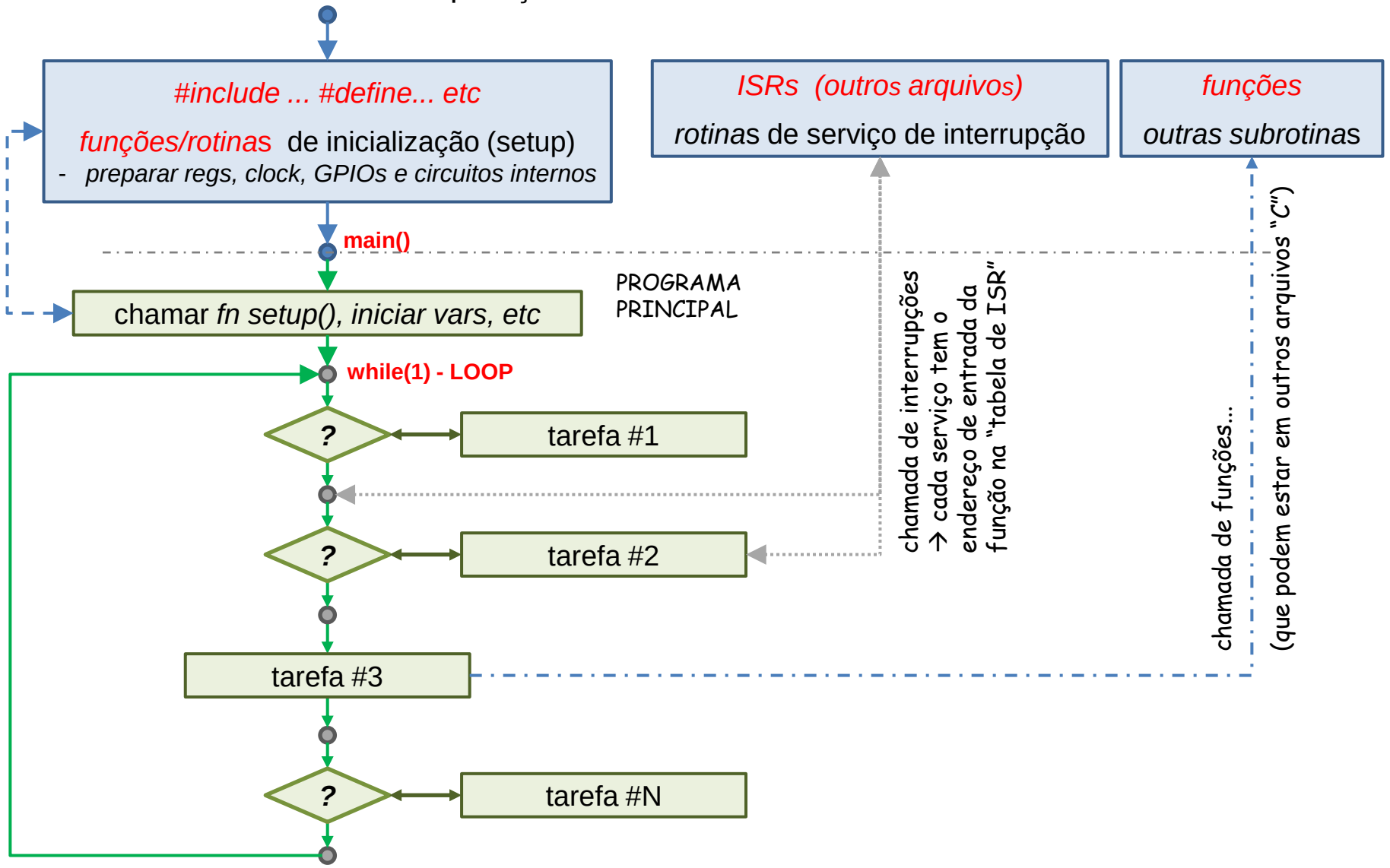
Bom dia,
Boa tarde,
Boa noite..!

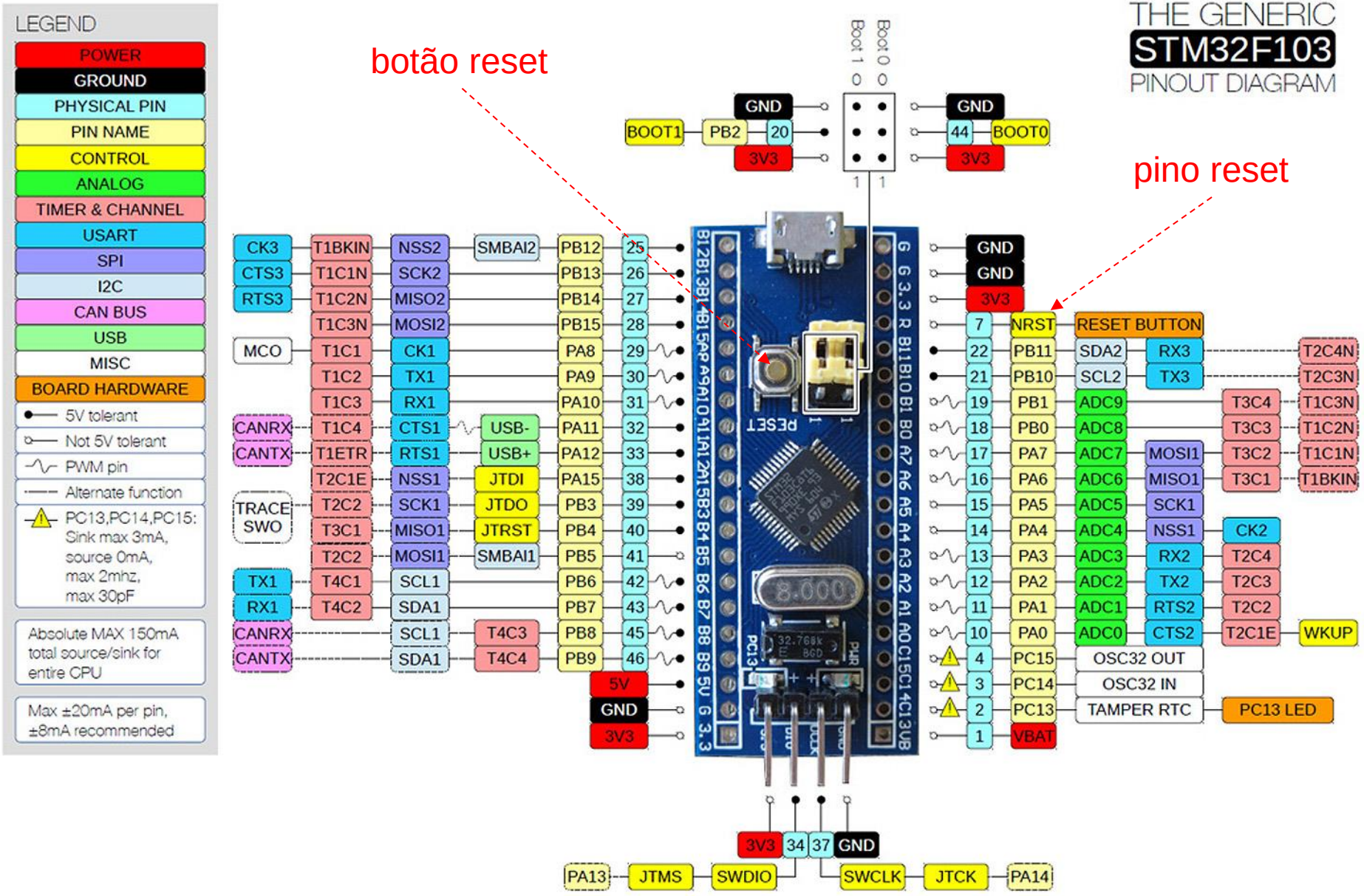
- Núcleo de PROCESSAMENTO do ARM (hardware)
- Básico de programação (Interrupções, Pilha, Funções)
- Conjunto de Instruções (ISA = *Instruction Set Architecture*)
- Programação ASSEMBLY para ARM Cortex

➢ Reformulando o modo de operação da CPU... conceito de SUPERLOOP...



- Núcleo de PROCESSAMENTO do ARM (hardware)
- Básico de programação (Interrupções, Pilha, Funções)
- Conjunto de Instruções (ISA = Instruction Set Architecture)
- Programação ASSEMBLY para ARM Cortex

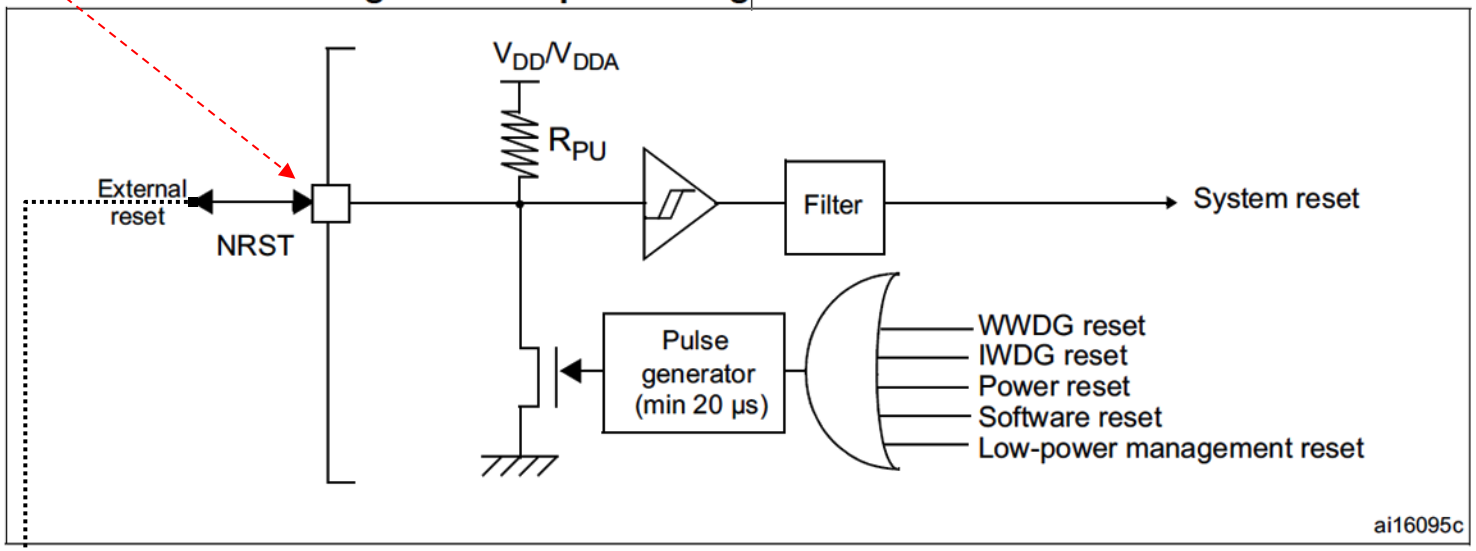
- RESET
- Watchdog timers
- Temporizadores (timers de propósito geral e específicos)
- PWM (saída analógica ainda que usando níveis lógicos digitais)
- GPIO (General Purpose Input / Output)

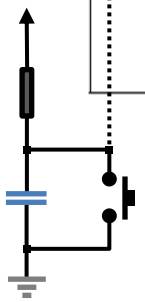

LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
botão reset
pino reset
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
GND
BOOT1
PB2
20
3V3
44
BOOT0
RESET
8.000
32.768k
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
CK2
T2C4
T2C3
T2C2
T2C1E
WKUP
PC13 LED
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14

- μProcessadores e SoC geralmente podem ser resetados de várias formas. No ARM:

Pino no chip NRST (externo)

Figure 7. Simplified diagram of the reset circuit

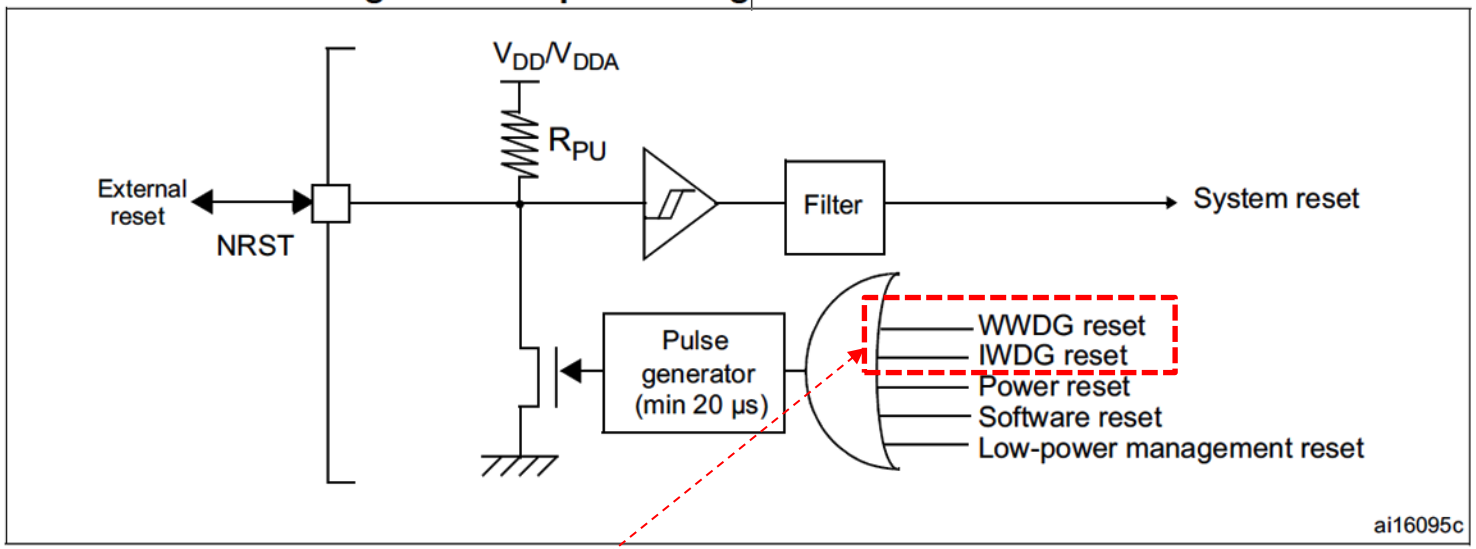




Circuito na placa...

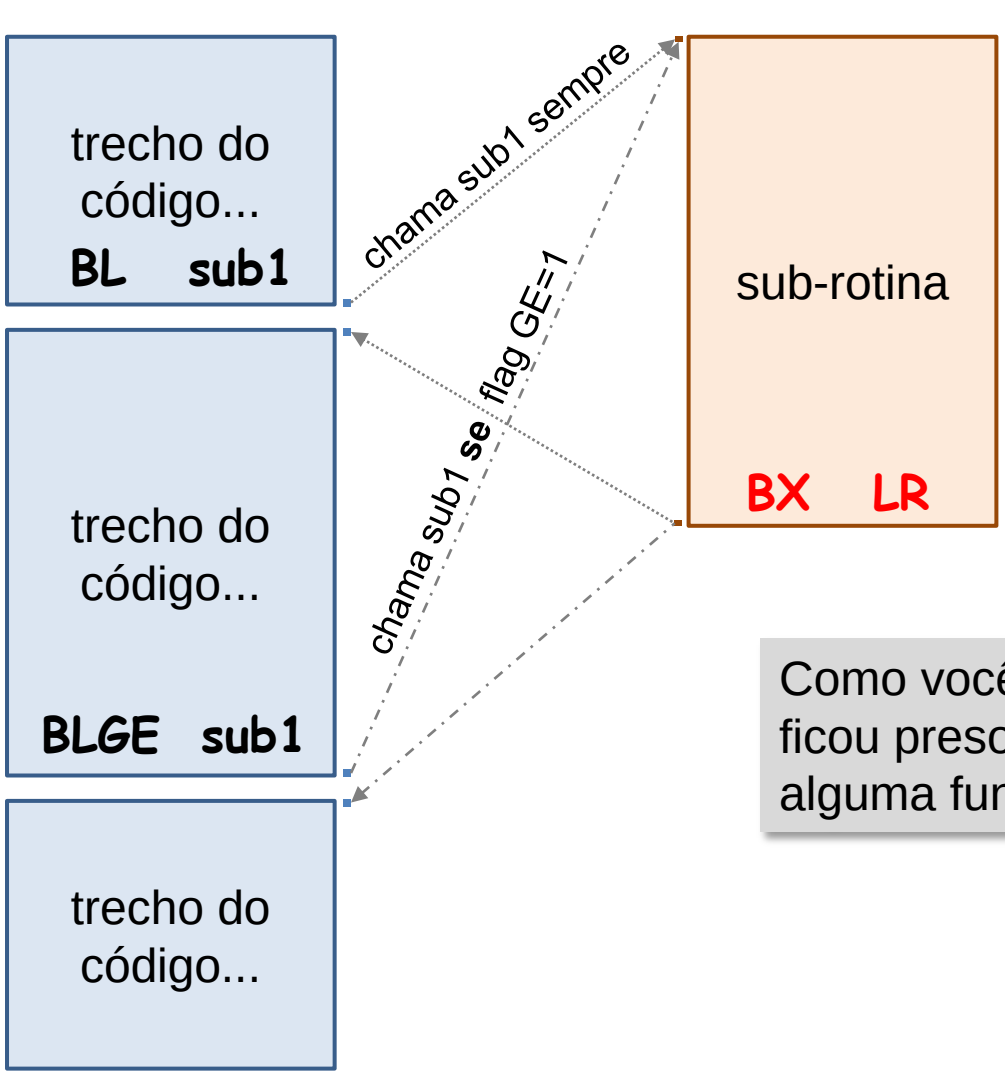
- μProcessadores e SoC geralmente podem ser resetados de várias formas. No ARM:

Figure 7. Simplified diagram of the reset circuit



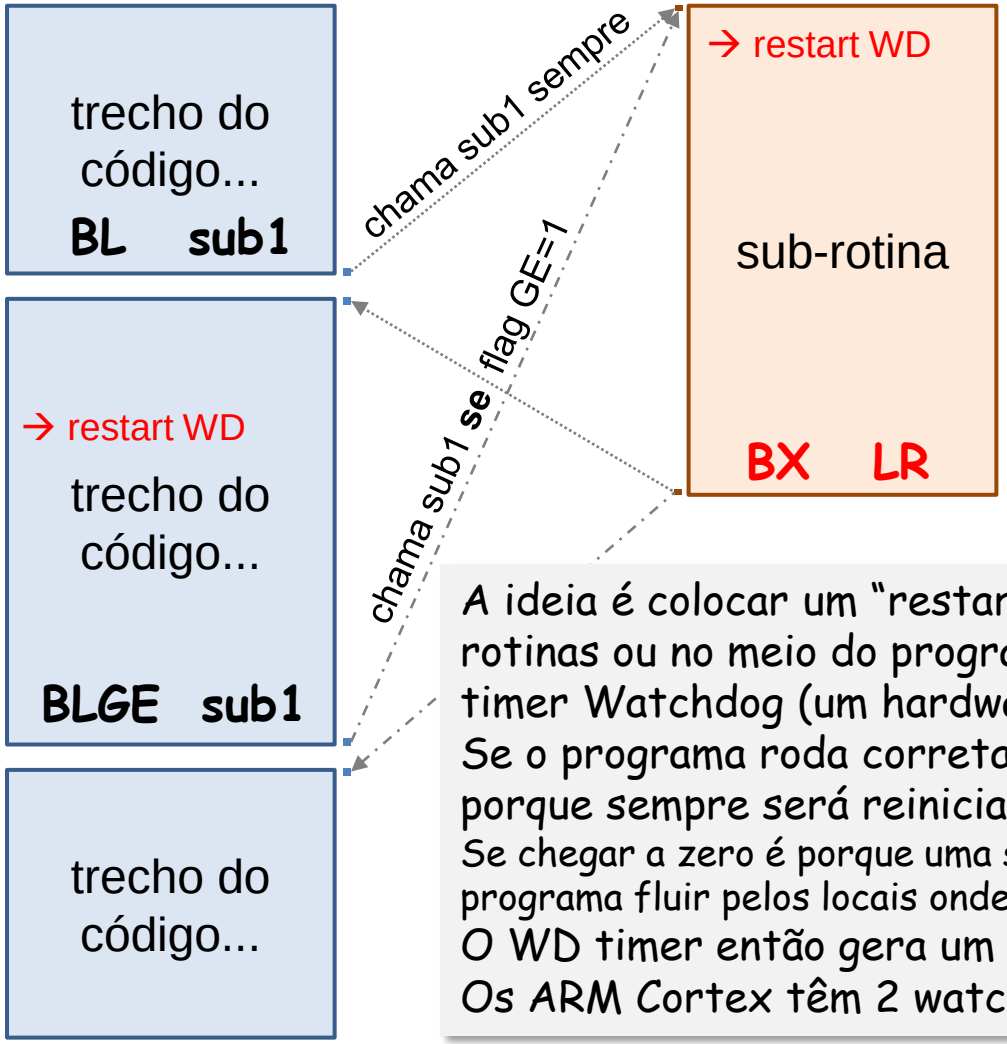
Watchdog timers !

- Recordando o conceito de sub-rotina (ou chamada de função)



Como você sabe se seu software ficou preso (em loop infinito) em alguma função ???

