**Autores : Gabriel Silva Marcatto e Rodrigo Maximiano Antunes de Almeida**

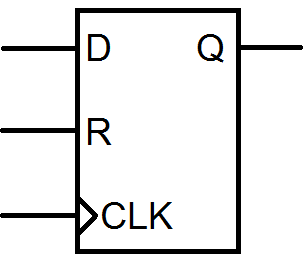
Um primeiro problema com essa abordagem, em conjunto da placa Freedom KL05z, é a falta de terminais suficientes para acionar todos os periféricos desejados. Para contornar esse problema é comum que os desenvolvedores procurem chips de expansão de I/O`s. No nosso caso vamos trabalhar com o 74hc595, que é um registrador de deslocamento (Shift Registers), ou seja, um conversor serial-paralelo. Com ele é possível controlar até 8 saídas de uma só vez com apenas 3 terminais. Para isso vamos focar no artigo de hoje em como este circuito funciona e como desenvolver um código que possa realizar a interface entre o microcontrolador e o expansor de saídas 74hc595.

Antes de entendermos o funcionamento do 74hc595, temos que entender o funcionamento de um FLIP-FLOP tipo D e de um deslocador de registro.

**FLIP-FLOP D**

Os FLIP-FLOPs são dispositivos básicos de memorização, ou seja, são basicamente memórias primitivas.

A seguir é apresentada a representação de um flip flop D genérico:

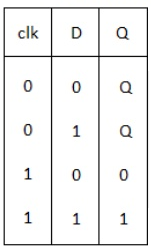


(D - dado, R - Reset, CLK- Clock, Q - Saída)

O seu funcionamento pode ser explicado do seguinte modo:

* Quando o **CLK** estiver em **nível alto**, a entrada **D**é reproduzida na saída **Q;**
* Quando o **CLK**estiver em **nível baixo,** a saída **Q** permanece com o **nível anterior;**
* Para **resetar**o valor de **Q** é necessário ter **nível alto** no **R** e **CLK**.

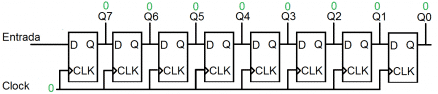
Se montarmos a tabela verdade desse dispositivo temos algo como:



**Deslocador de registro**

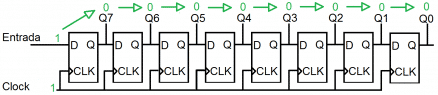
Os *deslocadores de registro* ou *shift registers* possuem diversos Flip Flops D em série, que permitem a conversão do dado enviado em forma serial para a forma paralela. A conversão se realiza por deslocamentos consecutivos.

Supondo uma série de Flip Flops D genéricos e que nossa **entrada** (dados) seja de 8 bits. Como início temos que as saídas **Q7~Q0** estão flutuando, por conveniência, vamos supor que **Q7~Q0** sejam **0**.  Logo temos:

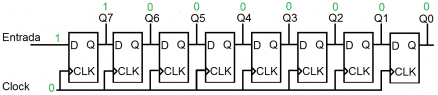


Para que **Q7** passe a ter o valor de **D**, é necessário que ocorra um pulso **clock**, porém como todos estão em série, ao dar o clock, ocorre uma reação em cadeia, ou seja, **Q6**passa ter o valor de **Q7**, **Q5** o de **Q6** e assim por diante.

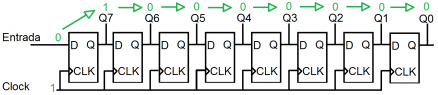
Para facilitar o entendimento, faremos um exemplo com uma entrada 10010101. Quando é enviado o **Bit 1**  e o **clock** passa a ser **1**, o circuito se comporta da seguinte maneira:



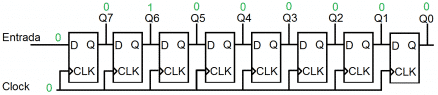
Quando o sinal do **clock** volta ser**0** , ou seja, quando o pulso de clock é finalizado, ocorre o descolamento das saídas:



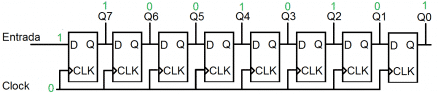
O mesmo ocorre quando o**Bit 0** é enviado e o **clock** passa a ser **1**:



Quando o pulso é finalizado, temos:

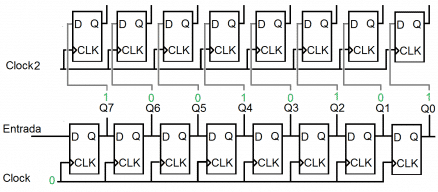


Para o valor de informação desejado, 10010101, o estado do registrador, após enviarmos todos os bits, é dado por:

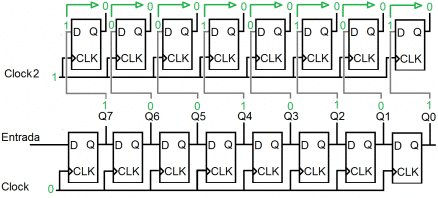


Pode se observar que a os bits foram recebidos de modo serial e estão agora apresentados de modo paralelo, completando a conversão.

