LABORATÓRIO TPSE I

Prática 02: Programando o Periférico GPIO como OUTPUT

Prof. Francisco Helder

15 de julho de 2024

1 Programando o Periférico GPIO

Muitas vezes, a melhor maneira de se familiarizar com uma nova plataforma de trabalho, tal como a BeagleBoneBlack, é iniciar utilizando as funcionalidades de GPIO, pois é mais fácil para começar a trabalhar com a plataforma e também todo projeto sempre tem LEDS, realizando ação de liga e desliga. Para fazer isso, você deve definir um pino como GPIO (entrada/saída de uso geral) e programa-lo como saída (output), então realiza o controle do componente a partir do seu estado. Nesse projeto iremos utilizar a documentação (**Technical Reference Manual** e **Processors Datasheet**) para realizar a programação do GPIO modulo 1 pino 28.

1.1 Ligação do Circuito para um LED

Primeiro, encontrar os pinos do processador para o GPIO que se encontram nos barramentos expansores P8 e P9, como visto na Figura 1, e que serão utilizados nas etapas que seguirão:

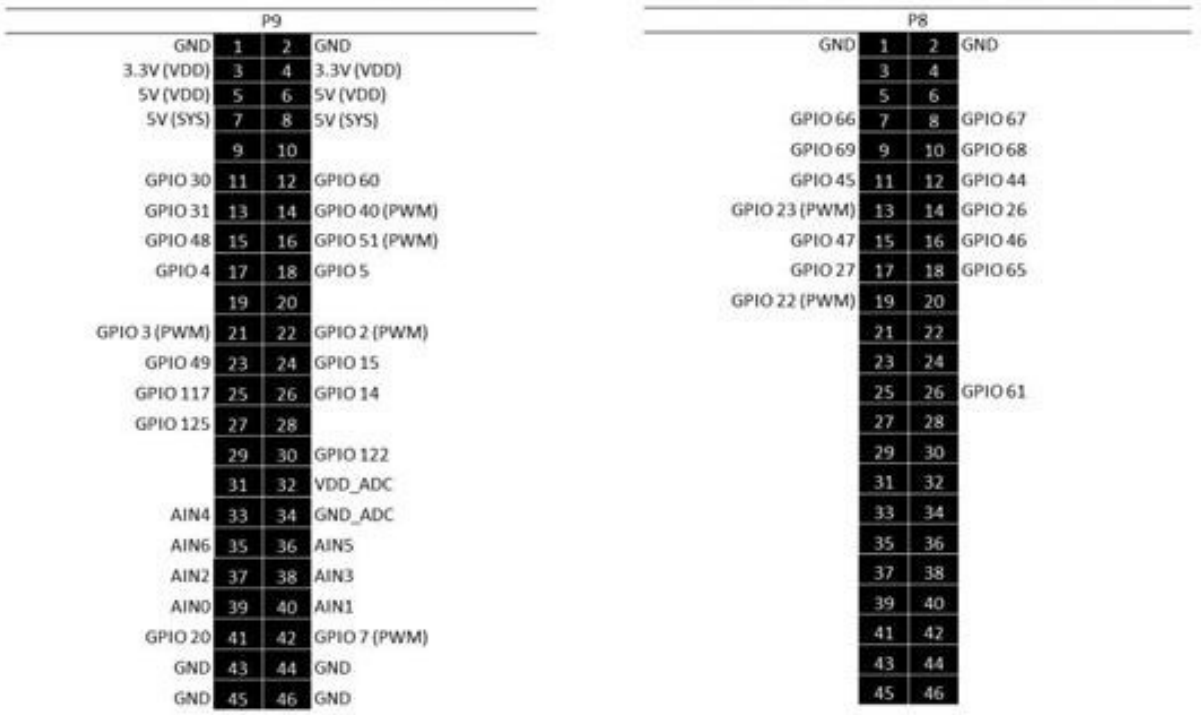

Figura 1: Pinout dos expansores P8 e P9.

Etapas para configurar GPIO via barramento expansor:

1. **Desligue a BeagleBone** - Antes de ligar qualquer coisa na BeagleBone, uma boa ideia é desligar e remover a fonte de energia.

2. **Conecte a tensão na BBB** - Usando um fio, ligue a tensão de 3.3V da BeagleBone (pinos de 3 ou 4 no expansor P9) para a trilha positiva da protoboard.

3. **Conecte ao terra** - Conectar o pino GND do BeagleBone (pinos 1 e 2 em ambos os expansores) a trilha negativa da protoboard.