- Recordando o conceito de sub-rotina (ou chamada de função)



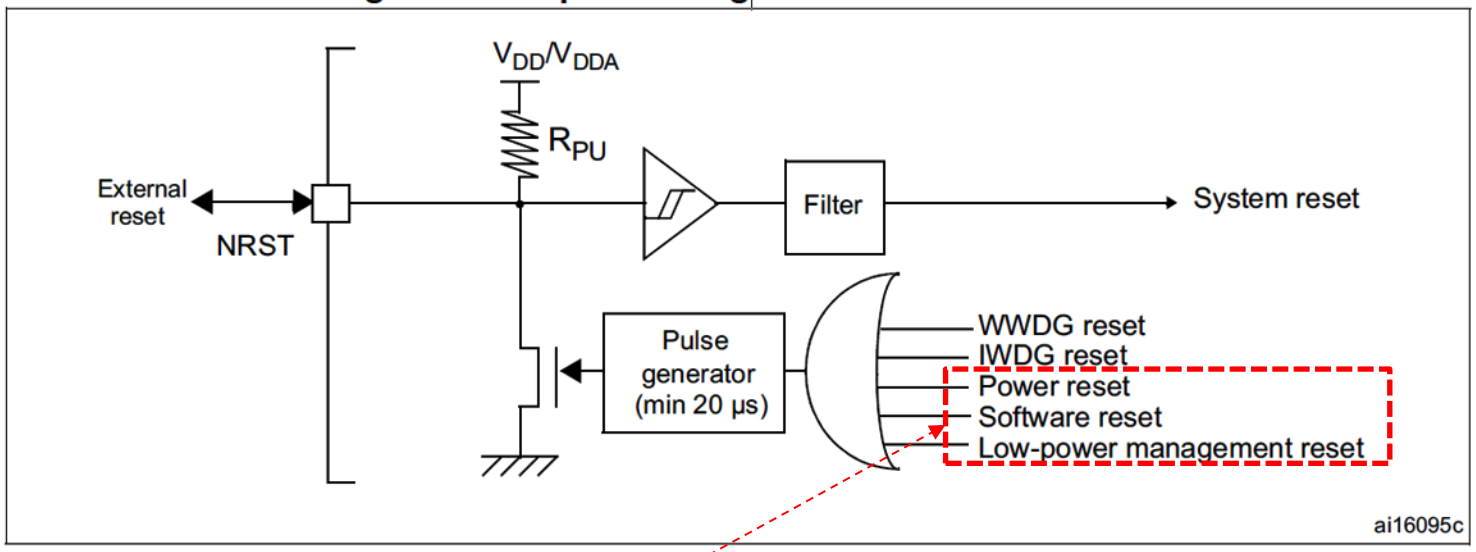
label: sub1

Um **watchdog timer** é um dispositivo eletrônico temporizador que dispara um "reset" ao sistema se o programa principal, devido a alguma condição de erro, deixar de fazer restart no **watchdog timer** *(Wikipedia)*

A ideia é colocar um "restart" do WD em entradas de sub-rotinas ou no meio do programa. É um reinício de contagem do timer Watchdog (um hardware que conta decrescente).
Se o programa roda corretamente, nunca o WD chega a zero, porque sempre será reiniciado de tempos em tempos...
Se chegar a zero é porque uma situação anormal (loop) está impedindo o programa fluir pelos locais onde há "restart do watchdog".
O WD timer então gera um pedido de interrupção ou reset.
Os ARM Cortex têm 2 watchdogs: IWDG e WWDG.

- μProcessadores e SoC geralmente podem ser resetados de várias formas. No ARM:

Figure 7. Simplified diagram of the reset circuit



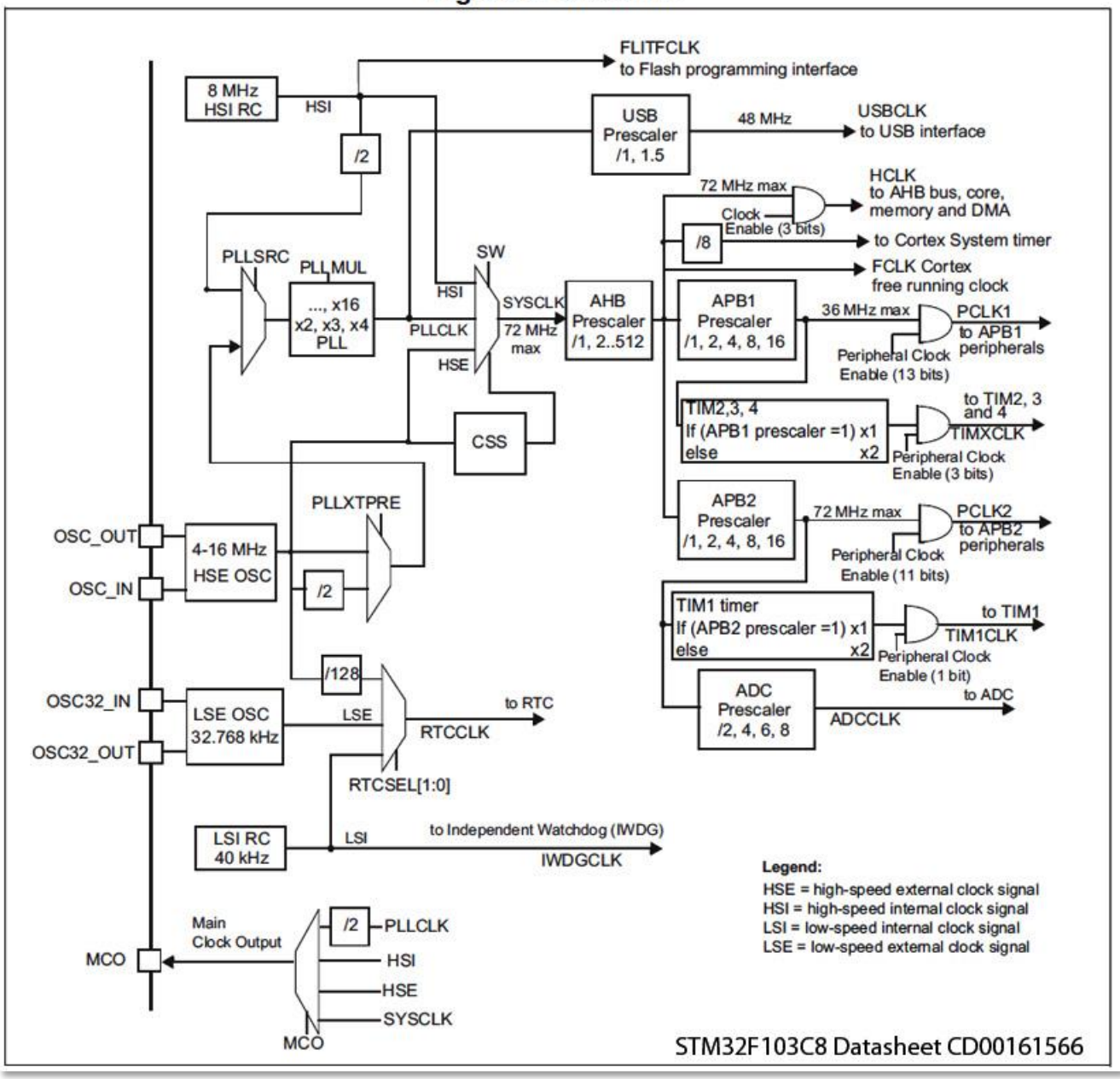
Outras formas de reset !

UFABC

✧ Sistemas de CLOCK dos processadores ARM

- μP e SoC podem ser programados para operar em frequências desejadas...

Figure 2. Clock tree



UFABC

✧ Sistemas de CLOCK dos processadores ARM

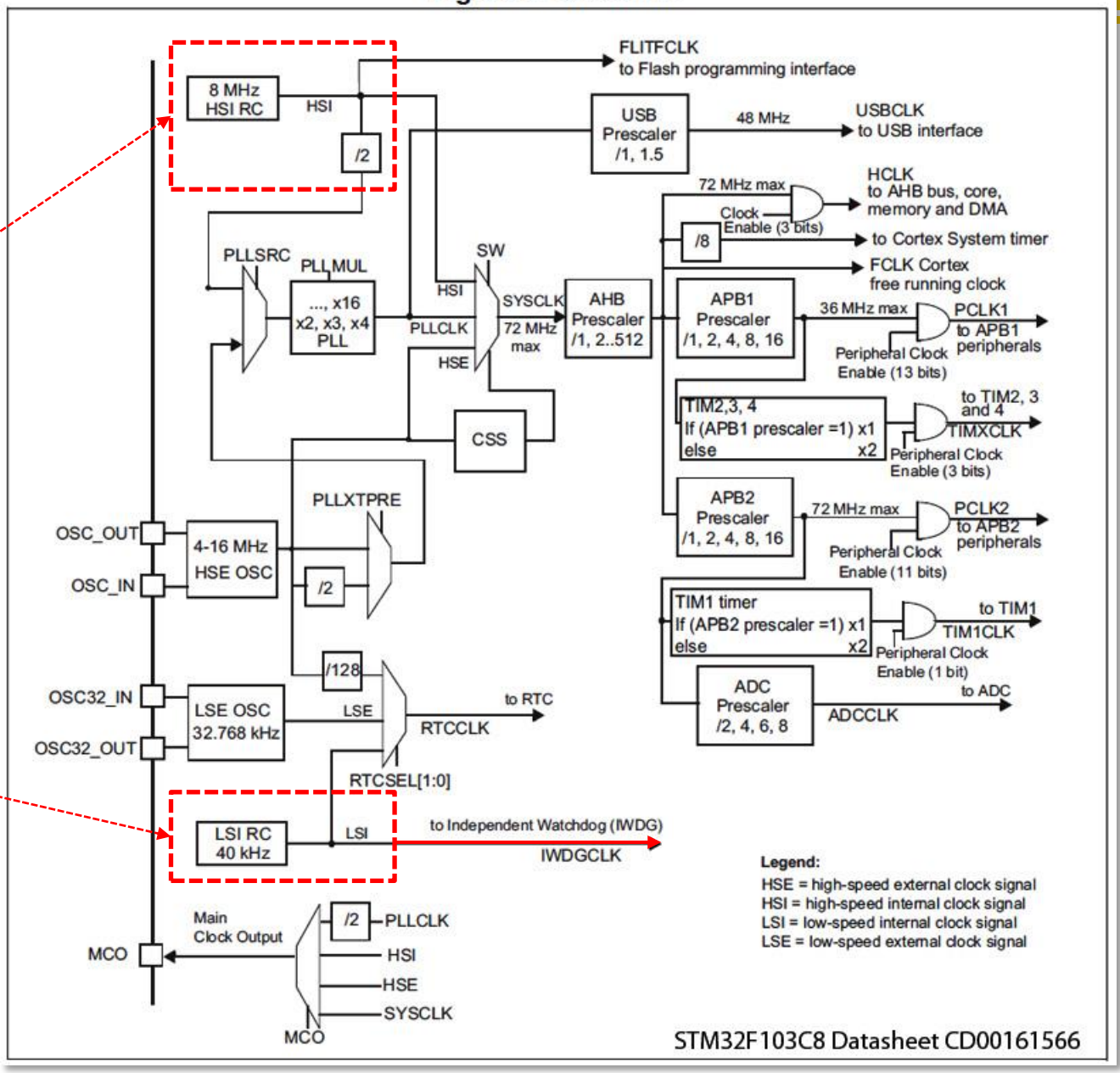
- μP e SoC podem ser programados para operar em frequências desejadas...

HSI RC (8MHz)
High-Speed Internal clk

ARM Cortex-M3 têm
2 geradores de clk internos

LSI RC (40KHz)
Low-Speed Internal clk

Figure 2. Clock tree



UFABC

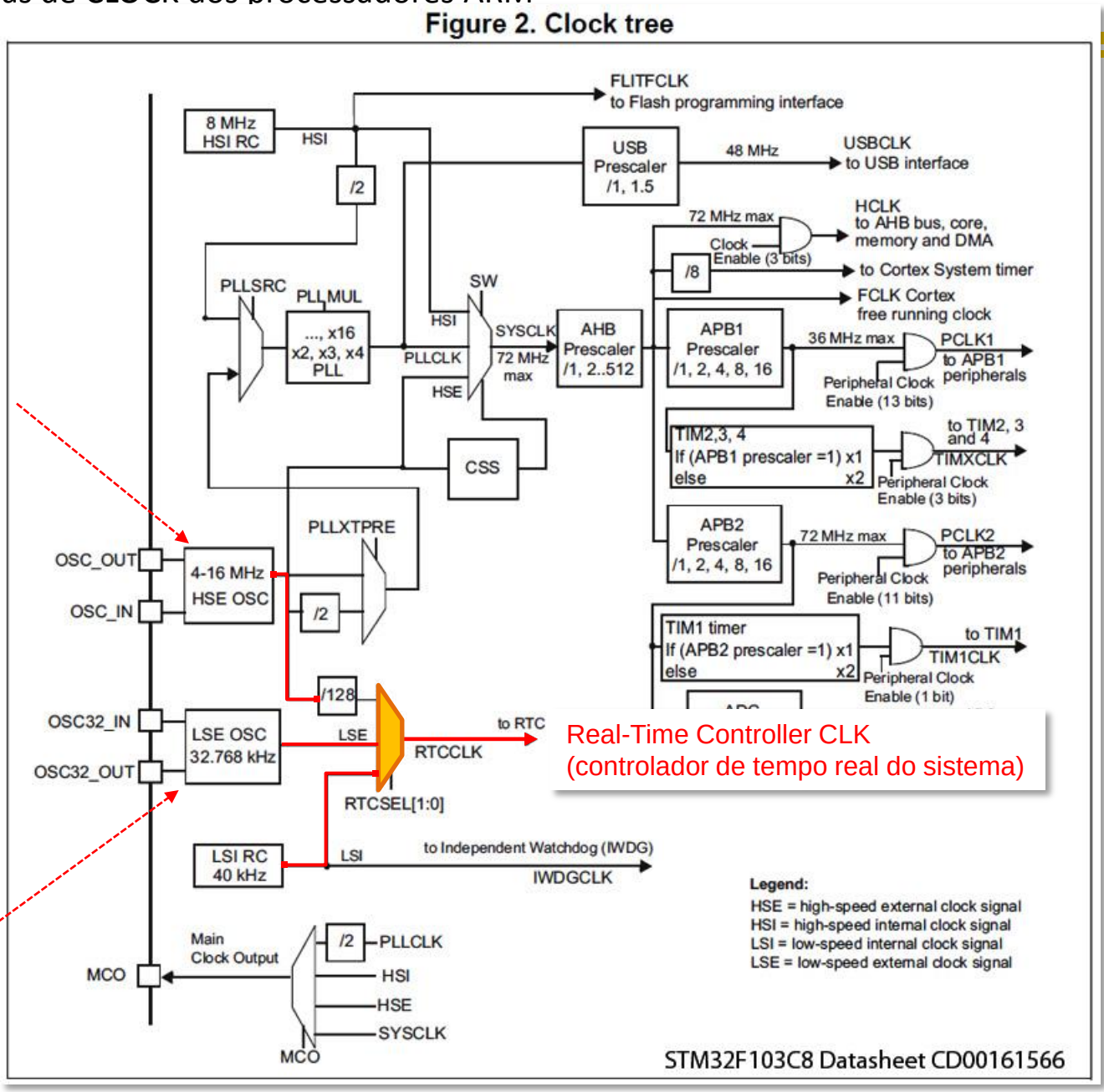
✧ Sistemas de CLOCK dos processadores ARM

- μP e SoC podem ser programados para operar em frequências desejadas...

oscilador externo (cristal 4...16 MHz)
High-Speed External clk

oscilador externo (cristal 32.768 KHz)
Low-Speed External clk

Figure 2. Clock tree



Real-Time Controller CLK
(controlador de tempo real do sistema)

UFABC

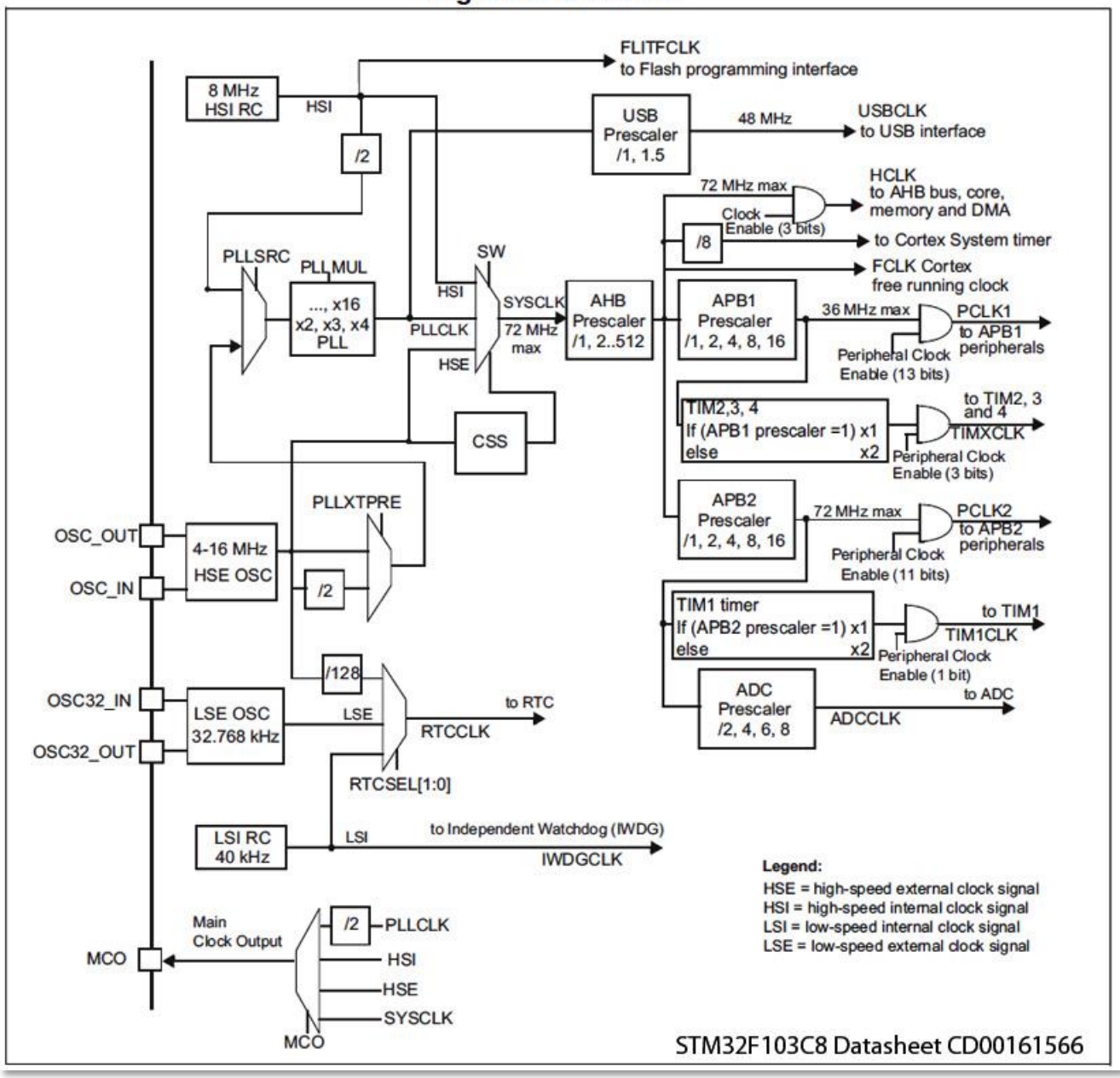
✧ Sistemas de CLOCK dos processadores ARM

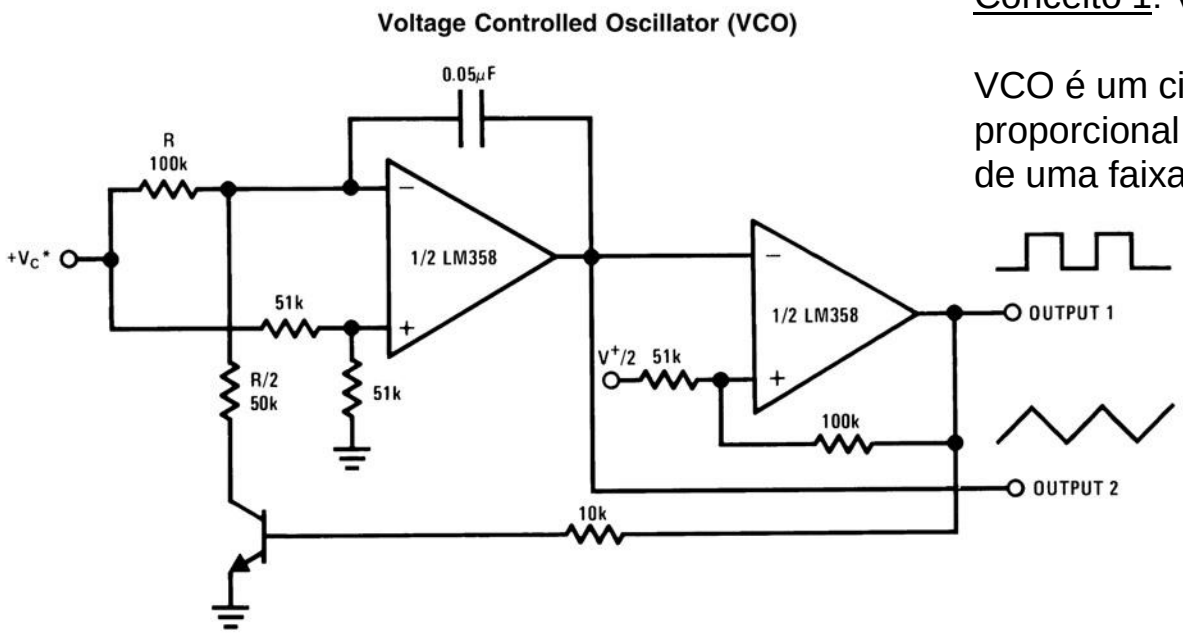
- μP e SoC podem ser programados para operar em frequências desejadas...

Multiplicador de frequência e PLL

Multiplicador? Pele... o que?