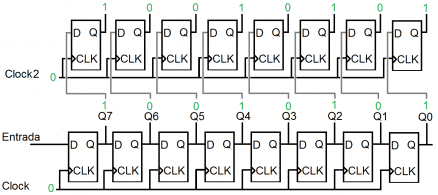
No 74HC595, para não se perder os dados antigos, quando novos dados estão sendo convertidos, as saídas dos flip flops são conectas a outros Flip Flops, o que permite a memorização dos dados na saída.



Para gravar os dados e permitir que saída seja os dados de entrada, basta dar um pulso de clock (Clock2) na primeira fileira.



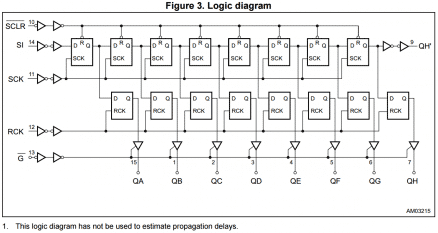
Quando o pulso é finalizamos vamos ter:



Quando novos dados chegam, a conversão ocorre na segunda fileira, porém como clock2 permanece 0, a saída se manterá a mesma. Quando a conversão é finalizada e se deseja jogar os dados da conversão para a saída, bastar dar um pulso de clock em clock2.

**O 74HC595**

O 74hc595 funciona de modo muito parecido com o apresentado até agora. Ele possui apenas algumas alterações, visando ampliar um pouco os recursos disponíveis. A seguir é apresentado o esquemático fornecido pela ST em seu datasheet.

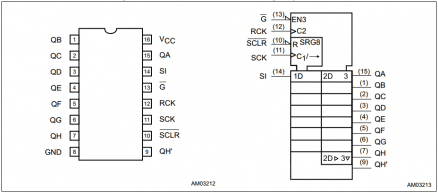


Entre as diferenças do circuito da ST e do apresentado até agora, podemos perceber um sinal (G) que desconecta ou conecta a saída em modo tristate, possibilitando que o desenvolvedor coloque o 74hc595 em paralelo com outros dispositivos de saída num mesmo barramento, sob condições pré definidas, sem que haja curto circuito.

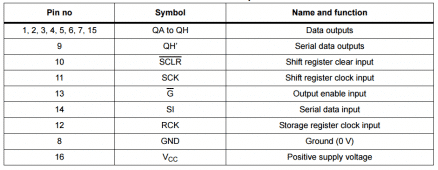
Para  economizarmos os  pinos da placa, e simplificar a utilização do shift register, vamos sempre deixar o sinal (G) em 1 e o reset no 0, já que as chaves possuem lógica invertida.

As entradas SI, SRK e SCK não necessitam de pinos especiais, ou seja, pode se escolher em qual terminal de IO se deseja por essas entradas.

Abaixo segue a imagem e footprint dos terminais do 74hc595.

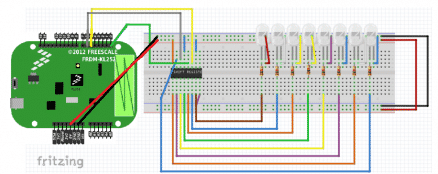


(Conexões dos terminais do 74hc595)



(Descrição dos pinos do 74hc595)

Com estas referências podemos montar o seguinte circuito:



Foi utilizado o software Fritzing para fazer a representação gráfica da ligação. Por escolha própria a entrada de dados (Si) foi colocada no pino 6, o RCK no pino 0 e o SCK no pino 5. Mas vale relembrar que eles podem ser conectados em qualquer pino.

Após realizarmos as conexões de hardware podemos começar a criação da biblioteca responsável pelo controle do CI.

É necessário realizar o controle apenas dos sinais Si, SCK e RCK. Como explicado, na biblioteca responsável por esse chip, deve possuir as seguintes funcionalidades:

* Configurar os pinos Si, SRC e RCK como output;
* Enviar um bit;
* Dar pulso de clock em SCK para cada bit enviado e;
* Após enviar os 8 bits, dar um pulso de clock em RCK.

Para simplificar o desenvolvimento dessa biblioteca vamos utilizar a biblioteca GPIO, desenvolvida no artigo passado. Assim o código da biblioteca responsável pela inicialização e envio de dados do 74hc595 pode ser dada por:

#include "io.h"

#define SCK  5

#define Si  6

#define RCK  0

void soInit(void) {

pinMode(RCK,  OUTPUT);

pinMode(clockpin, OUTPUT);

pinMode(datapin, OUTPUT);

}

void shift\_out(int value) {

    int x;

    digitalWrite(clockpin, LOW);

    digitalWrite(RCK , LOW);

    for (x = 0; x < 8; x++) {

        if (value & 0x80) {

            digitalWrite(Si, HIGH);

        } else {

            digitalWrite(Si, LOW);

        }

        digitalWrite(SCK, HIGH);

        digitalWrite(SCK, LOW);

        value = (value << 1);

    }

    digitalWrite(RCK , HIGH);

    digitalWrite(RCK , LOW);

}

#ifndef SO\_H\_

#define SO\_H\_

void soInit (void);

void shift\_out(int value);

#endif /\* SO\_H\_ \*/

É importante lembrar que o envio dos bits podem ser realizado do mais para o menos significativo ou do menos para o mais significativo, isso vai depender de como o hardware foi ligado.

Abaixo segue um exemplo de como realizar o acionamento de 7 leds com apenas 3 terminais de saída utilizando o 74hc595:

