EA871

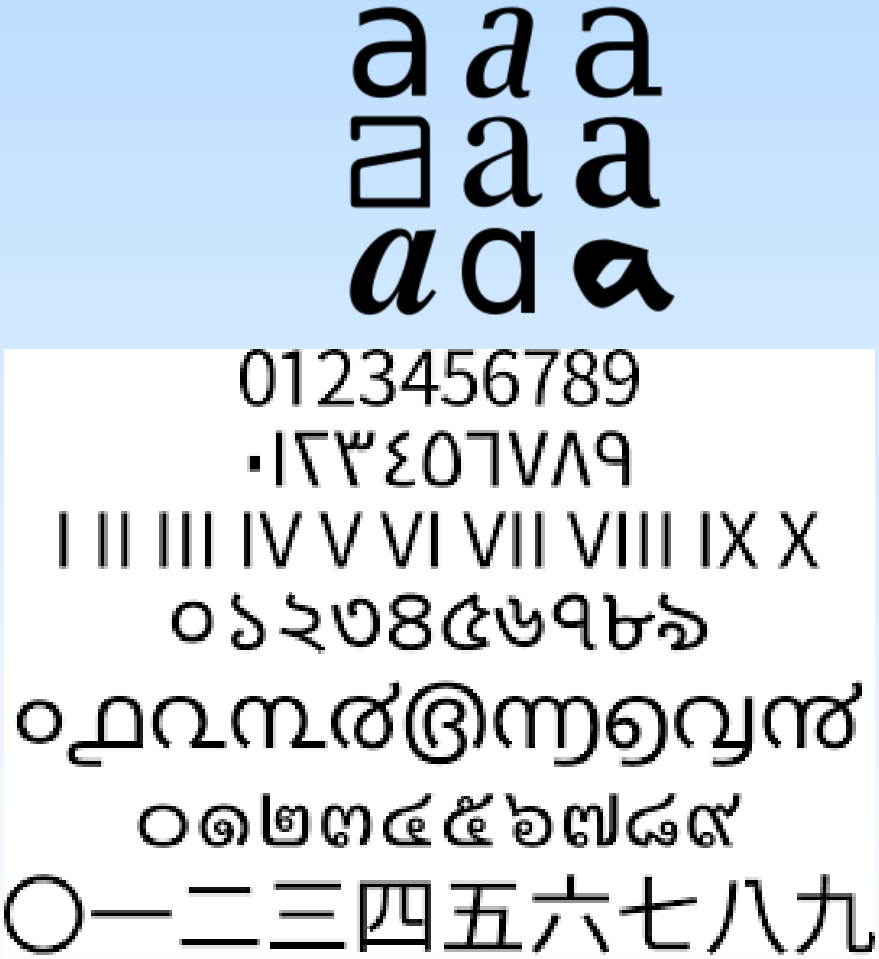
Temporizadores

HH:MM:SS em glifos

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