Figure 2. Clock tree



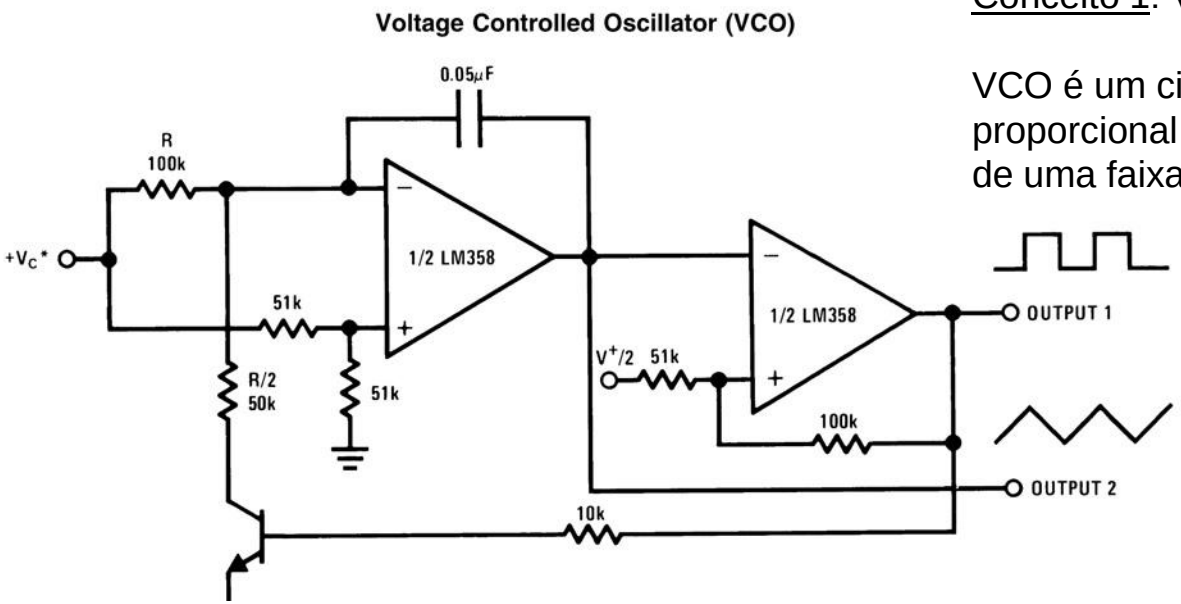


Conceito 1: VCO (*Voltage-Controlled Oscillator*)

VCO é um circuito que gera uma frequência de saída proporcional à tensão de controle de entrada, dentro de uma faixa pré-determinada:

Por exemplo:

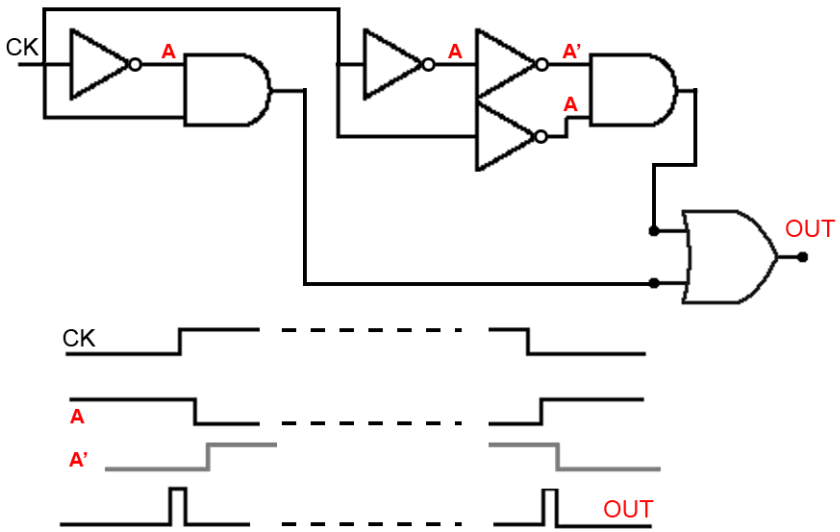
Vin	Freq
1 V	20 MHz
...	...
4 V	80 MHz



Conceito 1: VCO (*Voltage-Controlled Oscillator*)

VCO é um circuito que gera uma frequência de saída proporcional à tensão de controle de entrada, dentro de uma faixa pré-determinada:

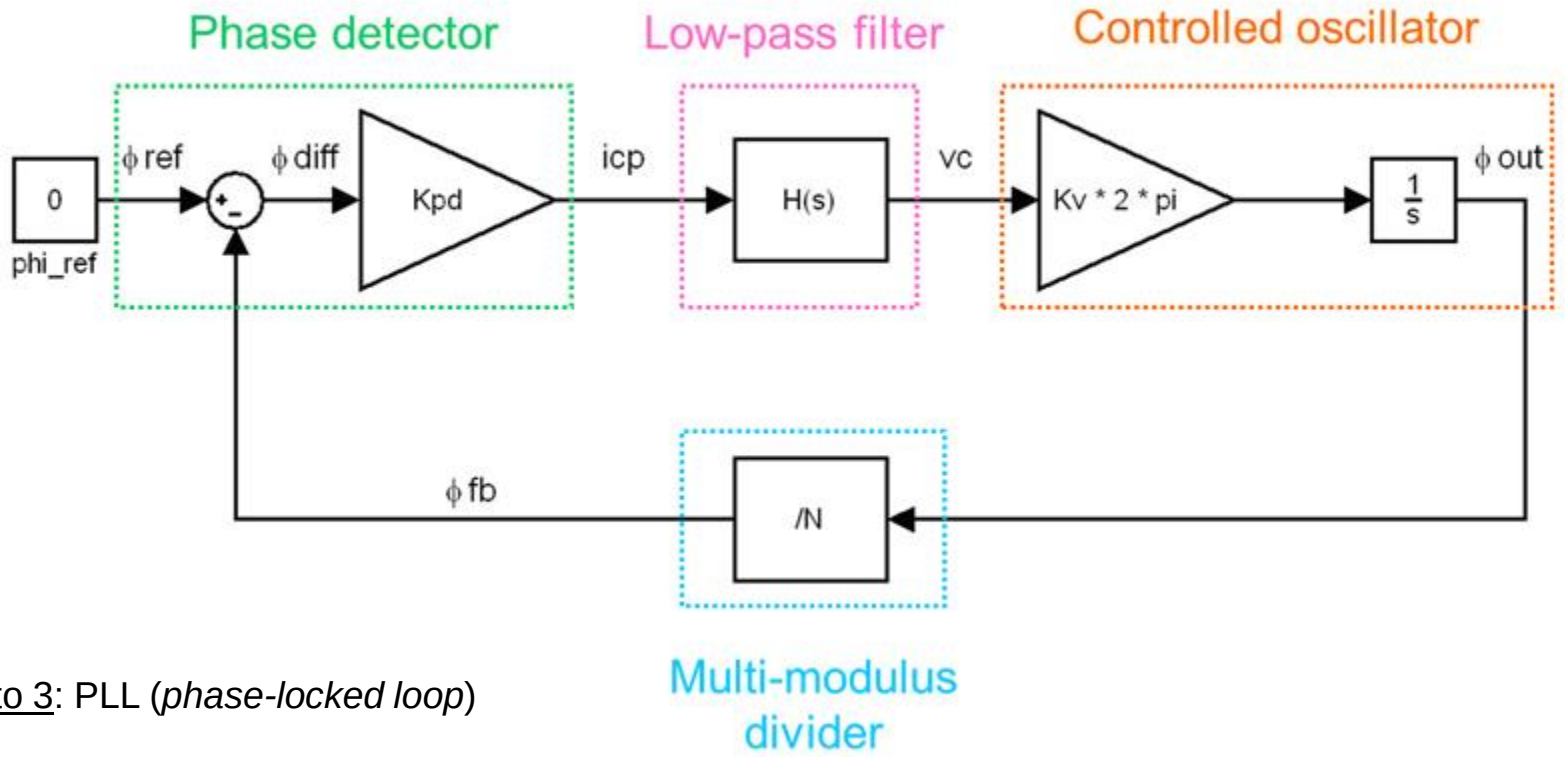
```
Por exemplo:
Vin          Freq

1 V          20 MHz
...            ...
4 V          80 MHz
```



Conceito 2: Multiplicador de frequência...

Circuito que multiplica (geralmente base 2) o clock de entrada gerando frequências mais altas.

Ex: Duplicador de frequência simples, gera um pulso de saída tanto na subida quanto na descida do CK.



Conceito 3: PLL (*phase-locked loop*)

Circuito que gera um sinal de saída cuja FASE está relacionada com a fase do sinal de entrada.

O circuito geralmente tem um VCO e um detector de fase que compara a saída do VCO com uma frequência de referência. Manter a fase de entrada e saída no “passo de bloqueio” implica manter as frequências de entrada e saída estáveis (iguais ou múltiplas uma da outra).

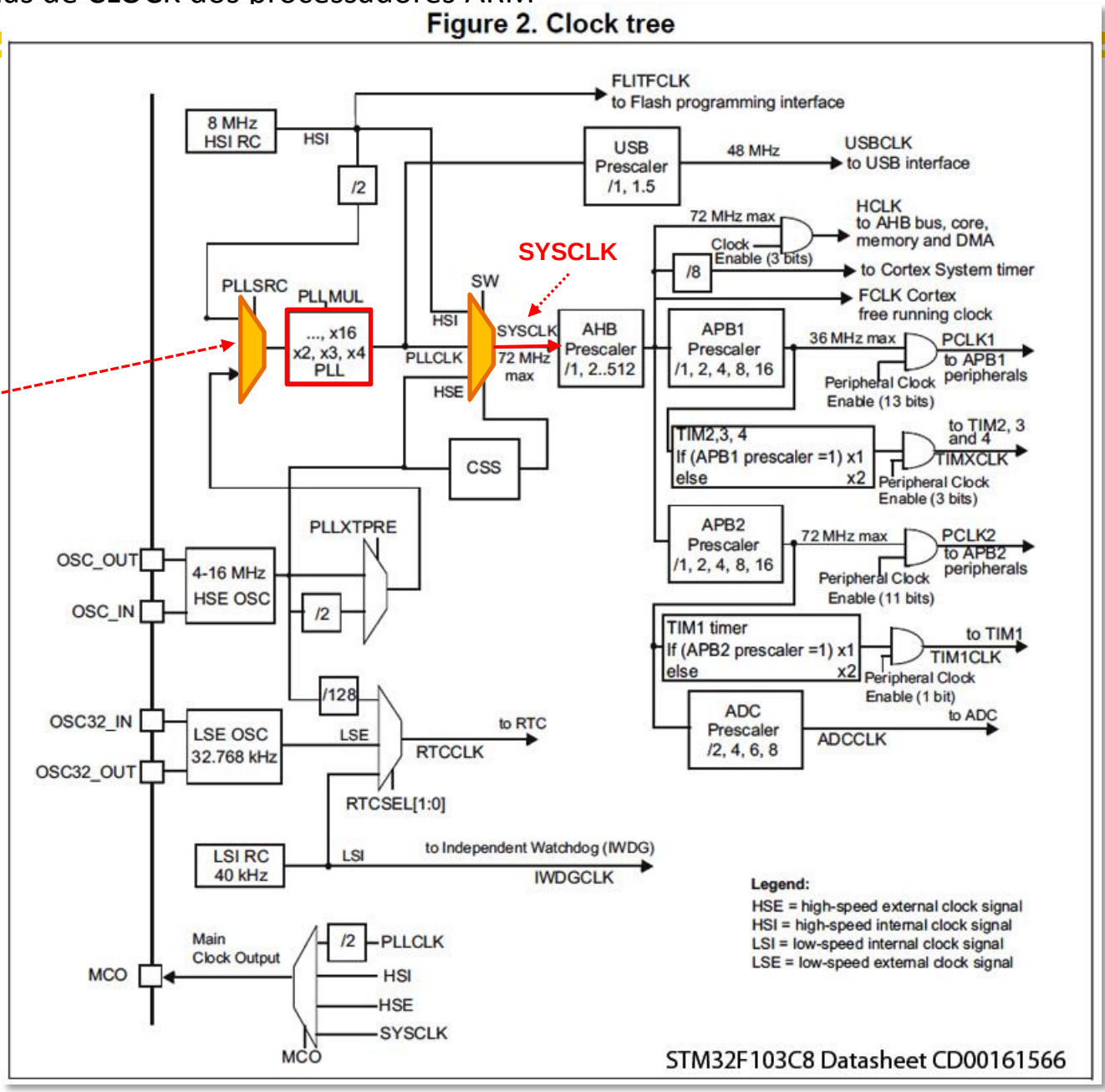
Exemplo: se (1) ***phi_ref*** = sinal de ***clock*** em 8MHZ e (2) o VCO opera em 72MHz, então (3) o módulo divisor que gera φ fb deve dividir a saída do VCO por “9”.
Ou seja, um sinal de 8MHz serve de referência para o VCO do circuito PLL gerar 72 MHz de forma estável.

UFABC

✧ Sistemas de CLOCK dos processadores ARM

- μP e SoC podem ser programados para operar em frequências desejadas...

Multiplicador de frequência e PLL



UFABC

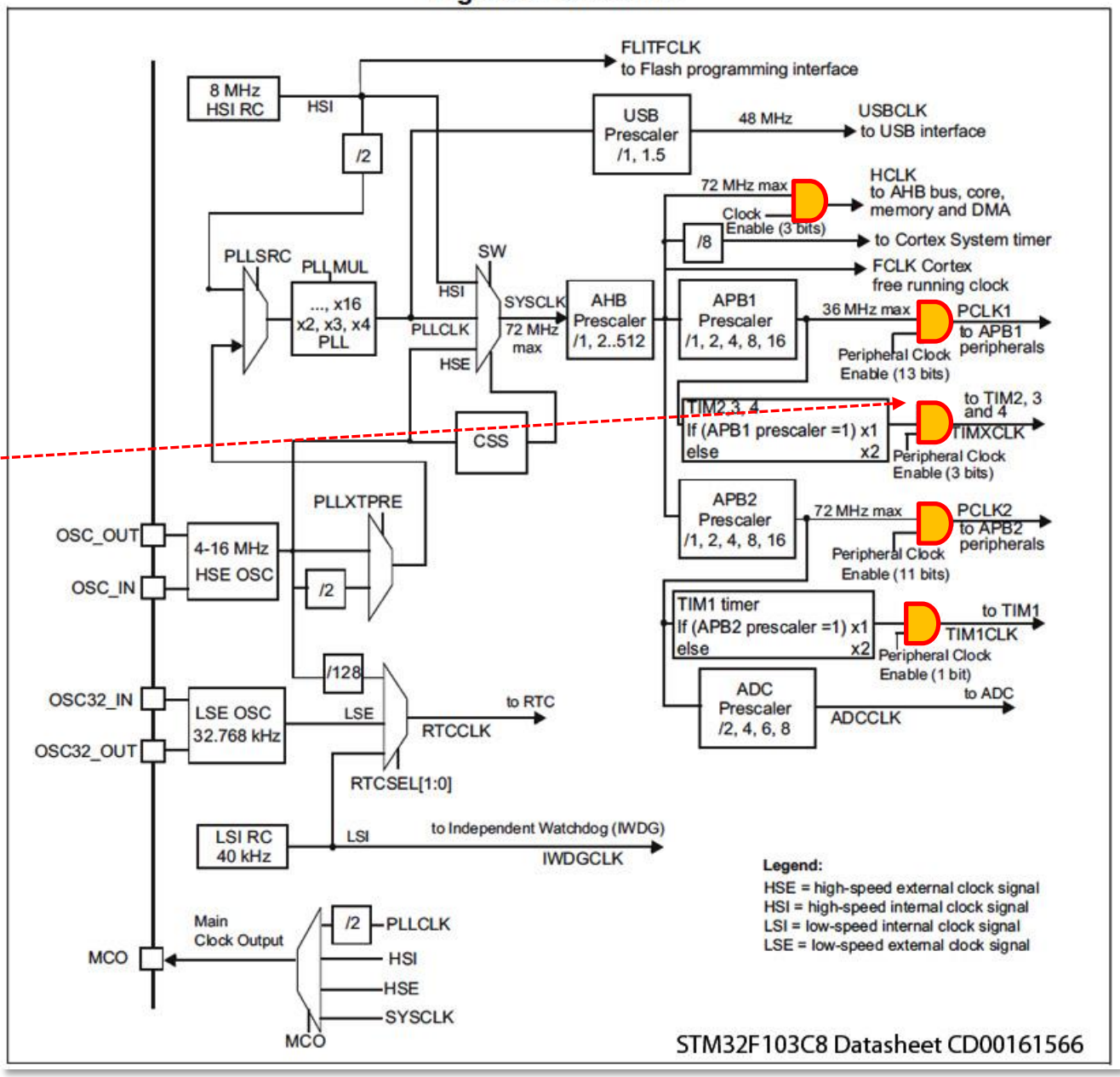
✧ Sistemas de CLOCK dos processadores ARM

- μP e SoC podem ser programados para operar em frequências desejadas...

CLOCK GATES

(as que temos que programar para habilitar os clocks nos periféricos) !

Figure 2. Clock tree



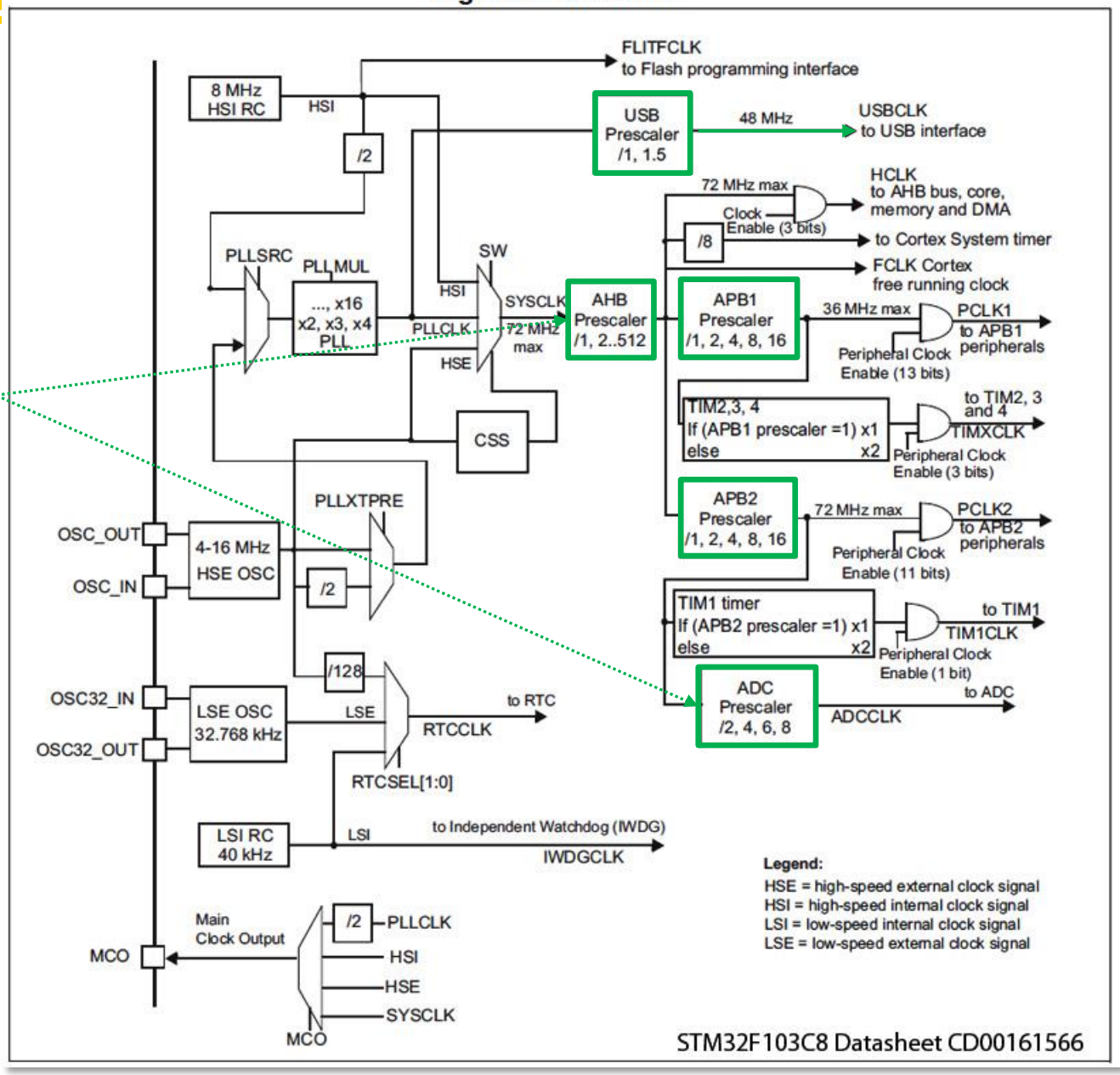
UFABC

✧ Sistemas de CLOCK dos processadores ARM

- μP e SoC podem ser programados para operar em frequências desejadas...

PRESCALERs
(divisores do clock SYSCLK para adequar a frequência para os timers, ADC, etc...)

Figure 2. Clock tree



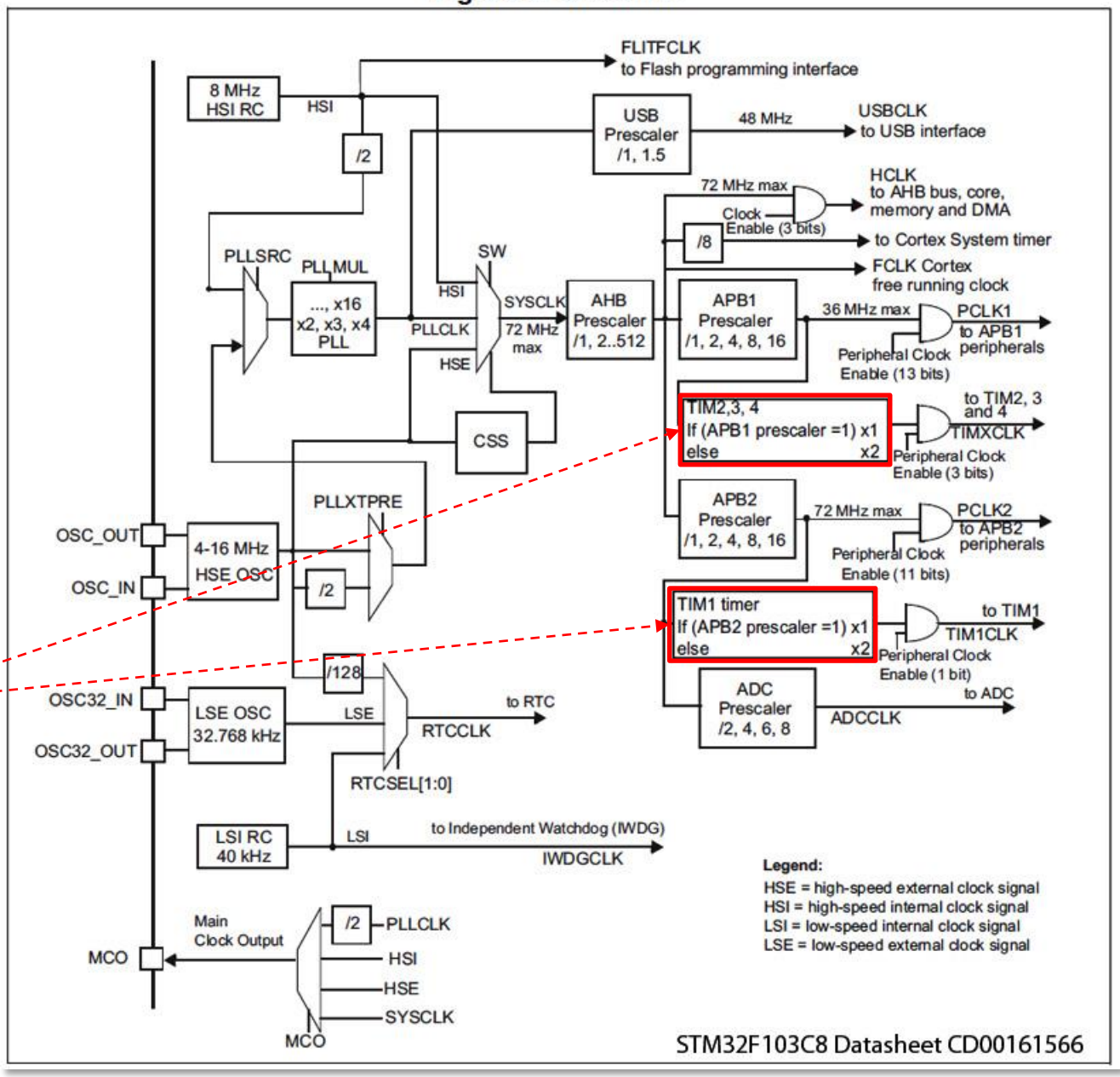
UFABC

✧ Sistemas de CLOCK dos processadores ARM

- μP e SoC podem ser programados para operar em frequências desejadas...

TIMERs
(nosso próximo assunto...)