4. **Ligue o pino GPIO para a protoboard** - Este exemplo usa GPIO1_28 (pino 12 no expansor P9). Use um jumper para conectá-lo a uma linha vertical na sua protoboard.

5. **Conecte o resistor** - Sem um resistor, um LED queima facilmente. Um resistor de 220 ou 470 deve cair a voltagem suficiente sem reduzir o brilho do LED demais. Ligue a resistência ao jumper que você puxou do pino 12, que liga eficazmente o resistor para GPIO1_28.

6. **Ligue o LED** - Ligue a perna negativa do LED - o cátodo, que normalmente é a perna mais curta - a faixa negativa da placa de ensaio onde estão ligados terra no Passo 3. Conecte a perna positiva - o ânodo - ao resistor.

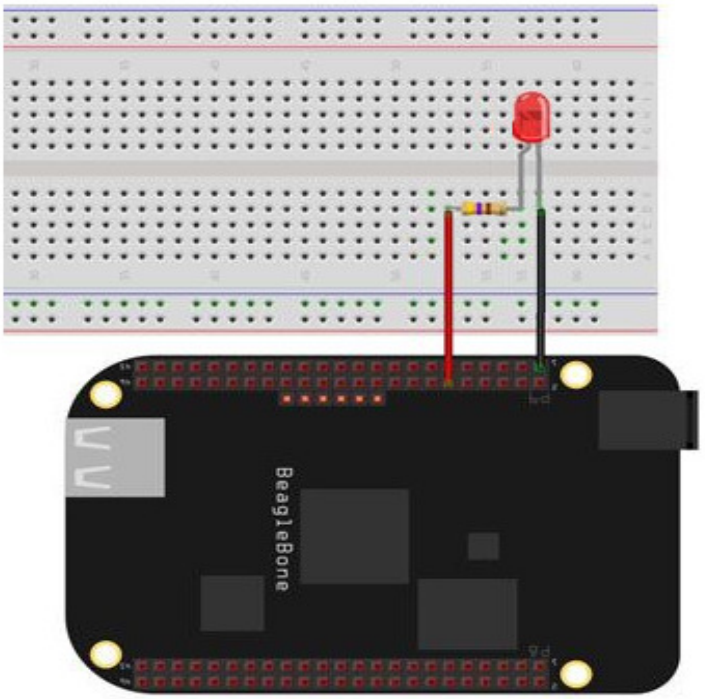

Figura 2: Circuito para ligação do LED.

Na Etapa 2, você conecta o pino de 3.3V do BeagleBone à placa protoboard. Na realidade, para este projeto específico, fazer essa conexão não serve para nada. Em geral, é uma boa prática, no entanto, ter sempre as faixas horizontais na sua placa de ensaio alimentado com uma tensão constante e com o terra do circuito.

você conectar o resistor a trilha positiva na sua placa de ensaio, o LED se acende, mas você não tem nenhum controle sobre ele. Sinta-se a vontade para experimentar!

1.2 Configurando e Controlando o pino 28 do GPIO

A interface de uso geral combina quatro módulos de input/output de uso geral (GPIO). Cada módulo GPIO fornece 32 pinos dedicados de uso geral com capacidades de entrada e saída; Assim, a interface de propósito geral suporta até 128 (4 × 32) pinos. O GPIO0 está no domínio Wakeup e pode ser usado para acordar o dispositivo por meio de fontes externas. GPIO[1:3] estão localizados no domínio periférico, como visto na Figura 3. Esses pinos podem ser configurados para as seguintes aplicações:

- Input(captura)/output(controle) de dados
- Interface de teclado
- Interromper a geração em modo ativo na detecção de eventos externos. Os eventos detectados são processados por dois submódulos paralelos independentes de geração de interrupção para dar suporte às operações do biprocessador.
- Geração de solicitação de despertar em modo inativo mediante detecção de eventos externos.

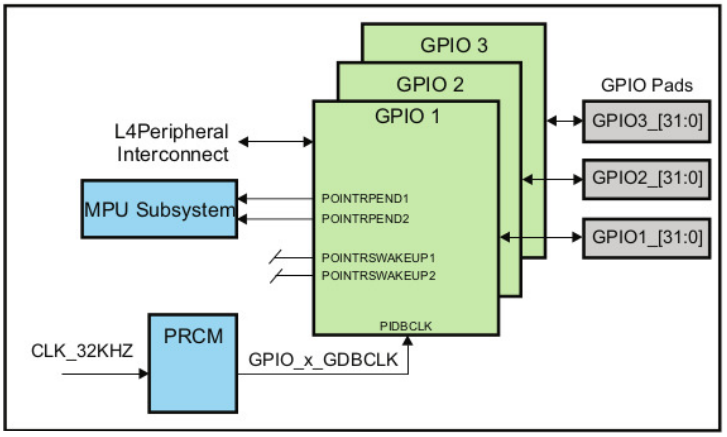

Figura 3: Módulos de GPIO[1-3].

1.2.1 Inicializando o Módulo de Clock do GPIO

O registrador PRCM base (SOC_CM_PER_REGS=0x44E0_0000) acompanhado do módulo de controle gerencia a passagem (isto é, a desativação) e a ativação dos clocks aos módulos da placa. Os clocks são gerenciados com base nas restrições de exigência dos módulos associados. Primeira coisa que deve realizar é a inicialização do módulo CM_PER clock para o GPIO módulo 1, como mostrado no exemplo abaixo.

```
#define CKM_PER_GPIO1_CLKCTRL (0x0AC)
#define SOC_PRCM_REGS       (0x44E00000)

#define SOC_CM_PER_REGS     SOC_PRCM_REGS + 0

HWREG(SOC_CM_PER_REGS + module) |= setting

setting   =  (1<<18) | (0x2<<0)
module    = CKM_PER_GPIO1_CLKCTRL
```

1.2.2 Inicializando o Módulo de Controle

O módulo de controle inclui status e lógica de controle não endereçados dentro dos periféricos ou o resto da infraestrutura da placa. Este módulo fornece interface para controlar as seguintes áreas do dispositivo:

- Functional I/O multiplexing
- Device control and status

- DDR PHY control and IO control registers
- EDMA event multiplexing control registers

O módulo de controle responde apenas ao tipo de dispositivo e POR (Power On Reset) interno. Ao ligar, os valores de reset para os registradores definem o estado seguro para o dispositivo. No modo de inicialização, somente os módulos a serem usados no momento da inicialização estão associados às PADs. Outros módulos de entrada são internamente ligadas e as placas de saída são desligadas. Após POR, o software define os registros de multiplexação funcional e de configuração do PAD para os valores desejados de acordo com a configuração da placa solicitada.

Os registos de controlo PAD são registadores de 32 bits para controlar o sinal multiplexado e outros aspectos de cada PAD I/O. Após o POR, o software deve configurar a funcionalidade de multiplexação do PAD e configurações dos registros para os valores desejados de acordo com a configuração solicitada. A configuração é controlada pelos PADs ou por um grupo de PAD. Cada pino configurável tem seu próprio registrador de configuração para controle pullup/down e para a atribuição a um dado módulo.

Para realizar a inicialização do pino 28 para o GPIO módulo 1, siga as instruções como mostrado no exemplo abaixo:

```
#define CM_conf_gpmc_ben1 (0x0878)
#define SOC_CONTROL_REGS  (0x44E10000)

mode  = 7;
module  = CM_conf_gpmc_ben1;

HWREG(SOC_CONTROL_REGS + module) |= mode;
```

1.2.3 Configurando Direção do GPIO

O registrador GPIO_OE é usado para habilitar os recursos de saída dos pinos. Na reinicialização, todos os pinos GPIO relacionados são configurados como entrada e capacidades de saída estão desativados. Este registrador não é usado com o módulo, sua única função é carregar a configuração dos PADs. Quando o aplicativo está usando um pino como uma saída e não quer a geração de interrupção a partir deste pino, o aplicativo pode/tem que configurar corretamente os registradores que habilita interrupção.

```
#define SOC_GPIO_1_REGS   (0x4804C000)
#define GPIO_OE       (0x134)

dir = 0;

addr_temp = SOC_GPIO_1_REGS + GPIO_OE;
val_temp =  HWREG(addr_temp);

val_temp &= ~(1<<pin);
val_temp |= (dir<<pin);

HWREG(addr_temp) = val_temp;
```

1.2.4 Realizando Controle do GPIO no modo OUTPUT

Registrador GPIO_DATAOUT (offset = 13Ch) [reset = 0h]

O registro GPIO_DATAOUT é usado para definir o valor dos pinos de saída GPIO. Os dados são escritos no registrador GPIO_DATAOUT de forma síncrona com o clock da interface.

Este registro pode ser acessado com operações diretas de leitura/gravação ou usando o recurso alternativo Set/Clear. Este recurso permite definir ou limpar bits específicos deste registro com um único acesso de escrita ao registro de setar a saída de dados (GPIO_SETDATAOUT) ou ao endereço de registro para limpar a saída de dados (GPIO_CLEARDATAOUT).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTLINE[n]																															

Figura 4: bits do registrador GPIO_DATAOUT, GPIO_SETDATAOUT e GPIO_CLEARDATAOUT.

Registrador GPIO_SETDATAOUT (offset = 194Ch) [reset = 0h]

Uma operação de escrita no registrador GPIO_SETDATAOUT, significa escrever 1 no bit no correspondente registrador, onde temos 32 bits para os 32 pinos (32 GPIO como saída) como mostrado na Figura 4; Um bit escrito 0 não tem efeito. Já Uma leitura no registrador GPIO_SETDATAOUT retorna o valor do registrador de saída de dados (GPIO_DATAOUT).

```
#define SOC_GPIO_0_REGS   (0x44E07000)
#define GPIO_SETDATAOUT   (0x194)

addr_temp = SOC_GPIO_0_REGS + GPIO_SETDATAOUT;
val_temp = 1<<pin;

HWREG(addr_temp) |= val_temp;
```

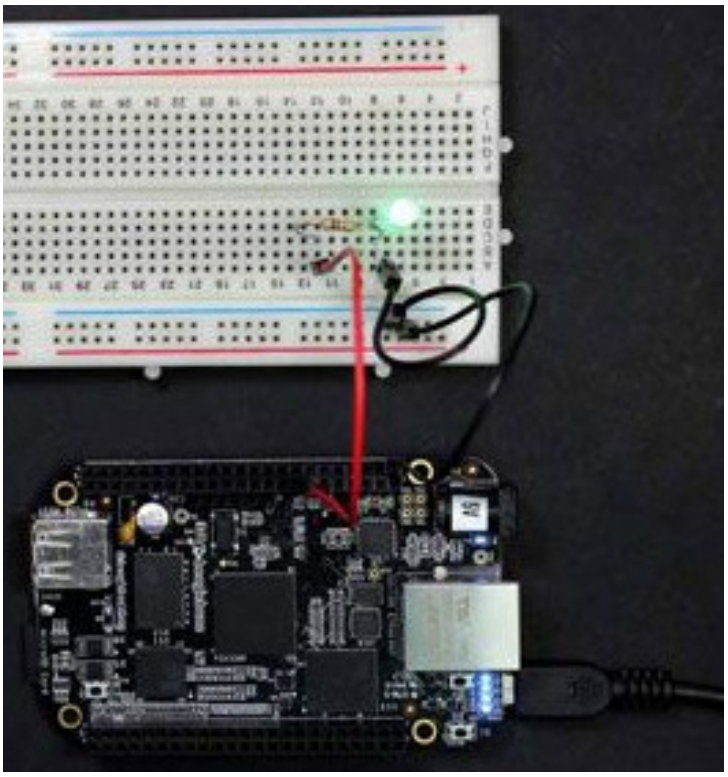

Figura 5: sistema completo da prática pisca LED.

Registrador GPIO_CLEARDATAOUT (offset = 190Ch) [reset = 0h]

Uma operação de escrita no registrador GPIO_CLEARDATAOUT, significa escrever 1 no bit correspondente no registrador GPIO_DATAOUT, onde temos 32 bits para 32 pinos (32

GPIO como saída) como mostrado na Figura 4; Um bit escrito 0 não tem efeito. Uma leitura do registrador GPIO_SETDATAOUT retorna o valor do registrador de saída de dados (GPIO_DATAOUT).

```
#define SOC_GPIO_0_REGS    (0x44E07000)
#define GPIO_SETDATAOUT    (0x190)

addr_temp = SOC_GPIO_0_REGS + GPIO_CLEARDATAOUT;
val_temp = 1<<pin;

HWREG(addr_temp) |= val_temp;
```

O resultado final da prática da placa com um circuito simples para relaização de um pisca LED é apresentado na Figura 5.

Se a intensidade luminosa do LED parece fraca, tente um valor menor de resistência. Não tente inferior a 220 Ohms, apesar de tudo.

2 Atividades Práticas

pratica 1:

Melhore esse sistema pisca LED, alterando a frequência (alta ou baixa) do pisca.

pratica 2:

Adicione no sistema pisca LED os quatros LEDs internos na placa (USR0, USR1, USR2 e USR3), e realize alterações na aplicação para controlar os cincos LEDs das mais diversas formas.