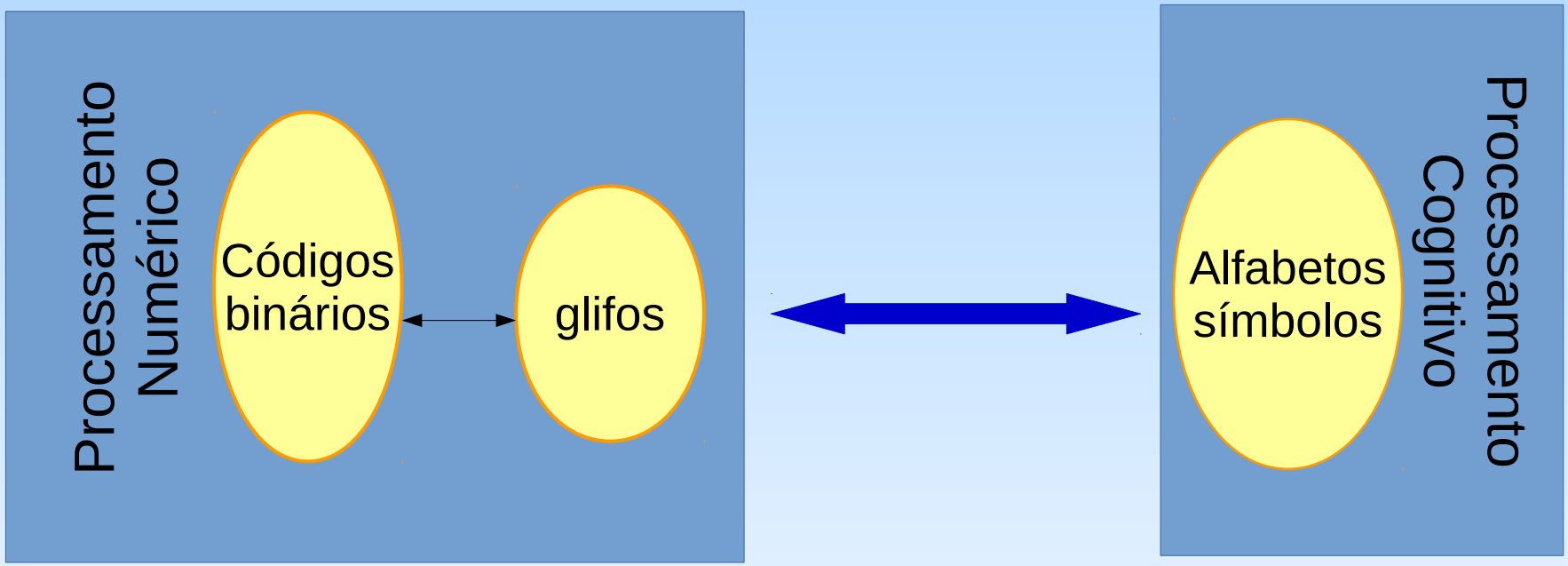
Glifos

- Símbolos gráficos que representam unidades de informação.