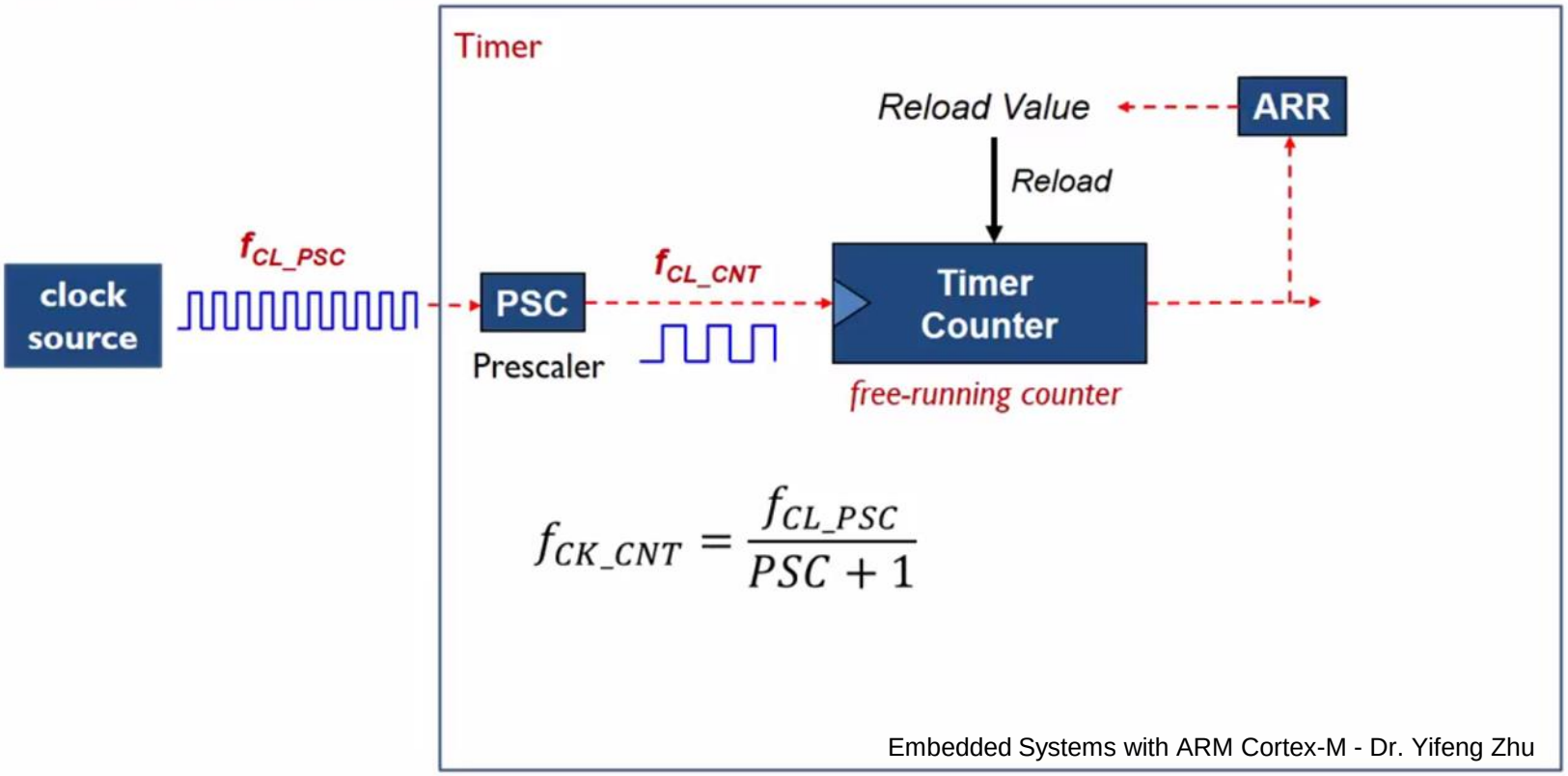
Figure 2. Clock tree



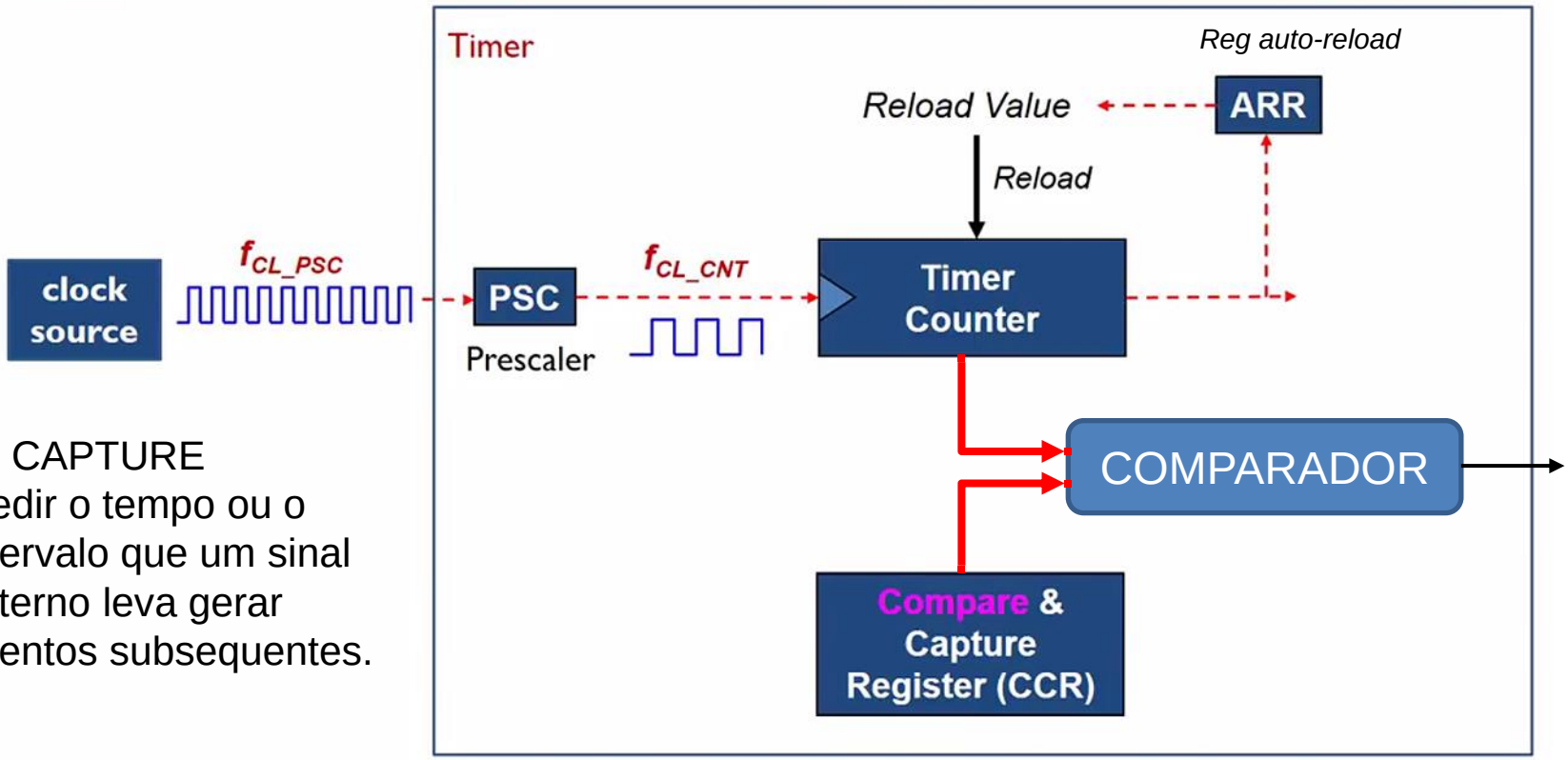
✧ Timers do processador ARM Cortex

- timers são contadores que operam paralelo à CPU gerando bases temporais para o sistema.
- ARM tem até 16 timers de uso geral (STM32F103C8 tem 4 timers)
- são independentes e podem contar UP, DOWN ou UP-DOWN
- Systick é um timer dedicado 24 bits (*já usamos para criar base de tempo 1 ms*)

Esquema básico de um timer no Cortex-M:



✧ Timers do processador ARM Cortex



Modo CAPTURE

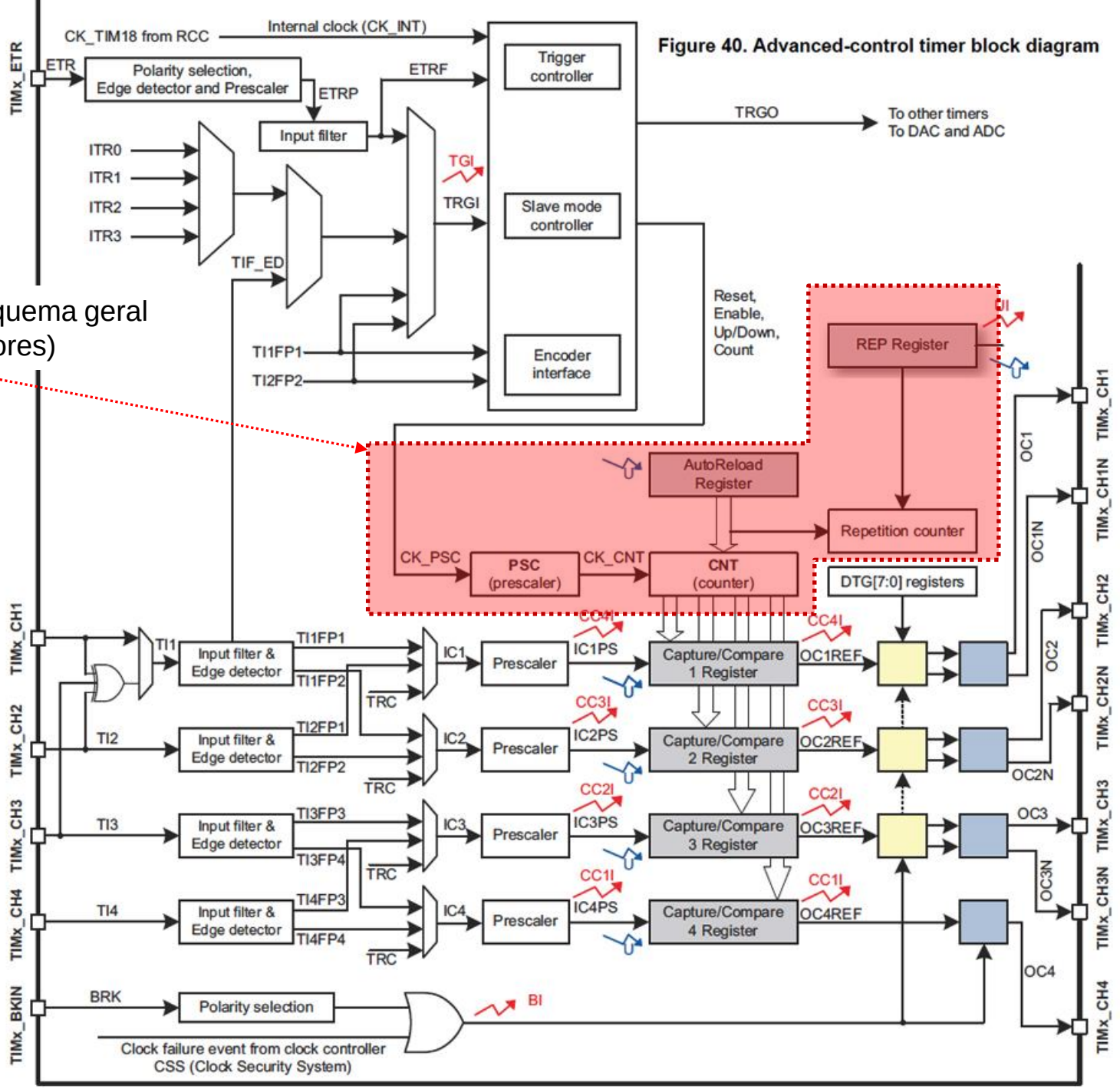
→ Medir o tempo ou o intervalo que um sinal externo leva gerar eventos subsequentes.

Modo COMPARAÇÃO (compara o *timer counter* com o valor de um registrador CCR):

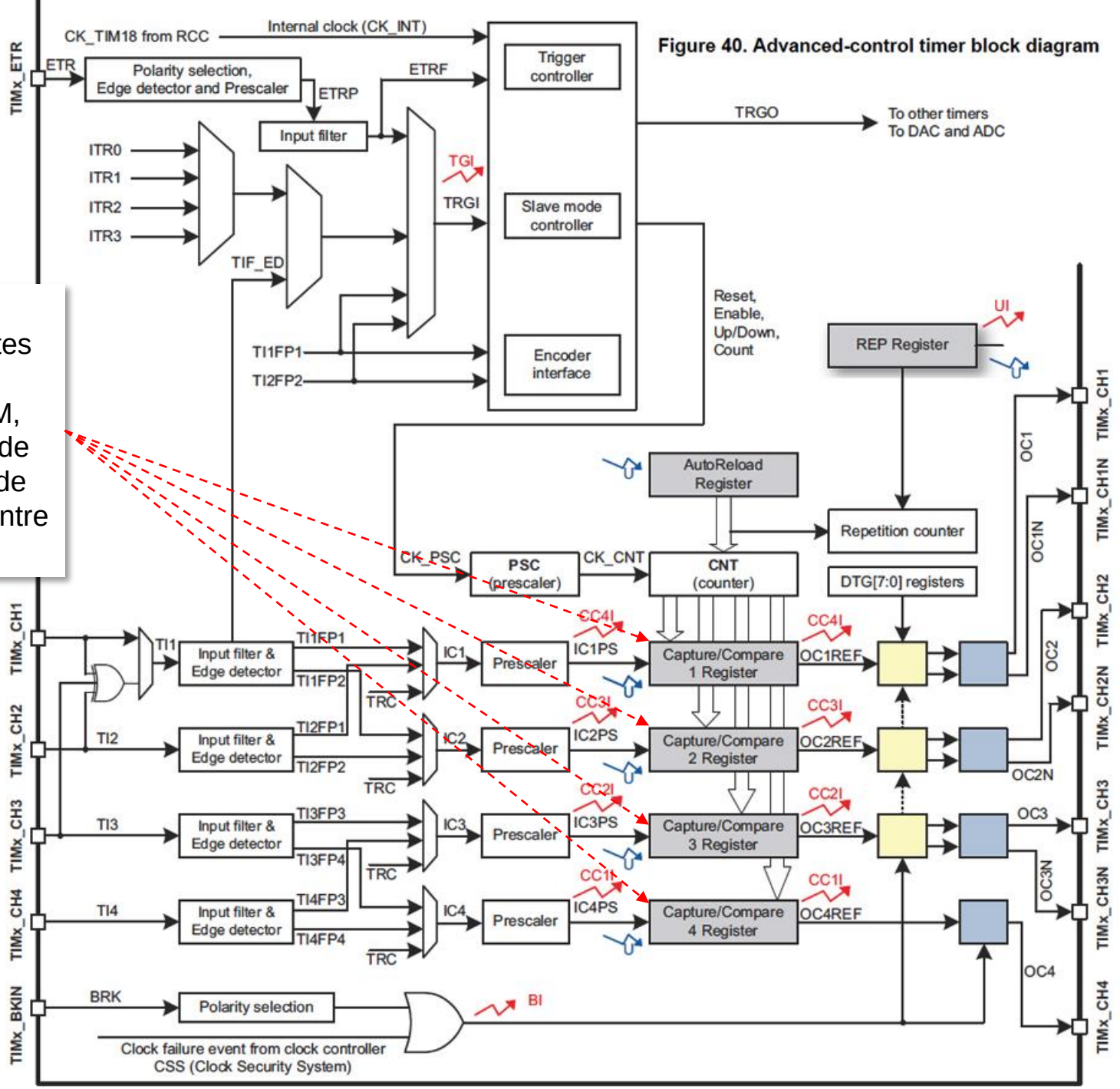
→ usado para gerar interrupções

→ usado para saída PWM (veremos PWM adiante...)

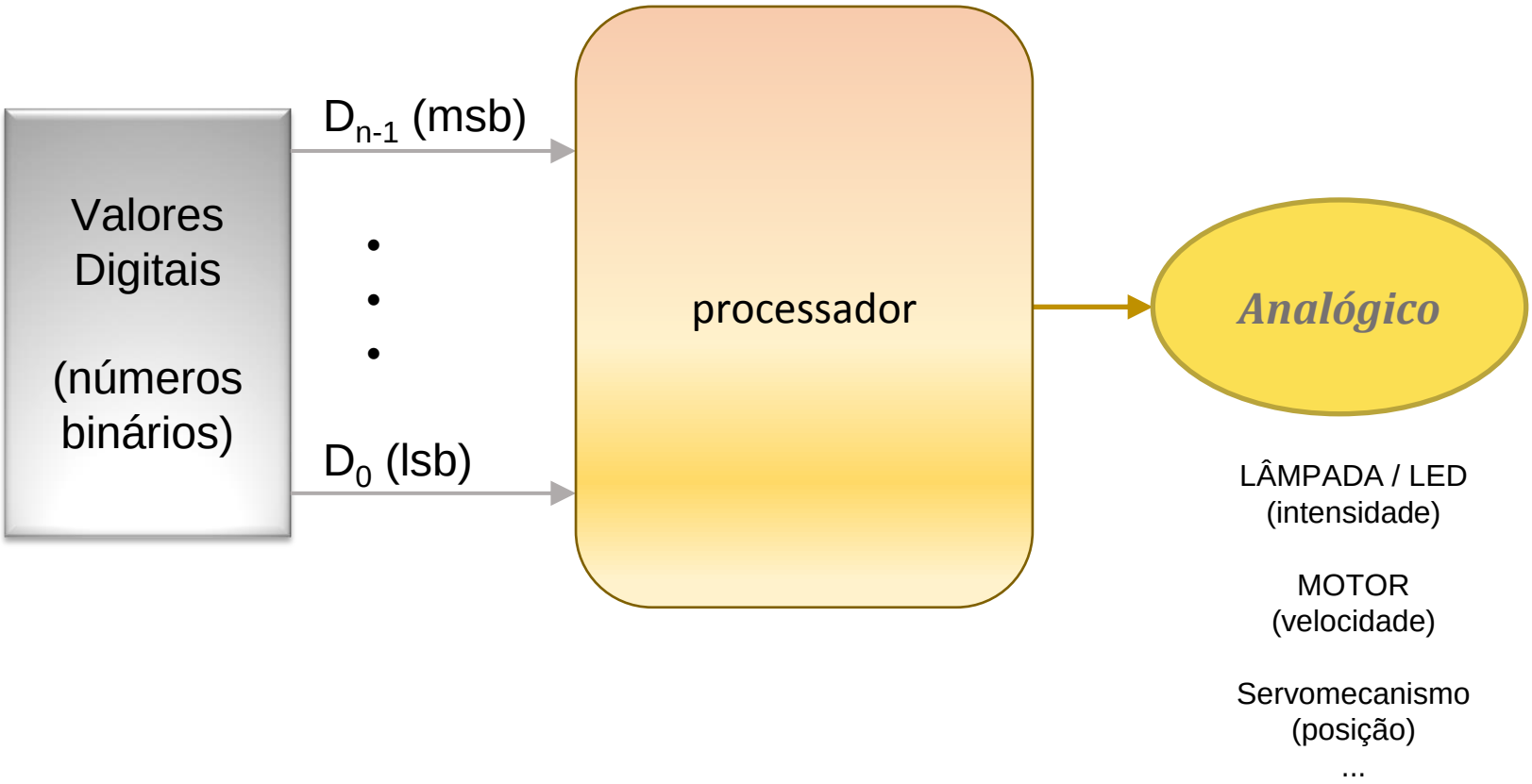
O que vimos no esquema geral
(slide anteriores)



Cada timer tem 4 canais independentes que podem fazer 4 comparações (PWM, int, etc) ou captura de tempo de um sinal de entrada (intervalo entre eventos, etc)...

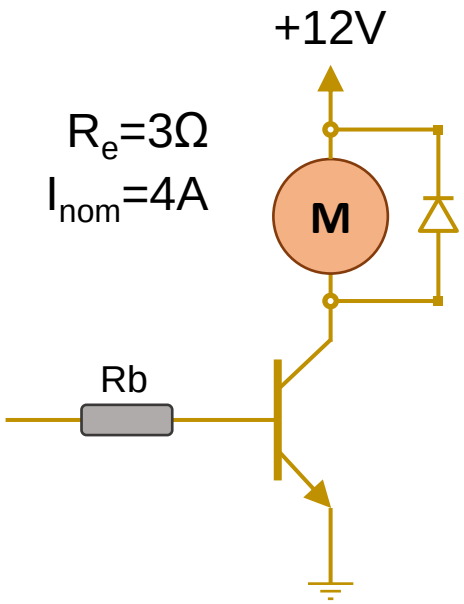


Problema que o controlador PWM resolve:



Problema que o controlador PWM resolve:

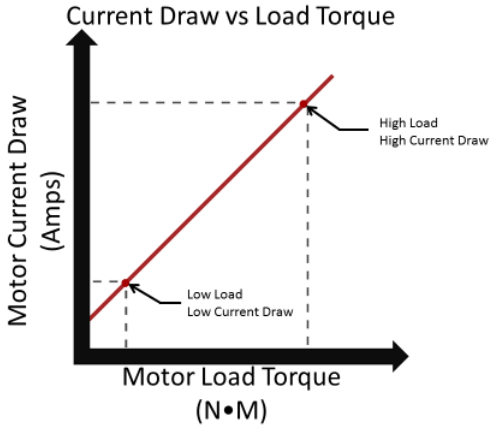
Prob: ½ da velocidade do motor...



Você conhece:

- Re = resistência equivalente do motor;
- Tensão nominal do motor;
- Corrente nominal do motor;

Pelas características do motor DC, se você usar ½ da corrente voê obtém ½ da velocidade...
(*em vazio – sem carga !!! – com carga muda ligeiramente, por isso o loop de controle*)



Curva típica IxT de um DC-Motor.

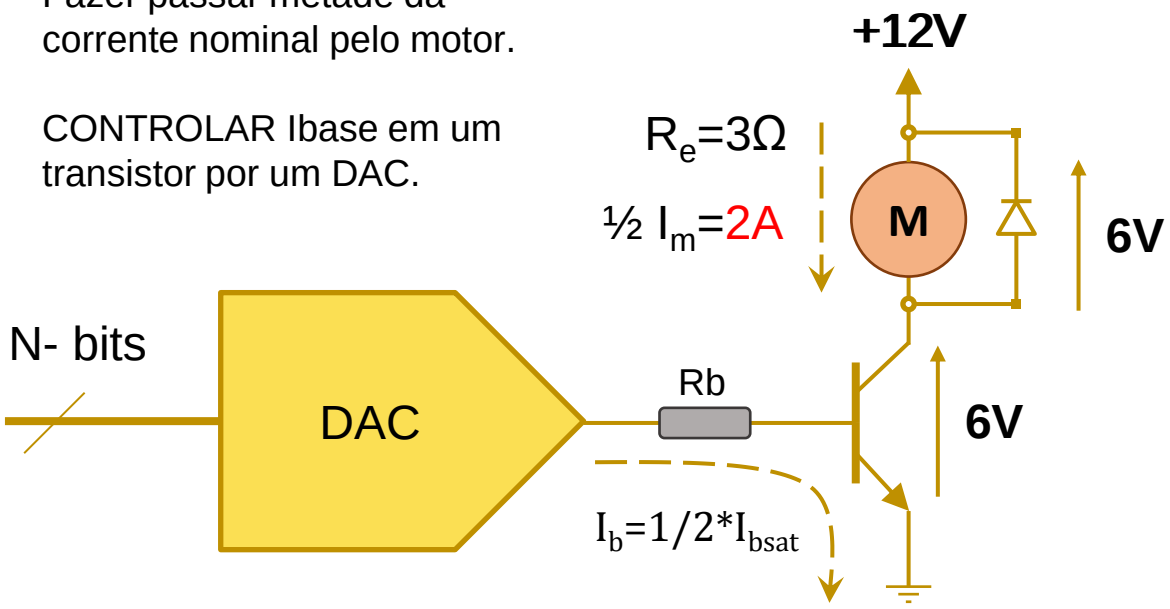
Problema que o controlador PWM resolve:

Solução 1:

Prob: ½ da velocidade do motor...

Fazer passar metade da corrente nominal pelo motor.

CONTROLAR Ibase em um transistor por um DAC.



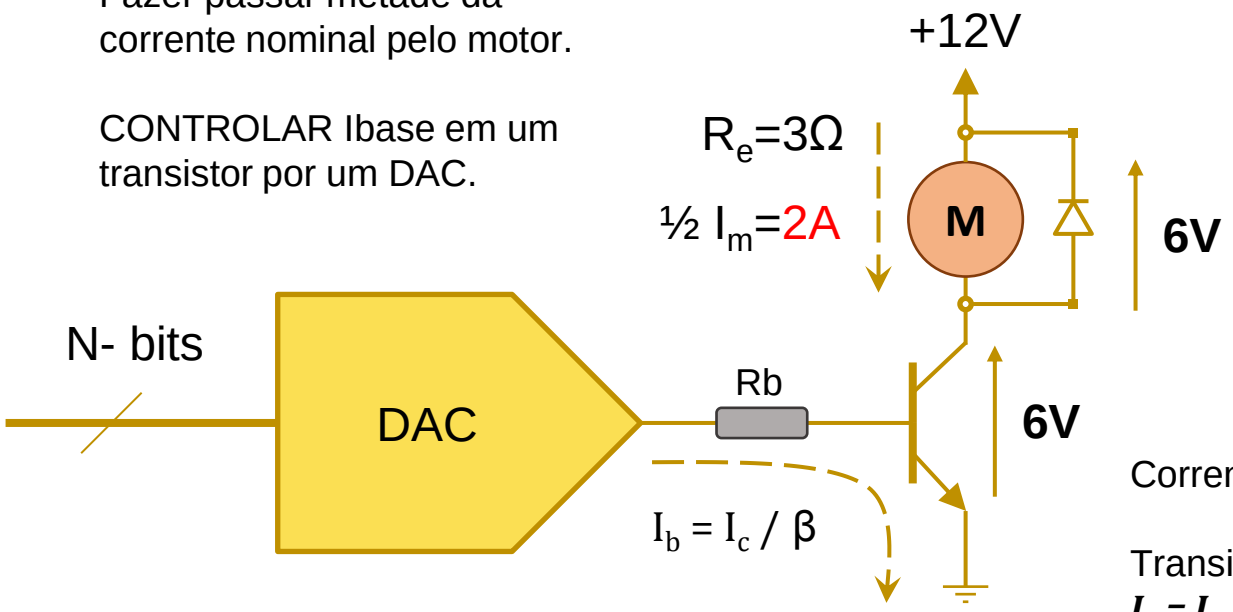
Problema que o controlador PWM resolve:

Solução 1:

Prob: ½ da velocidade do motor...

Fazer passar metade da corrente nominal pelo motor.

CONTROLAR Ibase em um transistor por um DAC.



Corrente de coletor = 2A

Transistor com β=100

$I_b = I_c / \beta$

Valor do resistor de base:

$Rb = (V_{DAC} - 0{,}7) / I_b$

Potência (contínua) dissipada no transistor:

$Pt = 6V * 2A = 12W$

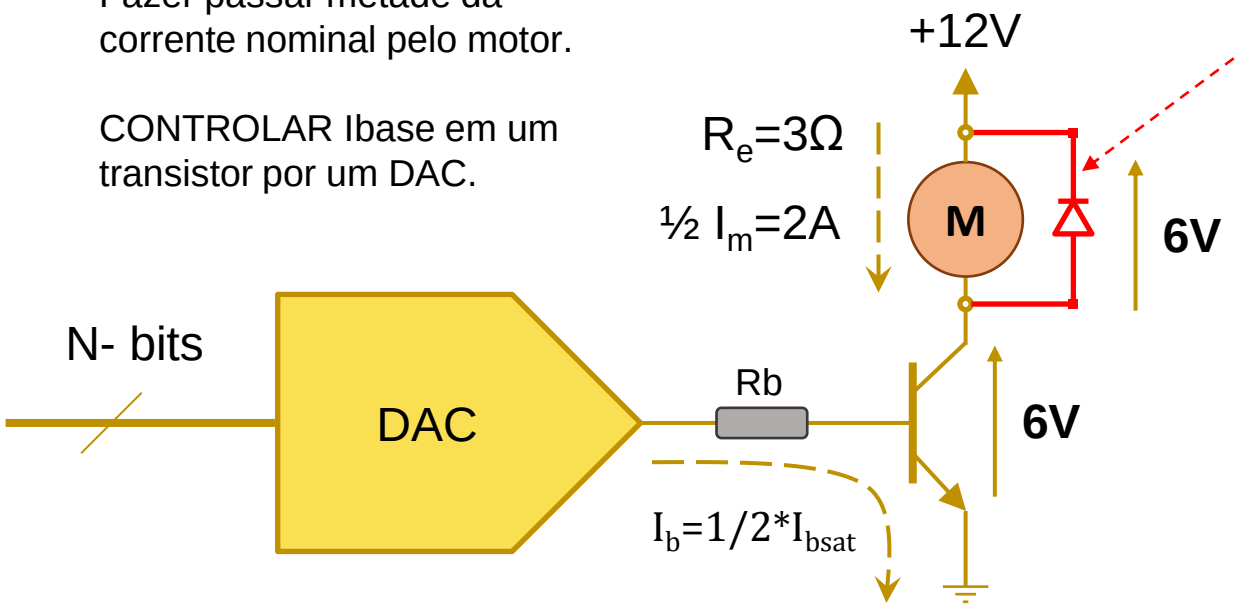
Problema que o controlador PWM resolve:

Solução 1:

Prob: ½ da velocidade do motor...

Fazer passar metade da corrente nominal pelo motor.

CONTROLAR Ibase em um transistor por um DAC.



Para que esse diodo ???

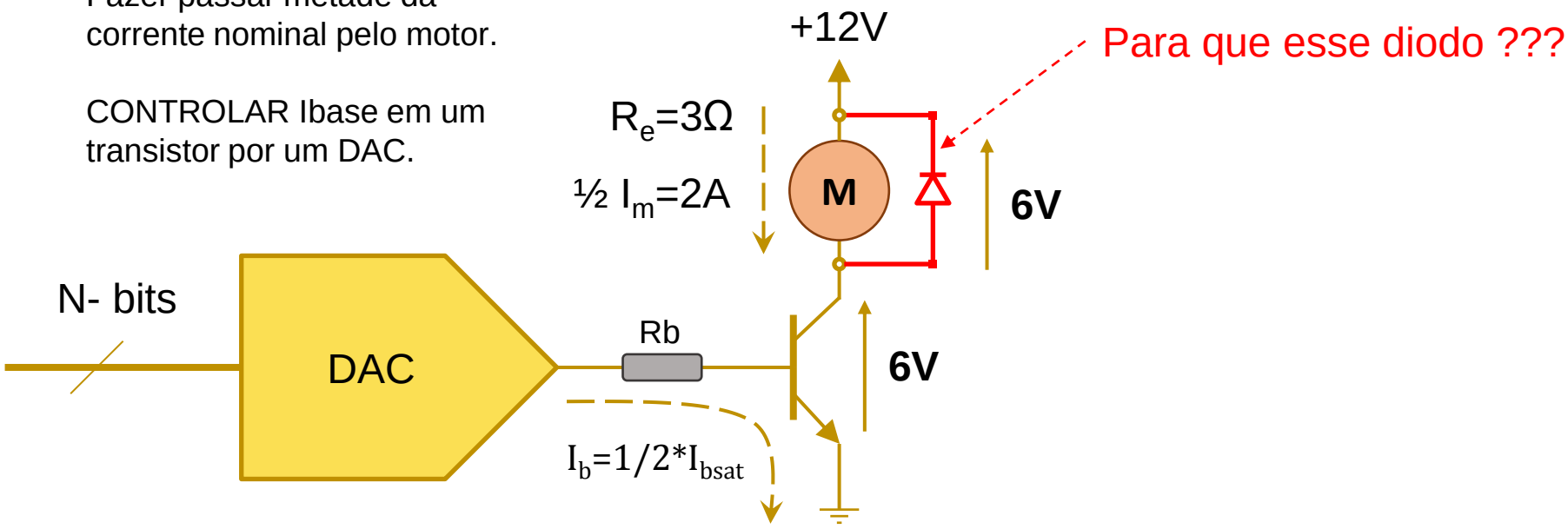
Problema que o controlador PWM resolve:

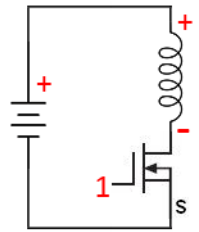
Solução 1:

Prob: ½ da velocidade do motor...

Fazer passar metade da corrente nominal pelo motor.

CONTROLAR Ibase em um transistor por um DAC.





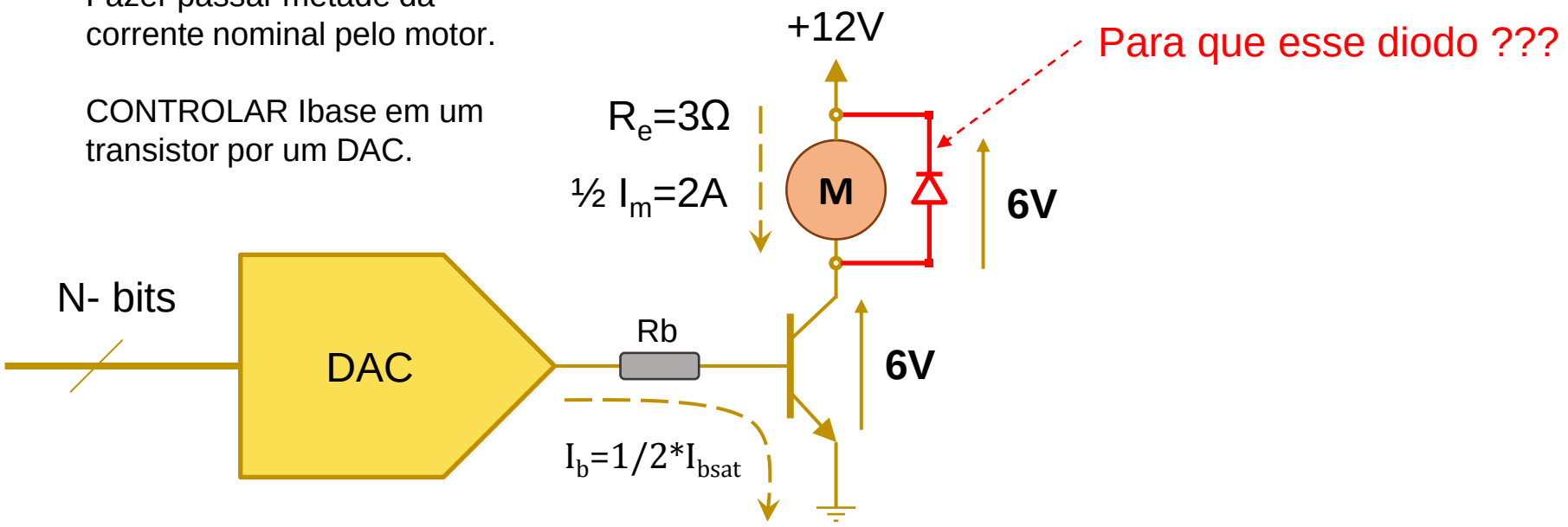
Problema que o controlador PWM resolve:

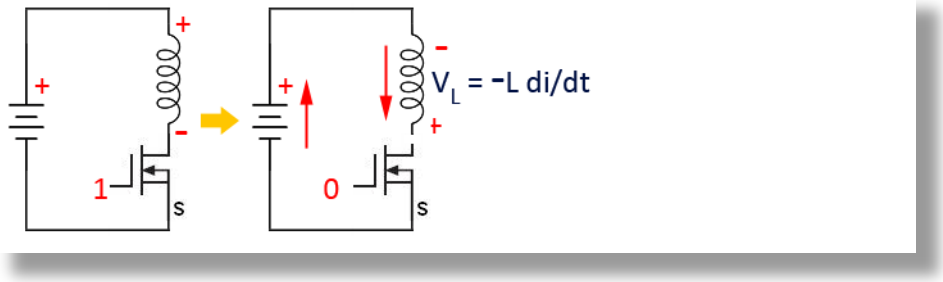
Solução 1:

Prob: ½ da velocidade do motor...

Fazer passar metade da corrente nominal pelo motor.

CONTROLAR Ibase em um transistor por um DAC.





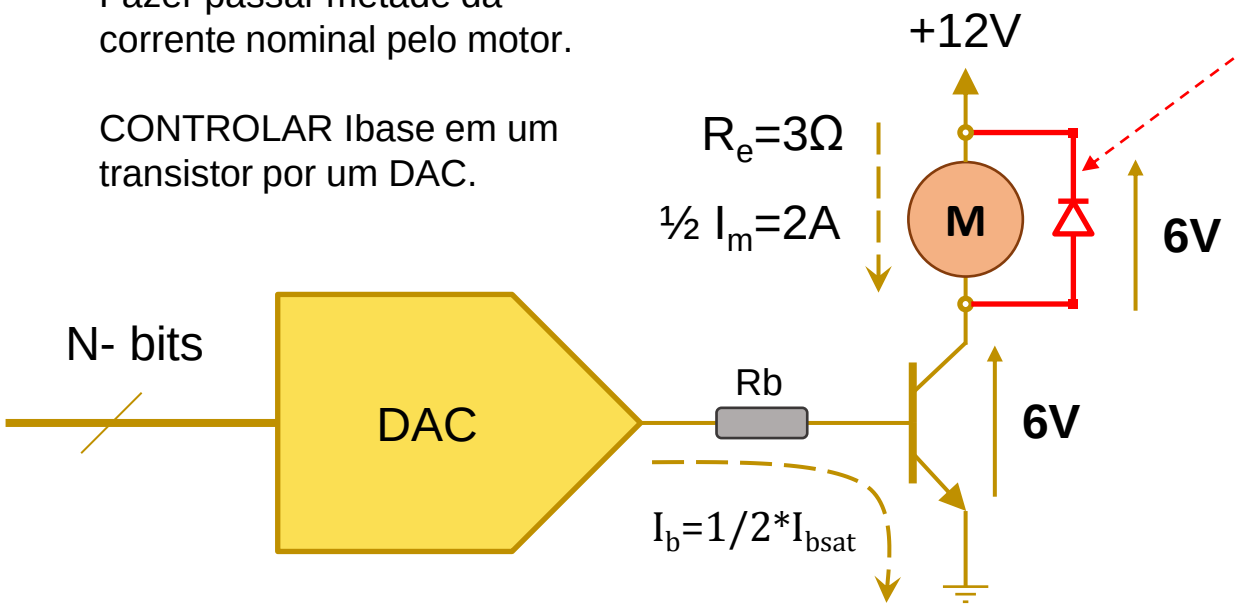
Problema que o controlador PWM resolve:

Solução 1:

Prob: ½ da velocidade do motor...

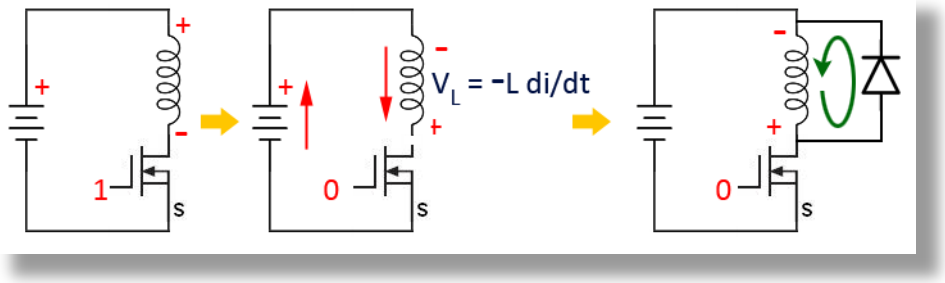
Fazer passar metade da corrente nominal pelo motor.

CONTROLAR Ibase em um transistor por um DAC.



Para que esse diodo ???

- R: O diodo serve para manter a corrente ***I*** no indutor depois do transistor entrar em corte. Isso dissipa a energia magnética acumulada, evitando uma tensão reversa no coletor do transistor que pode queimá-lo.



Problema que controlado /M resolve:

Solução 1:

elocidade do motor...

Fazer passar metade da
corrente nominal

CONTROLAR
transistor por um D

e diodo ???

O Prof. Ranhel avisou...
DIODO DE PROTEÇÃO!

(não usar diodo de proteção em carga indutiva queima o μC/SoC ou transistor de saída...)

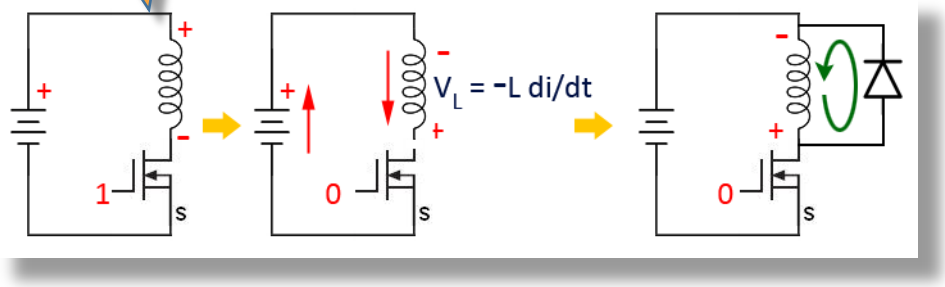
Agora , é por sua conta!!!

N- bits

DAC

I_b=1/2

ve para manter a
rrente ***I*** no indutor depois do
nsistor entrar em corte. Isso
ipa a energia magnética
nulada, evitando uma
te o reversa no coletor do
transistor que pode queimá-lo.



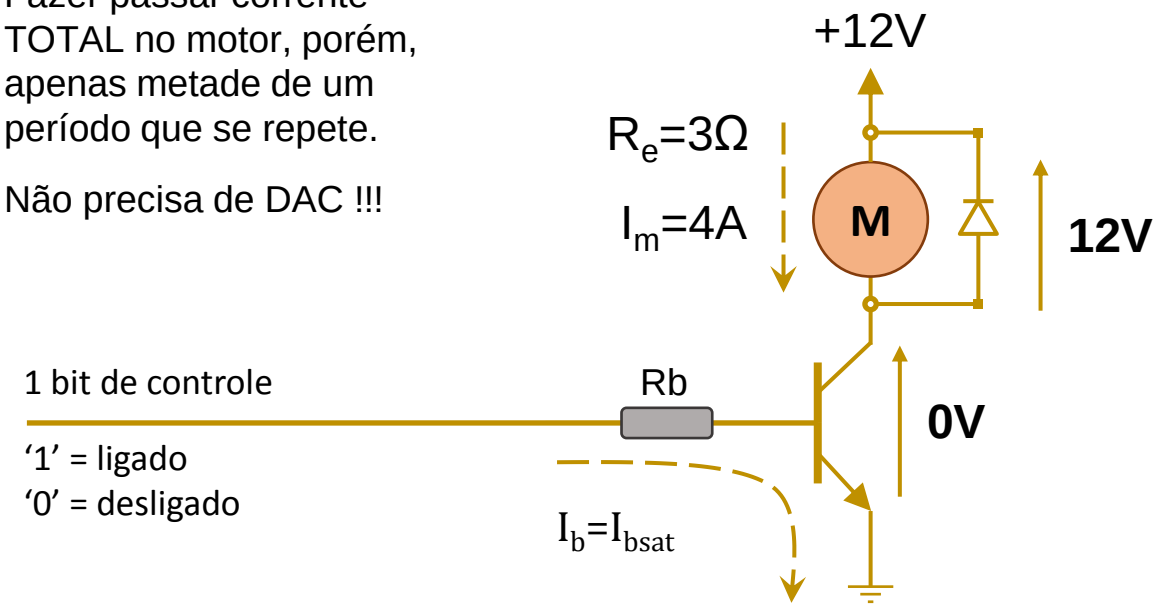
Problema que o controlador PWM resolve:

Solução 2:

Prob: ½ da velocidade do motor...

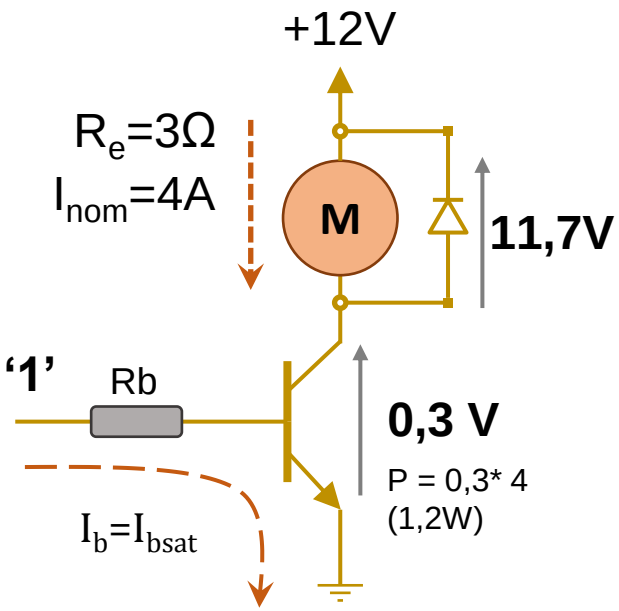
Fazer passar corrente TOTAL no motor, porém, apenas metade de um período que se repete.

Não precisa de DAC !!!



Problema que o controlador PWM resolve:

Solução 2:



Prob: ½ da velocidade do motor...

