espaço do utilizador, é crucial a definição das zonas de memória a utilizar. Tal é conseguido através da utilização de um ficheiro

LNK

específico e algumas instruções extra no programa.

Relativamente ao *bootloader* exemplo fornecido pela Microchip, a principal diferença é a remoção e simplificação de algum código relativamente aos LEDs utilizados, bem como alteração de alguns portos. De acordo com o circuito anterior, o porto utilizado para o botão de arranque do *bootloader* foi o RC0, e o porto utilizado para sinalizar o funcionamento do *bootloader* foi o RC1.

O *bootlader* foi desenvolvido e compilado no MPLAB com o MCC18 e configuração de optimização máxima. Caso pretenda compilar o *bootloader* e não utilize a opção de optimização máxima (só disponível na versão PRO, ou durante 60 dias) ocorrerá um erro a compilar, dado que o espaço disponível para o *bootloader* não será suficiente. A opção nesse caso será utilizar a optimização máxima ou aumentar o espaço reservado para o *bootloader*.

Ficheiros importantes (considerando o circuito da Figura 1):

- [HID Bootloader PIC18 Non J.zip](#) – Projecto MPLAB do bootloader;
- [HID Bootloader PIC18F2550_20MHz.hex](#) – *bootloader* compilado para o PIC18F2550 e cristal de 20MHz;
- [HID Bootloader PIC18F4550_20MHz.he](#) x – *bootloader* compilado para o PIC18F4550 e cristal de 20MHz (*não testado*);
- [Bootloader LNK](#) - ficheiros LNK para gerar o *bootloader* para diferentes PICs (*apenas foi testado o ficheiro para o PIC18F2550*).

Para carregar o *bootloader* deverá descarregar o ficheiro HEX, montar o circuito da Figura 1,

ligar um programador ao conector ICSP e programar o mesmo. Poderá utilizar o MPLAB ou a aplicação específica do programador.

Para testar pressione o botão SW2 durante a inicialização do PIC (por exemplo, pressionando SW1). Neste caso o LED D1 deverá acender. Caso o circuito esteja ligado ao PC pela porta USB, este irá reconhecer o dispositivo (não necessita de *driver*) e irá exibir uma mensagem indicando a ligação de um dispositivo HID. Nesta altura o LED irá piscar, indicando que está pronto a receber a aplicação.

Aplicação do Utilizador

Para que a aplicação possa ser programada no PIC juntamente com o *bootloader* HID, é necessário incluir um ficheiro

LNK

específico, bem como algumas instruções no código principal. Desta forma garante-se que a aplicação é armazenada numa zona de memória após a zona reservada para o *bootloader*

A título de exemplo consulte o projecto de teste fornecido. Este considera que, tal como no porto RC1, existem LEDs ligados aos portos RB0, RB1, RB2, RB3 e RB6, e utiliza o botão ligado ao porto RC0. Como poderá confirmar, aplicação em causa limita-se a activar alternativamente os portos RB0/RB2 e RB1/RB3, activando o porto RB6 sempre que o botão SW2 estiver pressionado.

Depois de compilada, a aplicação pode ser carregada para o PIC através da ligação USB. Para tal ligue o circuito ao PC e pressione o botão SW2 para arrancar o *bootloader*. O LED D1 deverá piscar indicando que o

bootloader

está a correr. De seguida deverá arrancar a aplicação HIDBootLoader.exe (também incluída na

[MAL](#)

), seleccionar a opção

Open HEX File

e, finalmente, a opção

Program/Verify

. Nesta altura poderá reiniciar o PIC, devendo este correr a aplicação agora programada.

Ficheiros importantes (considerando o circuito da Figura 1):

- [App_test.zip](#) – Projecto MPLAB da aplicação de teste;
- [App_test.hex](#) – aplicação de teste compilada;
- [HIDBootLoader.exe](#) – aplicação para enviar o programa do utilizador para o PIC utilizando o *bootlodoader HID*;
- [App LNK](#) - ficheiros LNK para gerar a aplicação para diferentes PICs (*apenas foi testado o ficheiro para o PIC18F2550*).

Referências

Foram utilizados os seguintes documentos para a implementação da solução aqui estudada. Caso queira aprofundar o seu conhecimento, aconselha-se a leitura dos mesmos:

- [Getting Started: Using the “USB Device – Bootloaders”](#) (considerando que a [MAL](#) foi instalada na pasta *C:/Microchip Solutions*) – este document e os restantes incluídos na instalação da [MAL](#), em particular relativamente ao funcionamento USB, estão bastante completos e incluem muitos exemplos;
- [PICkit 2 Programmer/Debugger User's Guide](#) - informação importante na utilização do PICkit 2 e ICSP;
- [USB Bootloader for PIC18F2550 Part 1](#) e [USB Bootloader for PIC18F2550 Part 2](#) – exemplo que inclui alteração do espaço de memória reservado ao *bootloader*, e outras alterações, por forma a permitir gerar o mesmo com a versão gratuita do MCC18;
- [USB Bootloader](#) – ideia semelhante mas não utiliza o modo HID, pelo que requer um driver específico (também fornecido na [MAL](#));
- [USB Bit Whacker](#) – placa e firmware que permitem utilizar o PIC interactivamente, activando e consultando o estado dos portos por USB. Inclui ainda ligações para vários outros projectos;
- [PIC USB Framework](#) – projecto semelhante ao ilustrado aqui, mas virado sobretudo para a utilização em Linux;
- [USB PIC Bootloader](#) – versão comercial idêntica à ilustrada aqui mas mais completa;
- [BEEPIC](#) – versão comercial desta ideia, idêntica ao conceito do ARDUINO (embora não

recorrendo a bibliotecas específicas).

TODO

O próximo passo na actualização deste documento será incluir a comunicação série através do PIC. Para tal existem alguns exemplos na [MAL](#) sobre a programação do PIC como emulador da porta série.

FAQ

Tenho um PIC diferente e queria utilizar o bootloader. O que faço?

Primeiro deverá confirmar que o PIC em questão possui funcionalidade USB (poderá confirmar verificando a existência dos portos D+/D-). O primeiro passo é descarregar o ficheiro HID Bootloader PIC18 Non J.zip e abrir o projecto no MPLAB. De seguida deverá alterar o dispositivo em *Configure à Select Device* e verificar no ficheiro *main.c* se a configuração utilizada é adequada ao PIC a utilizar e ao cristal/relógio escolhido.

Caso queira alterar o porto onde está ligado o LED ou o botão SW2, deverá editar o ficheiro *io_cfg.h*

.

Deverá compilar o projecto com a optimização máximo (compilador C18 PRO), caso contrário terá problemas de espaço para o *bootloader*. A alternativa será aumentar o espaço reservado para o *bootloader*.