Fonte: Internet

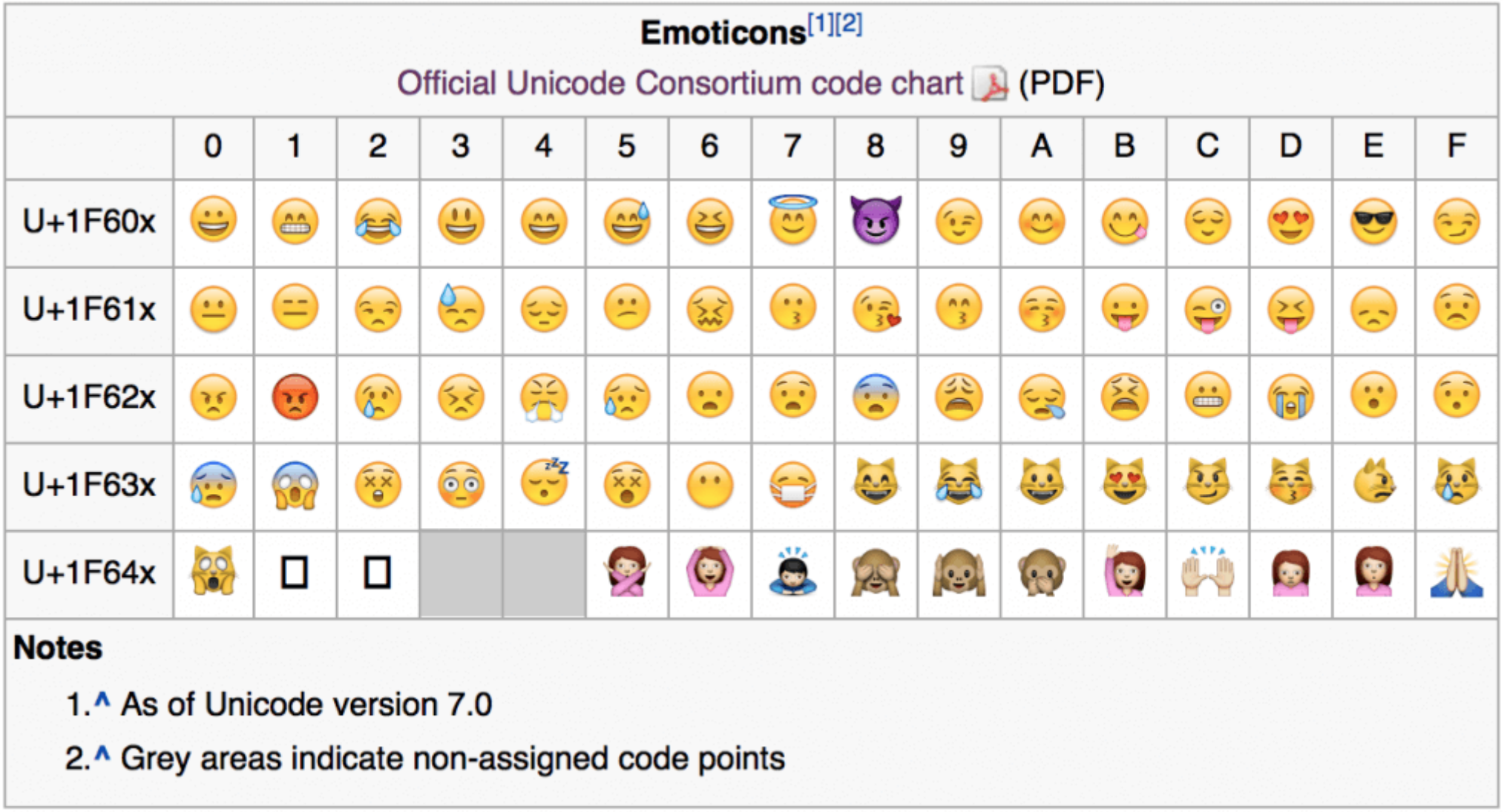
Interface Homem-Computador



ASCII: 00110011 00101010 00111000 **3*8**

UTF-8: 1100001110101001 **é**

Unicódigos



Emoticons[1][2]

Official Unicode Consortium code chart (PDF)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+1F60x	😀	😁	😂	😃	😄	😅	😆	😇	😈	😉	😊	😋	😌	😍	😎	😏
U+1F61x	😐	😑	😒	😓	😔	😕	😖	😗	😘	😙	😚	😛	😜	😝	😞	😟
U+1F62x	😠	😡	😢	😣	😤	😥	😦	😧	😨	😩	😪	😫	😬	😭	😮	😯
U+1F63x	😰	😱	😲	😳	😴	😵	😶	😷	😸	😹	😺	😻	😼	😽	😾	😿
U+1F64x	🙀	□	□			🙅	🙆	🙇	🙈	🙉	🙊	🙋	🙌	🙍	🙎	🙏

Notes

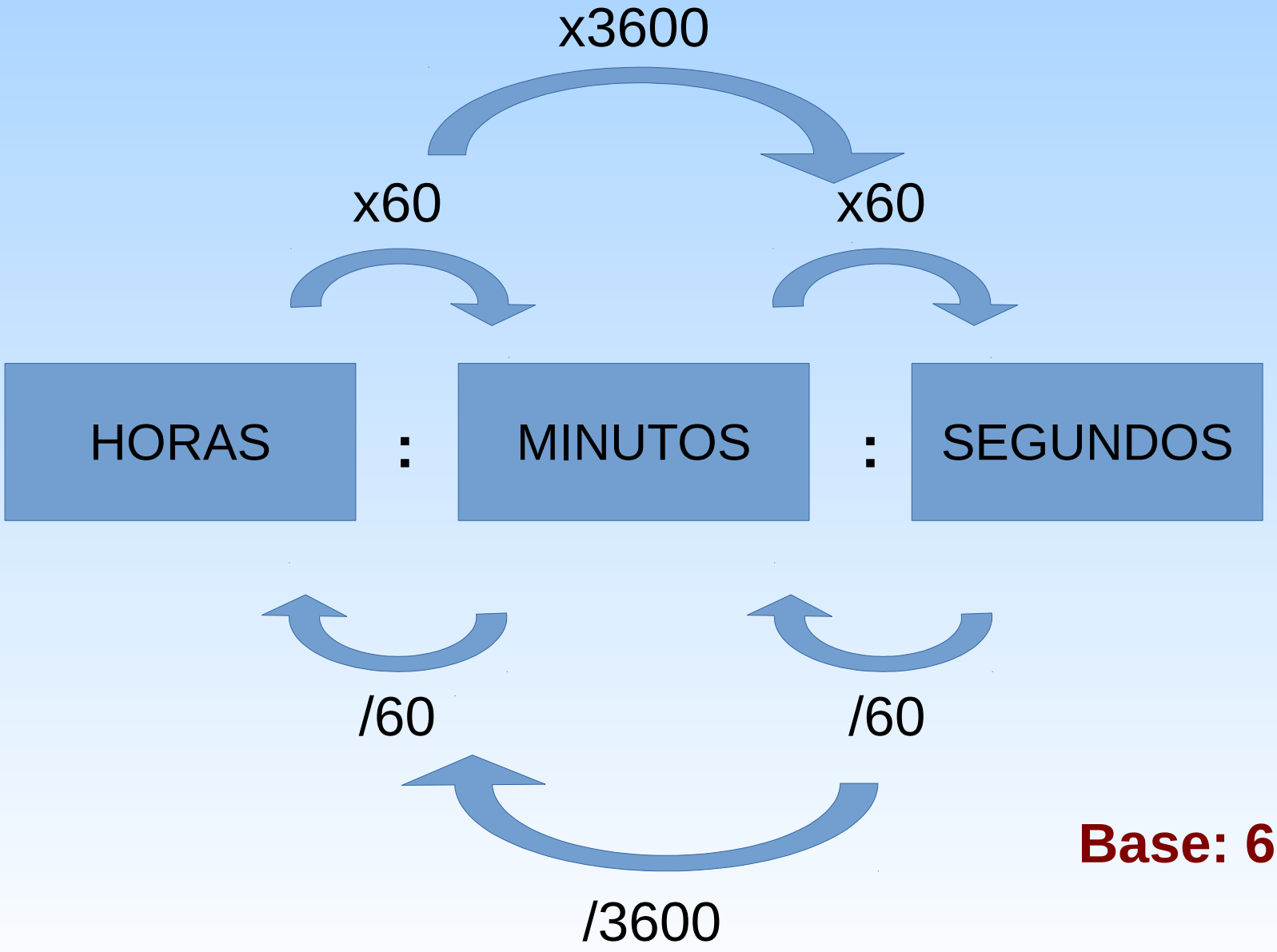
1. ^ As of Unicode version 7.0
2. ^ Grey areas indicate non-assigned code points

Fonte: https://guigarage.com/2015/01/support-emojis/

Algoritmos de Conversão

- **Correspondência biunívoca entre código-glifo**
  - Procedimentos baseados na busca pelos índices em tabelas (*LUT*, *Look Up Table*).
- **Unidade de informação constituída por uma sequência de glifos**:
  - Exemplos:
    - Números: sequências de dígitos
    - **Horário**: formato de 24 horas HH:MM:SS
  - Procedimentos baseados na identificação dos glifos e dos seus valores na unidade de informação.

Horário



Base: 60

Codificação em ASCII

- Números representados por dois algarismos do conjunto {‘0’,’1’,’2’,’3’,’4’,’5’,’6’}
  - ‘0’ – ‘6’: Códigos ASCII 0x30 – 0x36
- Separação de um número N em algarismo da casa dezena e da casa unidade
  - Casa dezena: N/10;
  - Casa unidade: N%10;
- Códigos dos algarismos diferem do algarismo ‘0’ no seu valor numérico:
  - Exemplo: valor = 6 → código ASCII = 0x36 = 6 + 0x30 = 6+’0’
- Preenchimento da casa dezena com ‘0’ quando N < 10