Controlar o ***tempo*** entre ativação / desativação do transistor

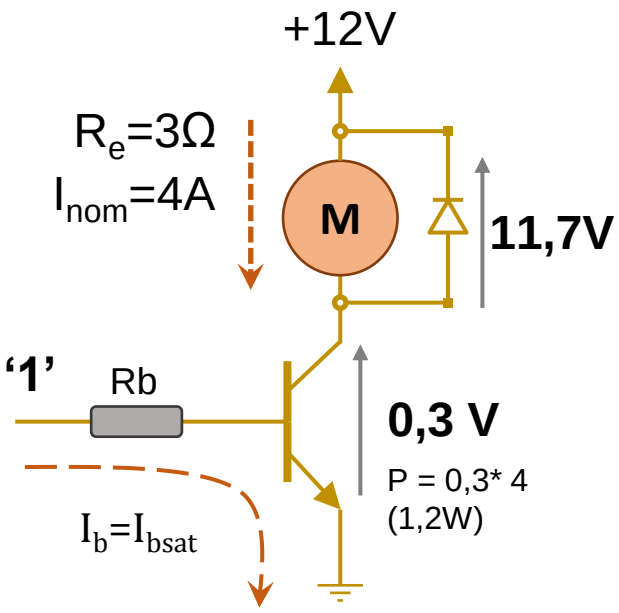
- Usar frequência relativamente alta;

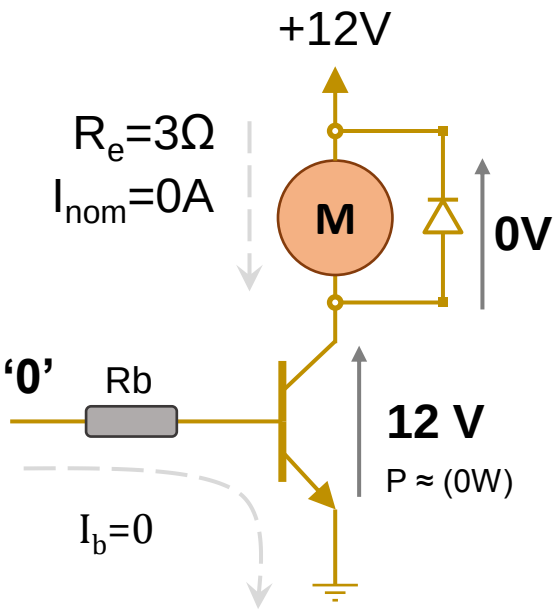
(τ pequeno em relação ao fenômeno físico)

- Inercia do fenômeno físico filtra essa frequência alta.

Problema que o controlador PWM resolve:

Solução 2:





Prob: ½ da velocidade do motor...

Controlar o ***tempo*** entre ativação / desativação do transistor

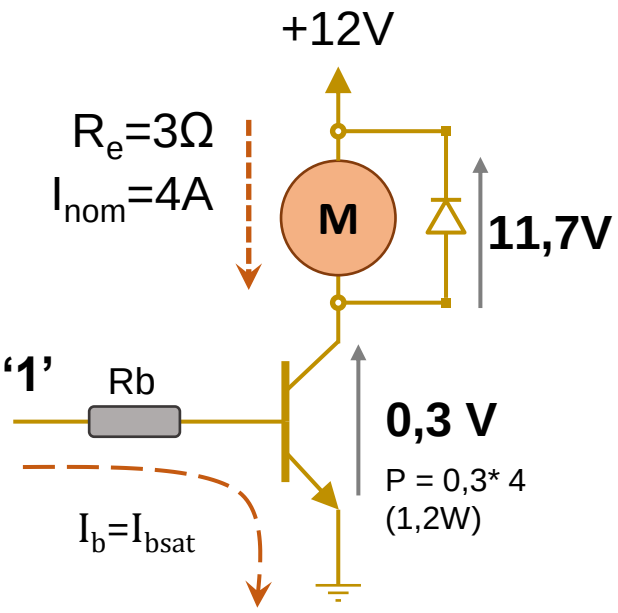
- Usar frequência relativamente alta;

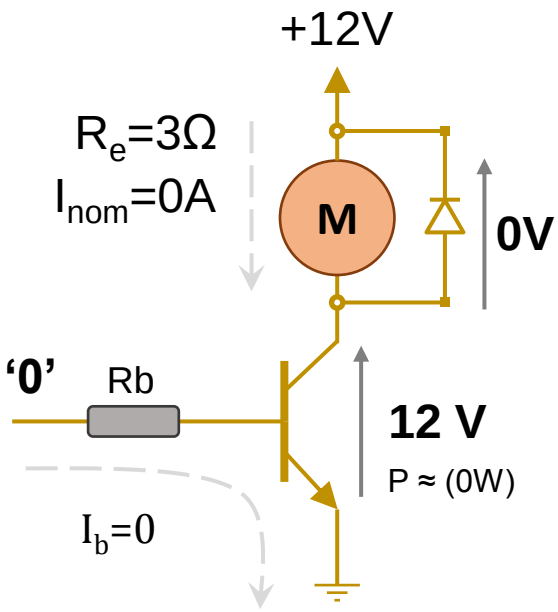
(τ pequeno em relação ao fenômeno físico)

- Inercia do fenômeno físico filtra essa frequência alta.

Problema que o controlador PWM resolve:

Solução 2:





Prob: ½ da velocidade do motor...

Controlar o ***tempo*** entre ativação / desativação do transistor

- Usar frequência relativamente alta;

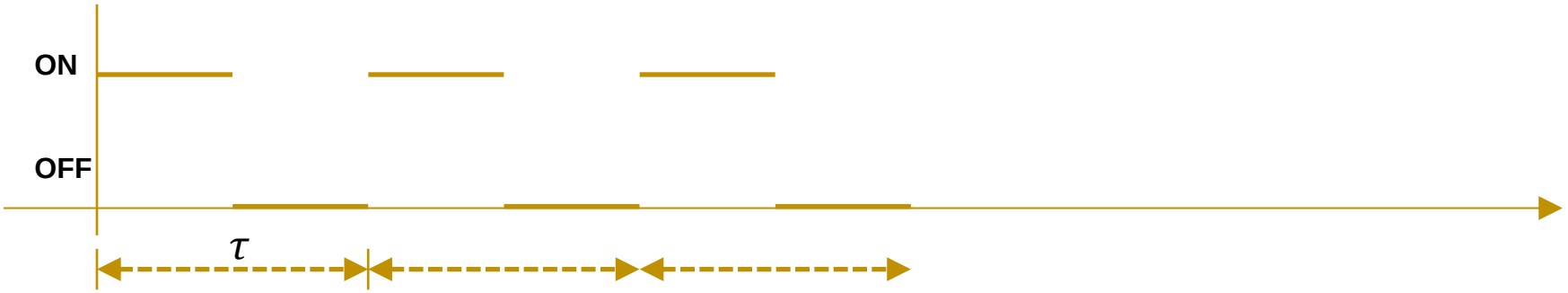
(τ pequeno em relação ao fenômeno físico)

- Inercia do fenômeno físico filtra essa frequência alta.

$$F_{op} = 1/\tau$$

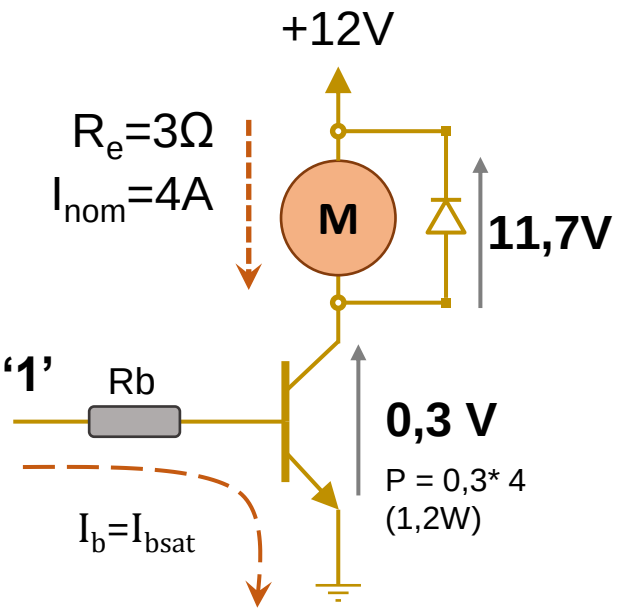
$$P_{(t)} = \int_{tn}^{tn+1} V.I\ dt$$

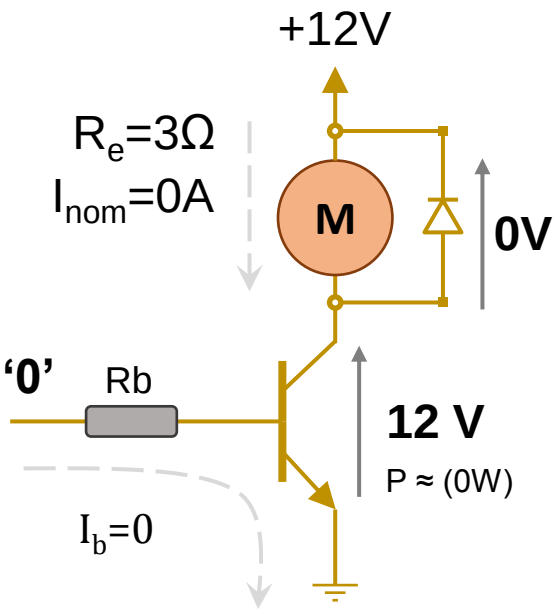
Nesse caso: (1,2)/2 = 0,6W
(20 vezes menos que a solução 1) !!!



Problema que o controlador PWM resolve:

Solução 2:





Prob: ½ da velocidade do motor...

Controlar o ***tempo*** entre ativação / desativação do transistor

- Usar frequência relativamente alta;

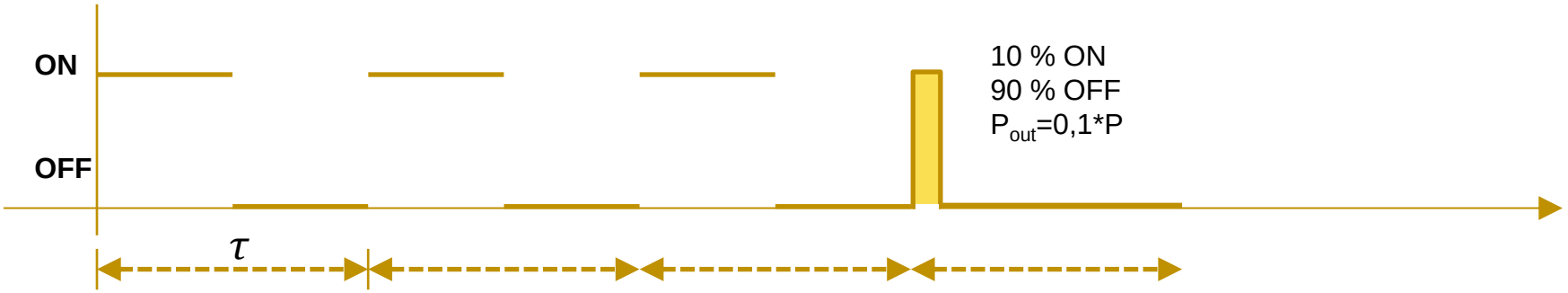
(τ pequeno em relação ao fenômeno físico)

- Inercia do fenômeno físico filtra essa frequência alta.

$$F_{op} = 1/\tau$$

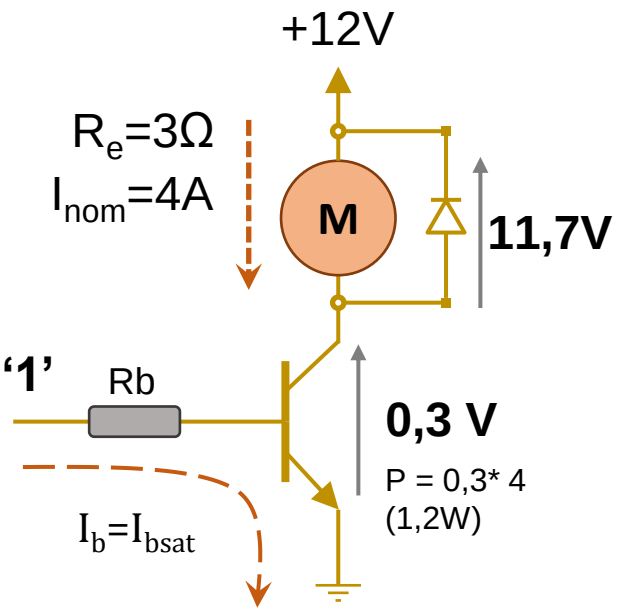
$$P_{(t)} = \int_{tn}^{tn+1} V.I\ dt$$

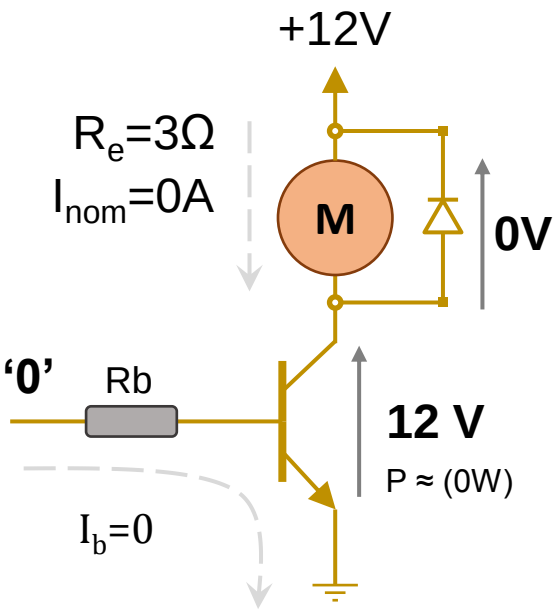
Nesse caso: (1,2)/2 = 0,6W
(20 vezes menos que a solução 1) !!!



Problema que o controlador PWM resolve:

Solução 2:





Prob: ½ da velocidade do motor...

Controlar o ***tempo*** entre ativação / desativação do transistor

- Usar frequência relativamente alta;

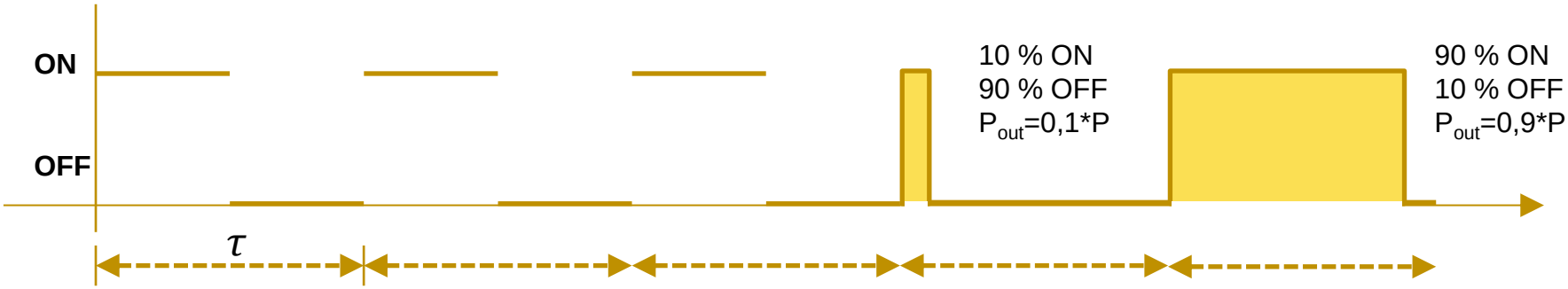
(τ pequeno em relação ao fenômeno físico)

- Inercia do fenômeno físico filtra essa frequência alta.

$$F_{op} = 1/\tau$$

$$P_{(t)} = \int_{tn}^{tn+1} V.I\ dt$$

Nesse caso: (1,2)/2 = 0,6W
(20 vezes menos que a solução 1) !!!



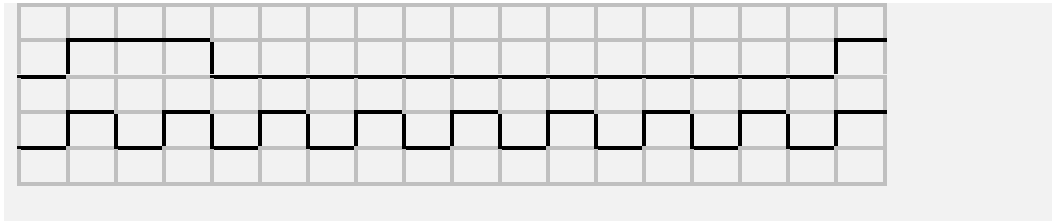
PWM tem sido usado como ***saída analógica*** em SoC e em μControladores.

PWM: *Gera sinais com relação temporal entre níveis lógicos '1' e '0' controlados e bem definidos.*

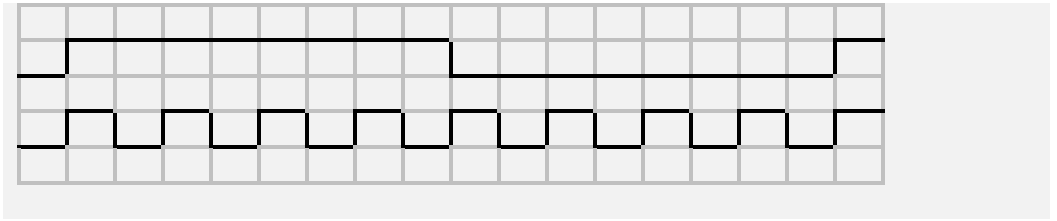
Ciclo de trabalho (duty cycle): % tempo do sinal 'alto' (1) em relação ao sinal 'baixo' (0)
Onda quadrada: ciclo de 50%

Usos comuns: controlar tensão média para o dispositivo elétrico:
- *Mais simples que conversor DC-DC ou conversor D-A;*
- *Velocidade de motor DC, dimmers*

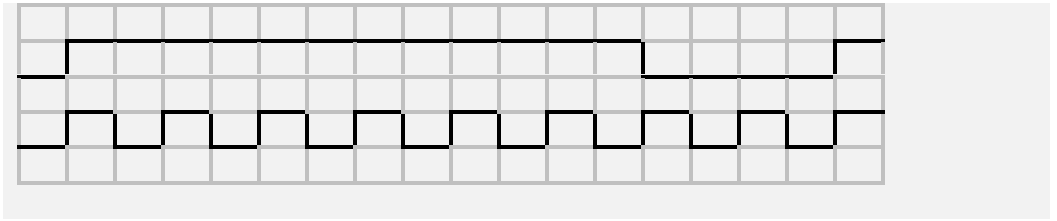
Outro uso: comandos codificados temporalmente, onde o receptor usa temporizador para decodificar sinal recebido.



25% duty cycle – average pwm_o is 1.25V

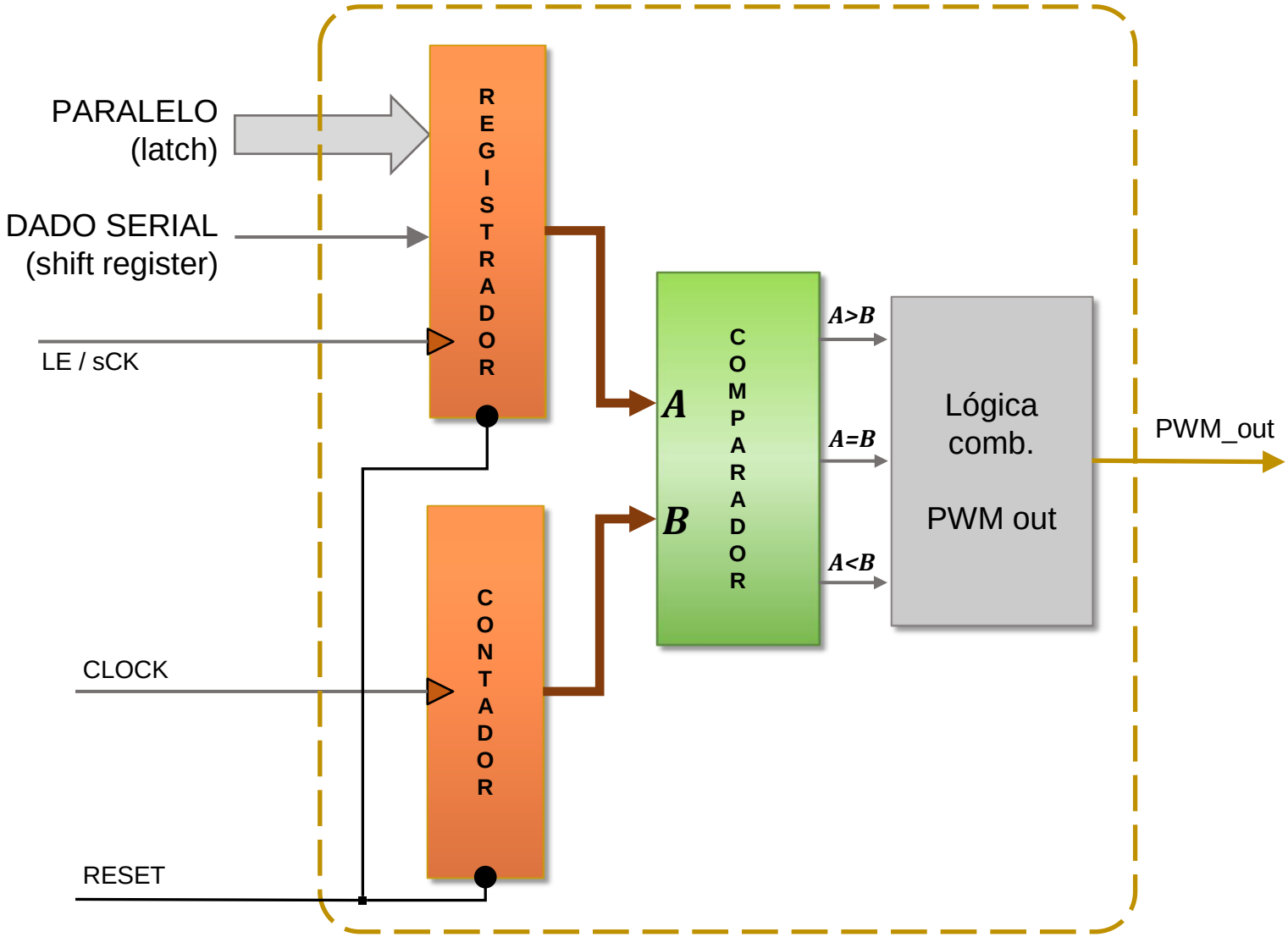


50% duty cycle – average pwm_o is 2.5V.

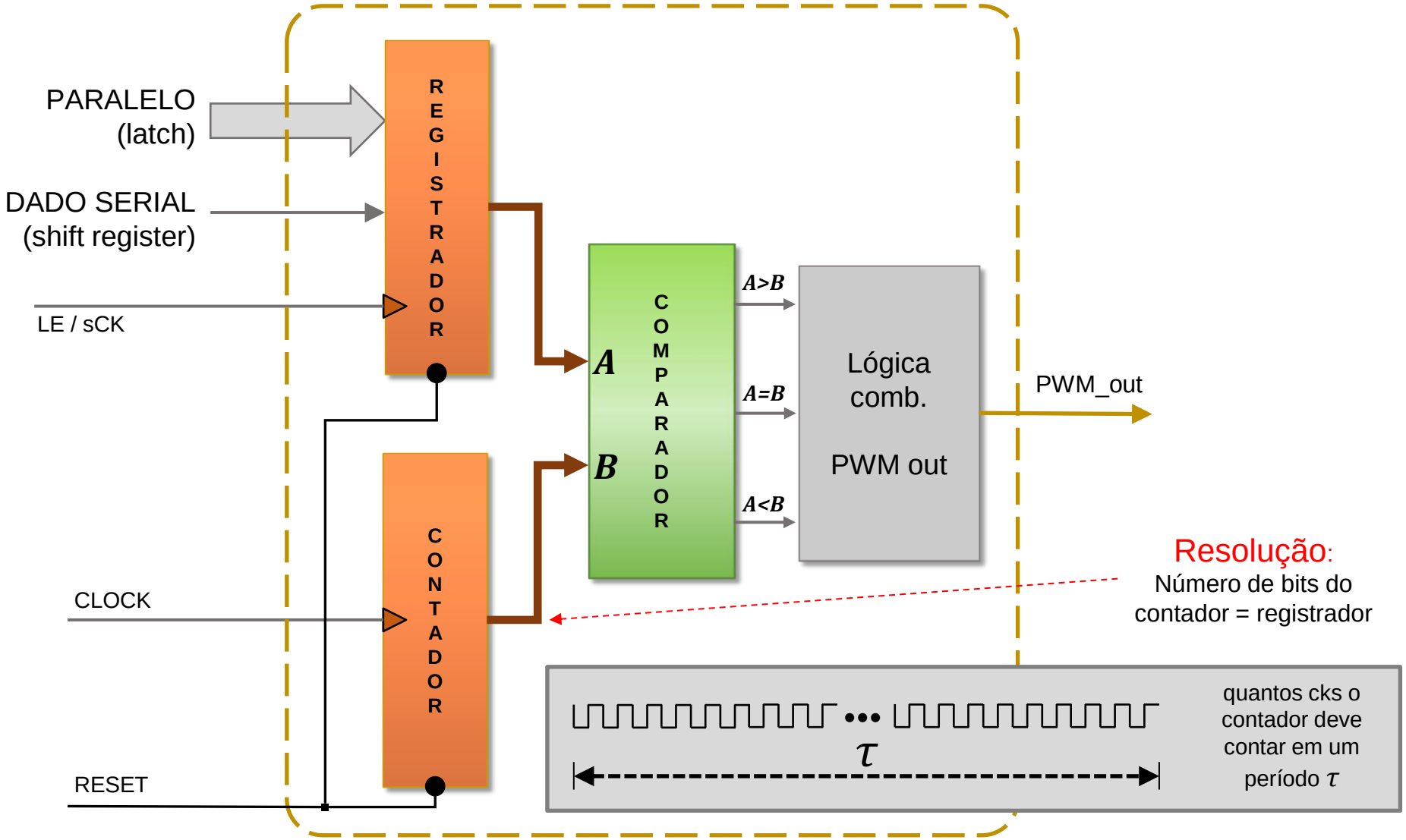


75% duty cycle – average pwm_o is 3.75V.

Construindo um *PWM Digital* – (processador pwm)



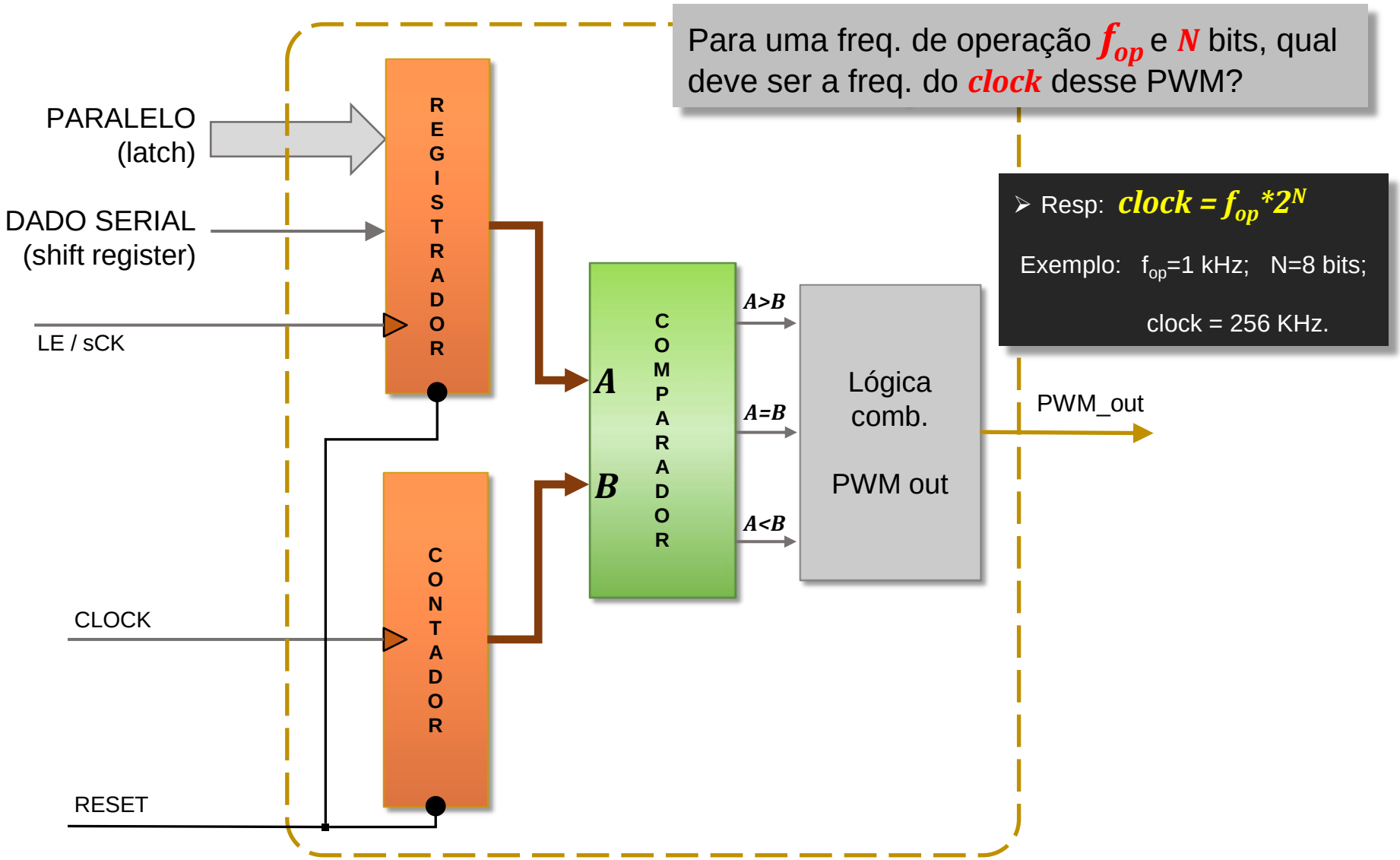
Construindo um *PWM Digital* – (processador pwm)



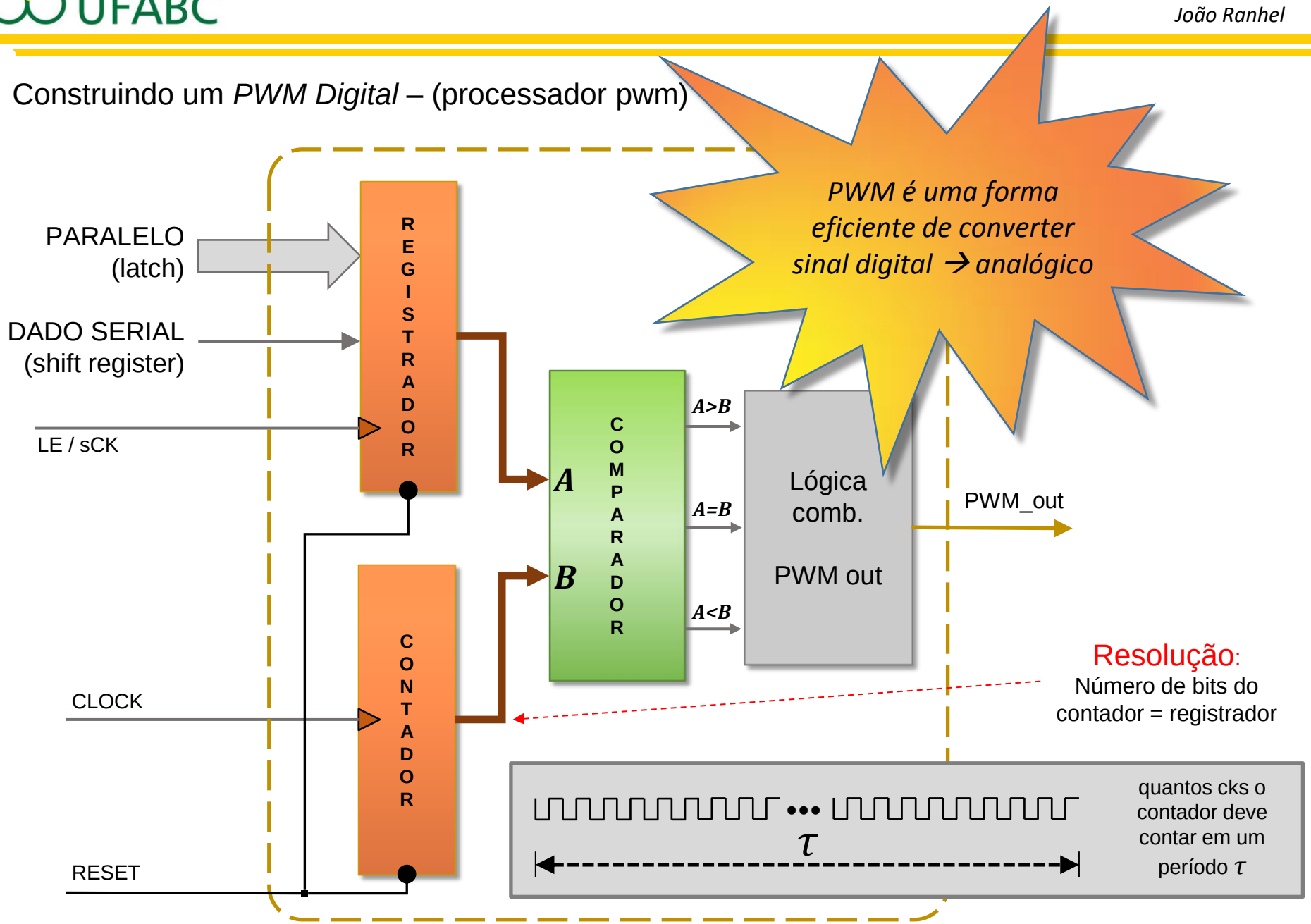
Construindo um *PWM Digital* – (processador pwm)

Para uma freq. de operação f_{op} e N bits, qual deve ser a freq. do ***clock*** desse PWM?

Construindo um *PWM Digital* – (processador pwm)



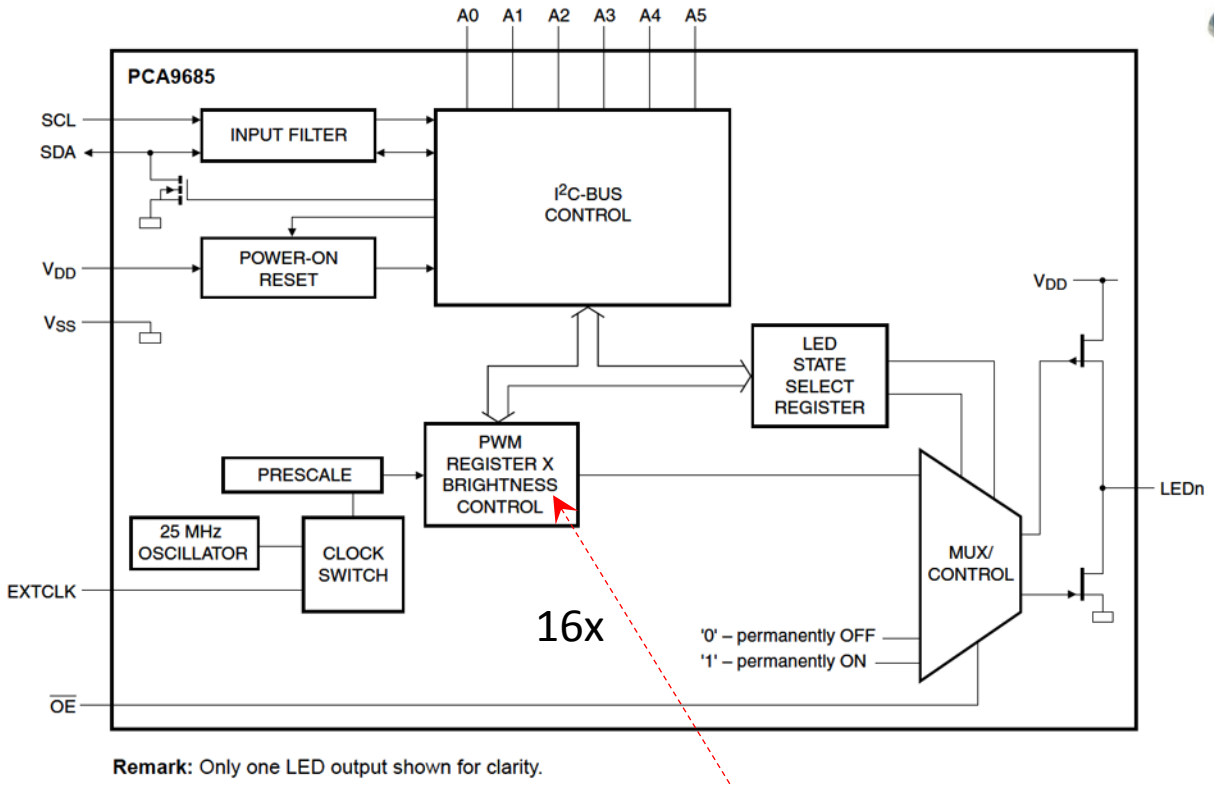
Construindo um *PWM Digital* – (processador pwm)

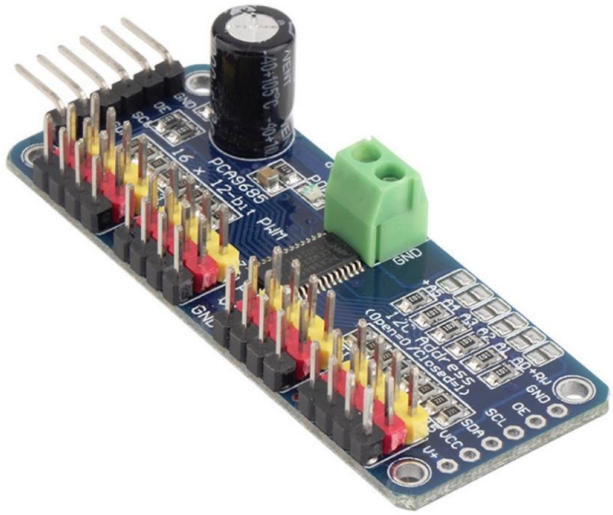


PCA-9685:

16-channel, 12-bit PWM Fm+ I²C-bus LED controller

(custo Digikey: US$ 1,06 p/ 2500 peças)

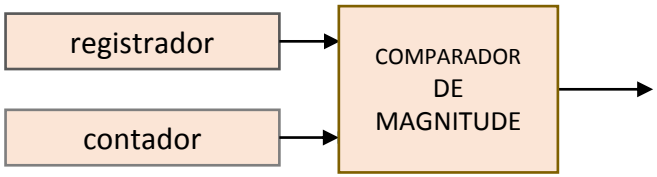


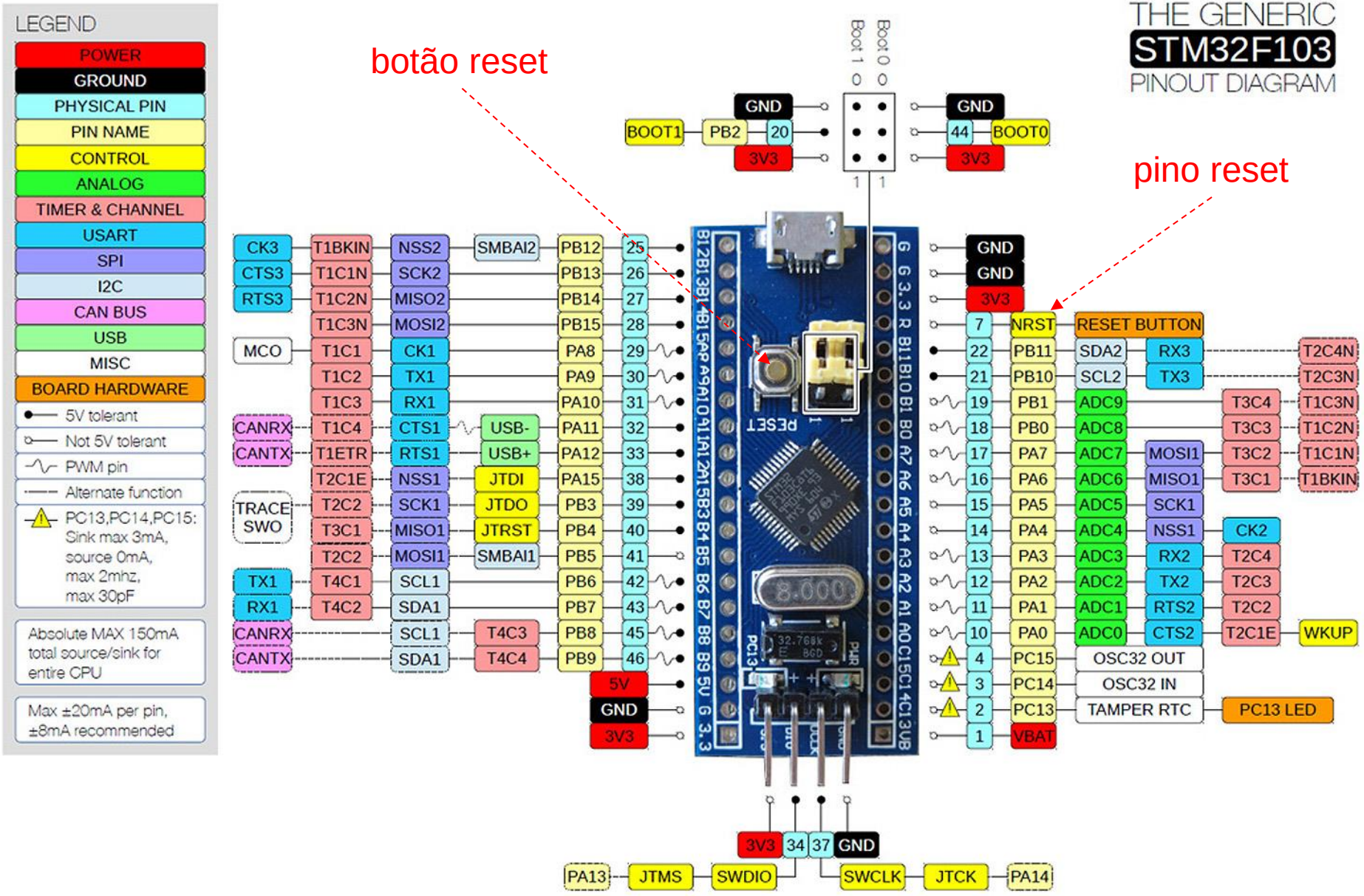


Placa Adafruit:

(custo: ~US$ 15 por placa)

(genérica chinesa: US$ 8)

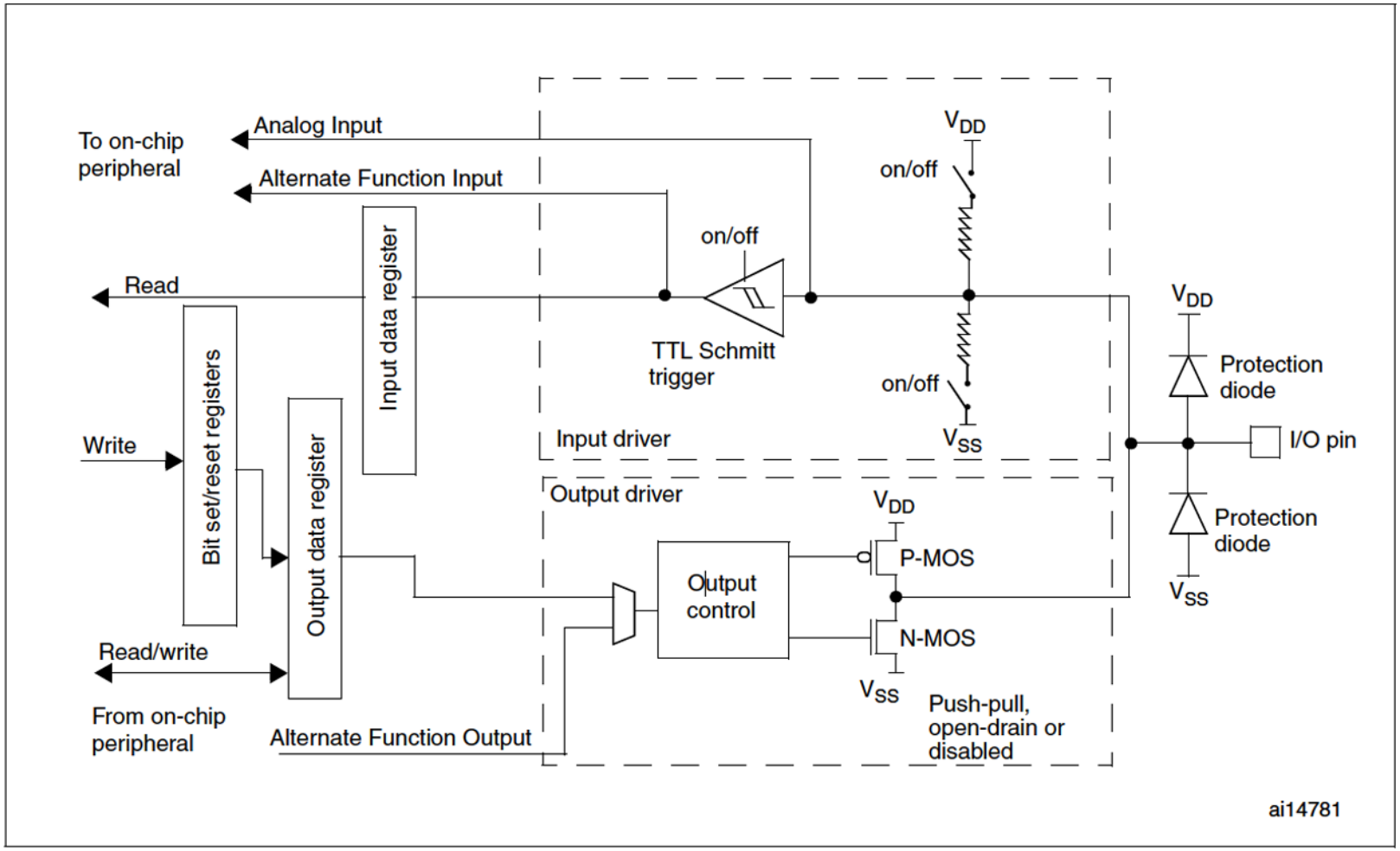



LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
botão reset
pino reset
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
GND
BOOT1
PB2
20
3V3
44
BOOT0
RESET
8.000
32.768k
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
CK2
T2C4
T2C3
T2C2
T2C1E
WKUP
PC13 LED
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14

- Como μControladores têm mais funções que pinos de I/O os fabricantes atribuem mais de uma função a cada pino. A funcionalidade do pino deve ser programada.

- Como μControladores têm mais funções que pinos de I/O os fabricantes atribuem mais de uma função a cada pino. A funcionalidade do pino deve ser programada.

Figure 13. Basic structure of a standard I/O port bit



- Como μControladores têm mais funções que pinos de I/O os fabricantes atribuem mais de uma função a cada pino. A funcionalidade do pino deve ser programada.

Cada pino de GPIO no ARM Cortex pode ser configurado nos modos:

- Input floating
- Input pull-up
- Input-pull-down

- (Input) Analog

- Output open-drain
- Output push-pull

- Alternate function push-pull
- Alternate function open-drain

- Como μControladores têm mais funções que pinos de I/O os fabricantes atribuem mais de uma função a cada pino. A funcionalidade do pino deve ser programada.

Cada pino de GPIO no ARM Cortex pode ser configurado nos modos:

- Input floating
- Input pull-up
- Input-pull-down

- (Input) Analog

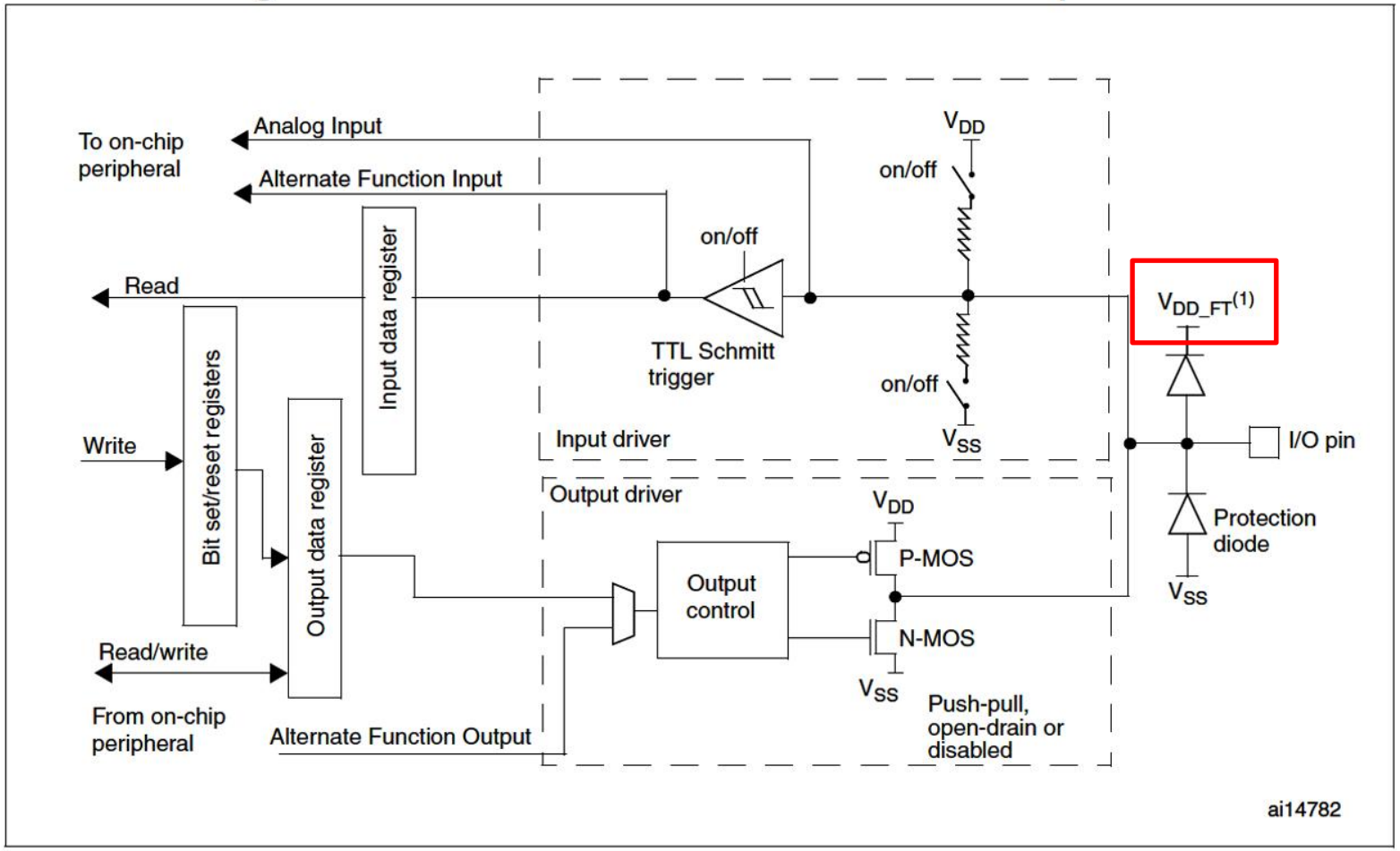
- Output open-drain
- Output push-pull

- Alternate function push-pull
- Alternate function open-drain

- ✓ Cada bit de porta de I/O é livremente programável, porém, os registros de porta de I/O devem ser acessados como palavras de 32 bits (*não são permitidos acessos de meia palavra ou de byte*).
- ✓ Durante e logo após o reset as funções alternativas ficam INATIVAS e as portas de I/O são configuradas no modo de *entrada flutuante.*

- Alguns pinos do CI são tolerantes à excitação de 5V (lógica de 5V)...

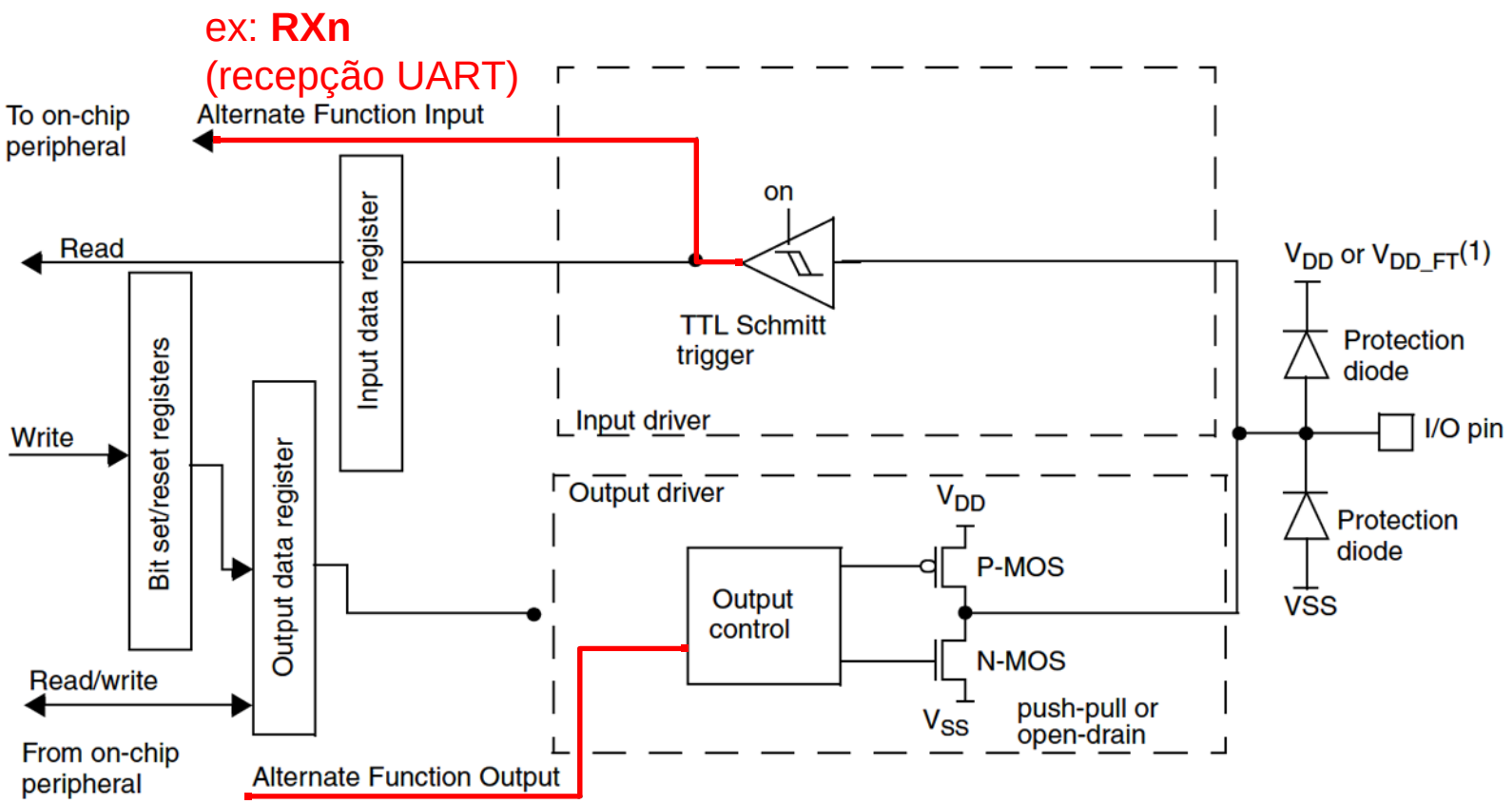
Figure 14. Basic structure of a five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

- Quando os pinos não são usados como GPIO devem ser configurados p/ modo de funções alternativas (exemplo: saída PWM – referenciada como **AFIOs**)

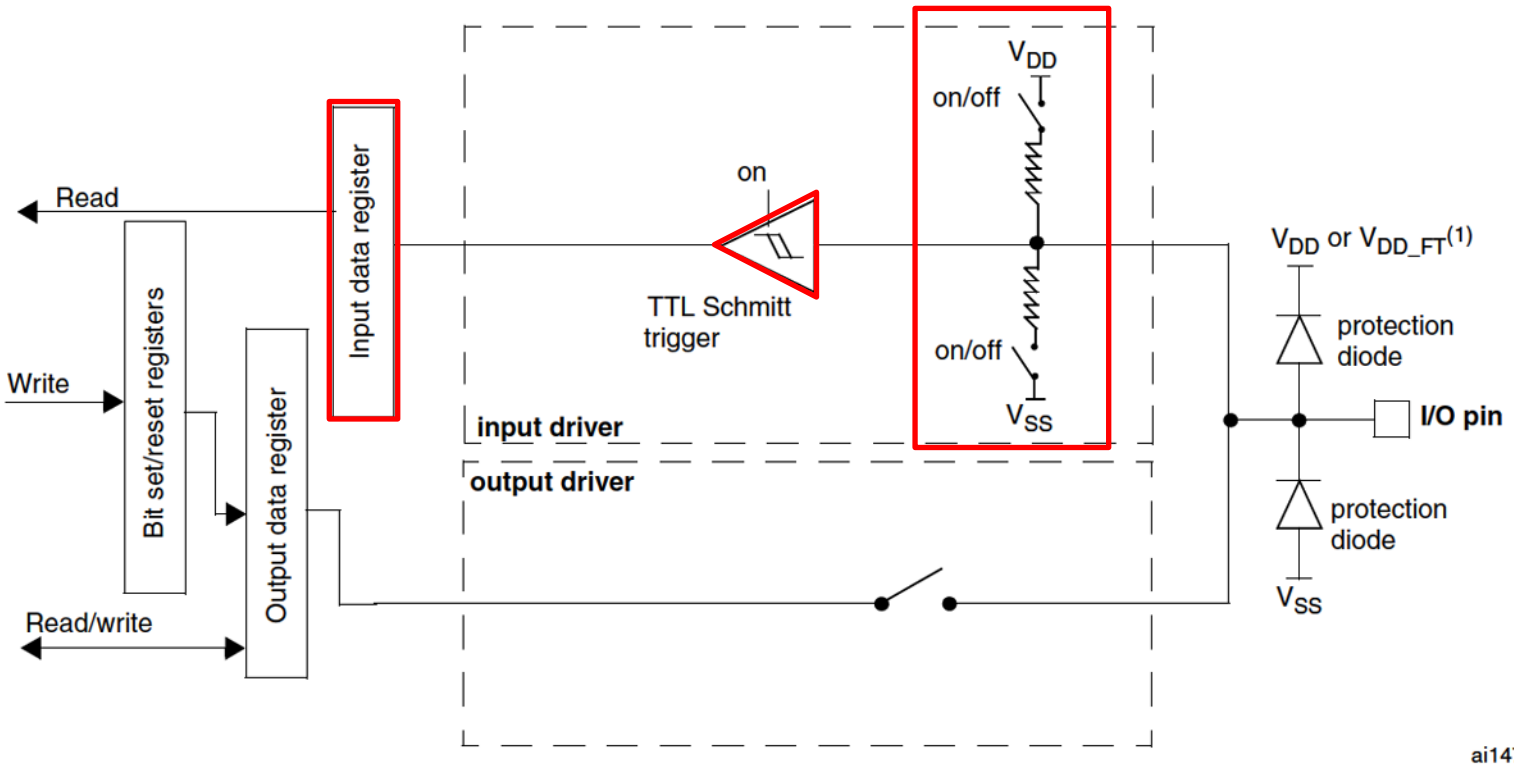
Figure 17. Alternate function configuration



ex: **PWM**
(um dos 4 canais de um TIMERx)

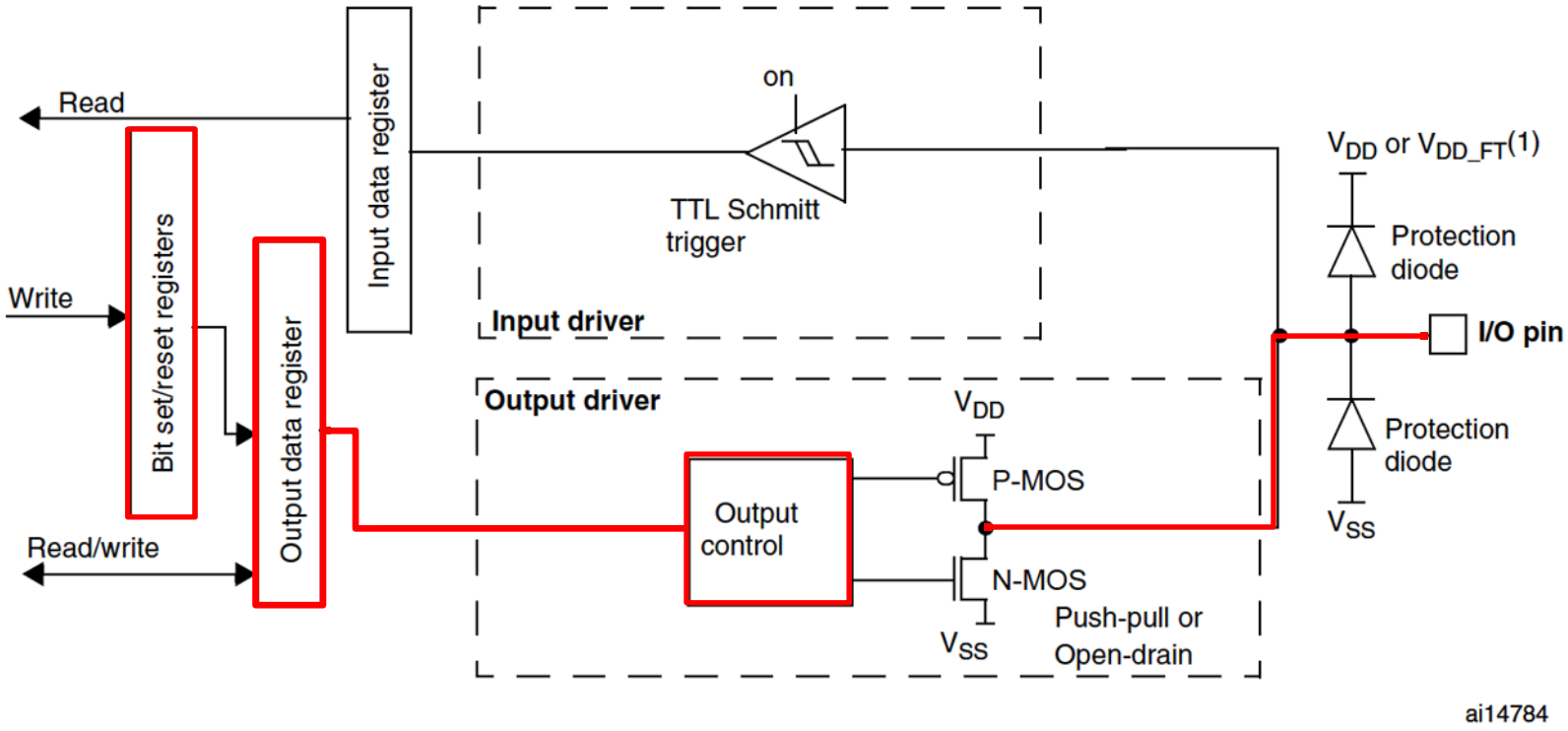
- Esta é uma estrutura básica para INPUT digital...

Figure 15. Input floating/pull up/pull down configurations



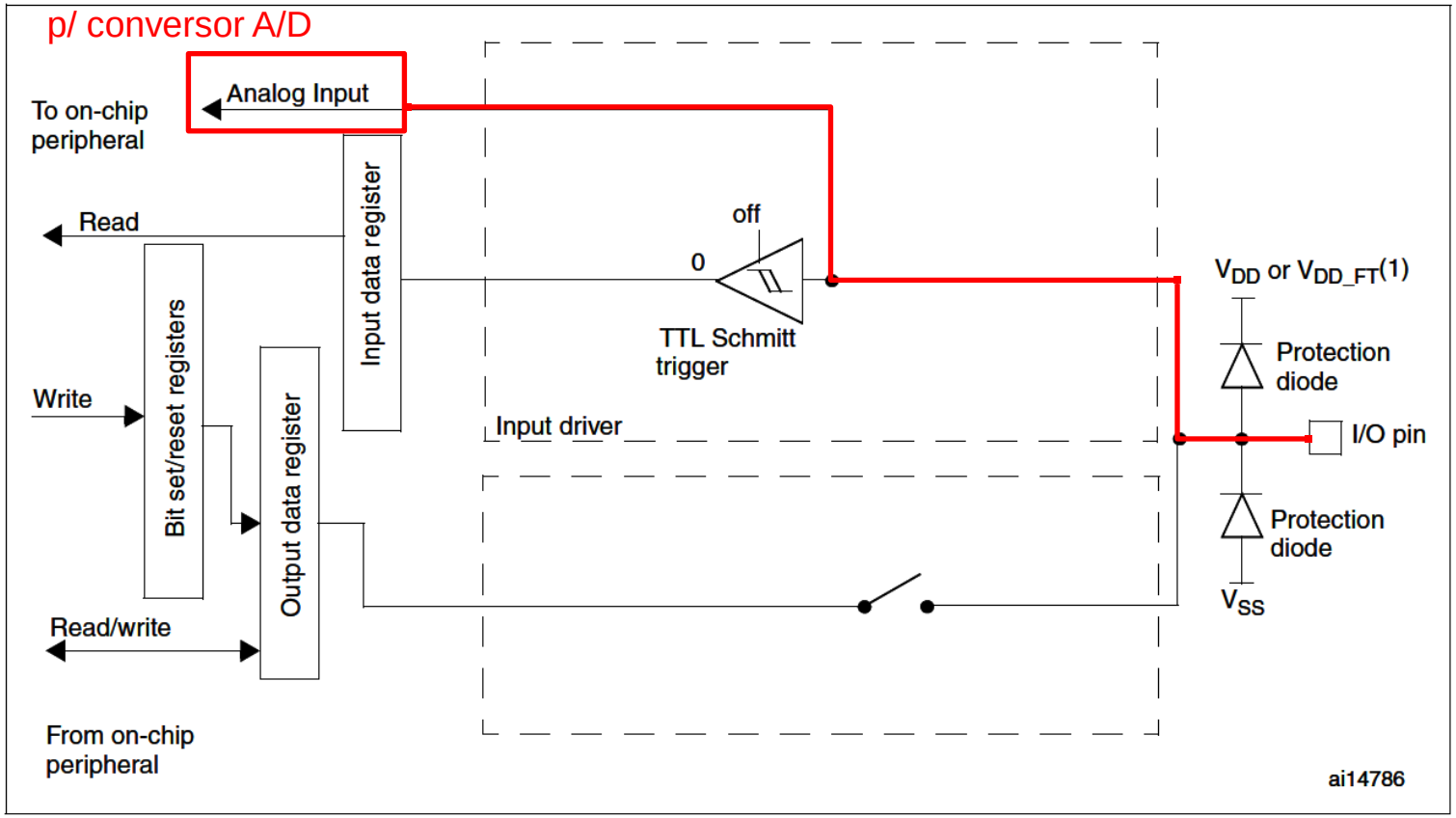
- Esta é uma estrutura básica para OUTPUT digital...

Figure 16. Output configuration



- Esta é uma estrutura básica para OUTPUT digital...

Figure 18. High impedance-analog configuration



✧ Resumo da aula de hoje...

- Circuito de RESET
- Conceito de Watchdog timers
- Circuito de CLOCK do ARM – programador de clock
- Duplicadores de frequência, divisores de frequência, e PLL
- Timers – como são estruturados e configurados os timers do ARM Cortex
- PWM – pulse width modulation (uma saída analógica dos μC e SoC)
- Configuração dos pinos de entrada e saída do CI (GPIOs)
- Configuração dos pinos para SAÍDA, ENTRADA, função ALTERNATIVA e entrada ANALÓGICA.

✧ Próximo LAB ...

- ***CONJUNTO DE INSTRUÇÕES*** e a forma de operação do ARM.

Obrigado...

Até a próxima aula.