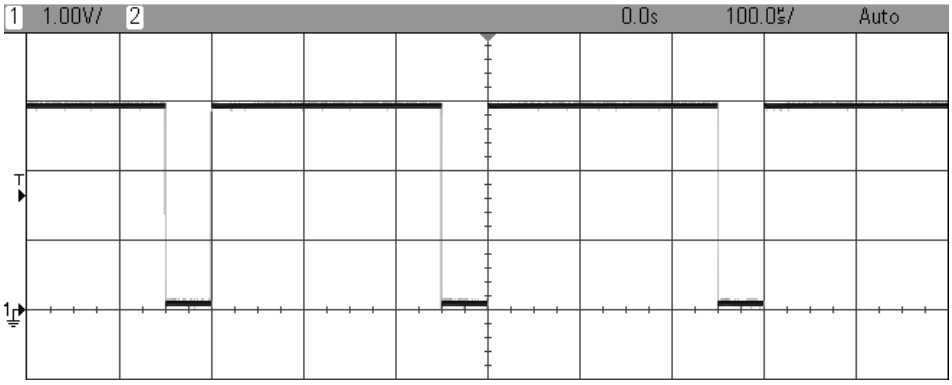
Exercício avaliativo 2

Instituição: Instituto Federal de Educação, Ciência e Tecnologia - Paraíba (IFPB).
Disciplina: Microprocessadores e microcontroladores**.**
Professor: Fagner de Araujo Pereira.
Aluno (a): _____

Exercício avaliativo 2 (Peso 100 pontos)

1. A figura abaixo mostra o sinal em um pino do microcontrolador STM32F407 observado na tela de um osciloscópio, ajustado verticalmente em 1V por divisão e horizontalmente em 100µs por divisão.

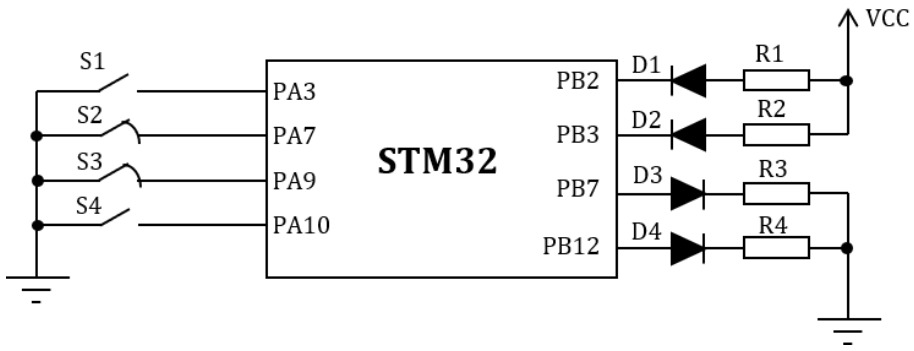


Considerando que o sinal observado é gerado ao executar o trecho de código seguinte, responda às questões que seguem.

```
while(1)
{
   GPIOC->ODR |= (1 << 9);        //nível lógico alto no pino
   Delay_us(X+Y);                 //atraso de alguns (X+Y) microssegundos
   GPIOC->ODR &= ~(1 << 9);       //nível lógico baixo no pino
   Delay_us(X-Y);                 //atraso de alguns (X-Y) microssegundos
}
```

a) Qual o pino utilizado?
b) Quais os valores de X e de Y?

2. Considere que os pinos do lado esquerdo da figura foram configurados como entradas com resistores de *pull-up* em um microcontrolador STM32. Os pinos foram conectados a chaves (Sn) que podem estar abertas ou fechadas, conforme mostrado. Considerando que o microcontrolador executa o código seguinte, determine o estado (aceso ou apagado) dos LEDS (Dn) conectados na porta B para a configuração de chaves apresentada.




```
while(1)
{
   uint16_t leitura = GPIOA->IDR;      //faz a leitura dos pinos da porta A
   leitura |= 0x93A1;                  //lógica OR
   leitura &= 0x7FBA;                  //lógica AND
   GPIOB->ODR = leitura;               //atualiza o estado dos pinos da porta B
}
```

3. Considerando ainda a mesma figura mostrada na questão anterior, escreva um trecho de código que faça periodicamente a leitura das chaves S e reproduza o seu estado nos LEDs D. Adote a seguinte regra:

** Se a chave Sn (n=1, 2, 3 ou 4) estiver fechada, o respectivo LED Dn (n=1, 2, 3 ou 4) deverá acender, e vice-versa. A leitura das chaves deve ser feita simultaneamente e a atualização do estado dos LEDs também deve ser feita simultaneamente.*

4. O modo de operação dos pinos de GPIO pode ser configurado em tempo de execução. Entretanto, é possível congelar os registradores de controle GPIO aplicando uma sequência de gravação específica ao registrador GPIOx_LCKR. Quando a sequência correta é aplicada ao bit 16 neste registrador, o valor de LCKR [15:0] é usado para bloquear a configuração das I/O's (durante a sequência de gravação o valor LCKR [15:0] deve ser o mesmo). Quando a sequência correta for aplicada a um bit de porta, o valor do bit de porta não pode mais ser modificado até o próximo reset do microcontrolador ou reinicialização do periférico GPIO. Cada bit GPIOx_LCKR congela o bit correspondente nos registradores de controle.
Explique como realizar o congelamento da configuração dos pinos PB2, PB3, PB13 e PB15 no STM32F407.
Dica: consulte a seção GPIO no manual de referência do STM32F407.

5. Os microcontroladores STM32 possuem diversas opções de fontes de clock e ainda permitem a escolha de diferentes velocidades. Isso traz flexibilidade aos projetistas de sistemas embarcados na adoção de estratégias que garantam alto desempenho ou baixo consumo de energia**.** O código mostrado a seguir configura o sistema de clock de um microcontrolador STM32F407 que usa um cristal externo de 25MHz. Após sua execução, qual a velocidade de clock nos barramentos, AHB, APB1 e APB2?

```
void Configure_Clock(void)
{
   #define PLL_M      5
   #define PLL_N      50
   #define PLL_P      2

   //Configura a fonte de clock (HSE), os fatores de divisão e multiplicação do PLL,
   //prescalers e dos barramentos AHB, APB
   RCC->CR |= 0x00010000;              //habilita HSE
   while(!((RCC->CR) & 0x00020000));   //espera HSE ficar pronto

   RCC->CFGR |= 0x00009400;    //HCLK = SYSCLK/1, PCLK2 = HCLK/2, PCLK1 = HCLK/4

   //configura a fonte de clock e os parâmetros do PLL
   RCC->PLLCFGR = PLL_M | (PLL_N << 6) | (((PLL_P >> 1) -1) << 16) | (0x00400000)
   RCC->CR |= 0x01000000;           //habilita o PLL
   while(!(RCC->CR & 0x02000000));  //espera o PLL ficar pronto

   //seleciona o PLL como fonte de SYSCLK
   RCC->CFGR |= 0x00000002;

   //espera o PLL ser a fonte de SYSCLK
   while((RCC->CFGR & 0x0000000C) != 0x00000008);
}
```