HH:MM:SS ↔ segundos

Hh:Mm:Ss → segundos

- a ← (H-’0’)*10+(h-’0’); b ← (M-’0’)*10+(m-’0’); c ← (S-’0’)*10+(s-’0’);
- S ← $a*60^2+b*60+c$

segundos → HH:MM:SS

- SS ← S%60;
- MM ← (S/60)%60;
- HH ← (S/60)/60;
- Conversão de v (SS/MM/HH) para string:

  Se v < 10 então str[0] ← ‘0’; str[1] ← v + ‘0’;

  senão str[0] ← v/10+’0’; str[1] ← v%10+’0’;

Projeto-Exemplo

- Converter o valor (inteiro) de uma variável “counter” para o formato de horas HH:MM:SS e exibí-lo na primeira linha do visor do LCD, alinhado à margem direita.

Técnica de Programação

- Uso do **operador ternário (?:)** para codificar tomadas de decisão *if-then-else*

  Se v < 10 então str[0] ← ‘0’; str[1]← v + ‘0’;

  senão str[0]← v/10+’0’; str[1] ← v%10+’0’;

  - Sintaxe do operador ternário

    expressão booleana **?** <códigos para **V**> **:** <códigos para **F**>;

  - Comando equivalente com uso de operador ternário:

    str[0] ← (v < 10) ? ’0’ : (v/10)+’0’;

    str[1] ← v%10 + ‘0’;

Técnicas de Programação

- Modularização dos códigos
  - Conversão de segundos para o formato HH:MM:SS

    char * ttoa (unsigned int seconds, char *string)
  - Conversão de horas no formato HH:MM:SS para segundos

    unsigned int atot (char *string, unsigned int *seconds)

Pseudo-Código

- GPIO_initLCD();
- InitLCD();
- Inicializar counter;
- Laço:
  - ttoa(counter, string);
  - escreveMensagem (0x08, string);
  - delay2us(250000);
  - counter++;

CodeWarrior IDE Development Suite

Informações Adicionais

- Time Formats

  https://docs.oracle.com/cd/E41183_01/DR/Time_Formats.html
- Convert seconds to HH:MM:SS

  https://www.tools4noobs.com/online_tools/seconds_to_hh_mm_ss/

EA871

Temporizadores

SysTick, PIT e LPTMR

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Revisão

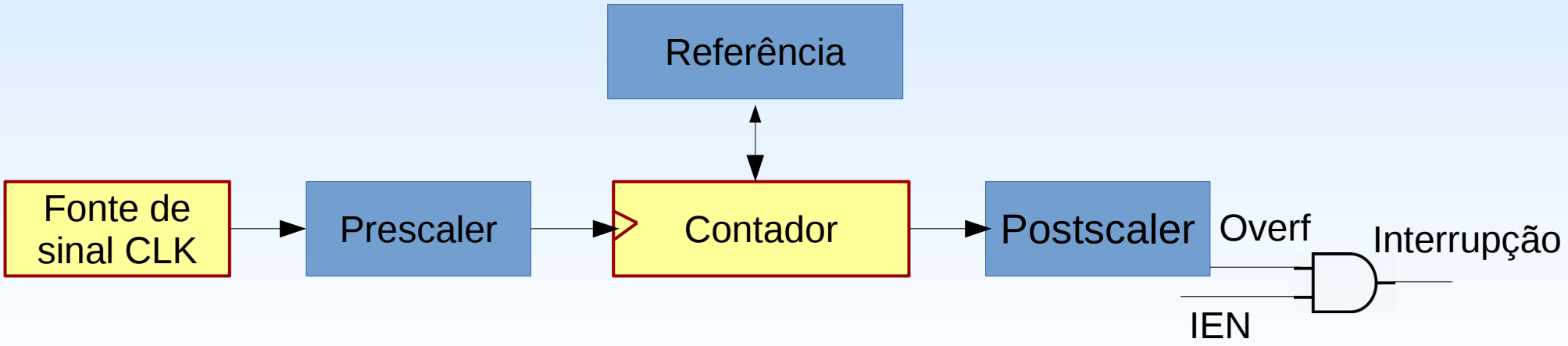
- Duas funções básicas de **temporizadores** em microcontroladores:
  - Medição de um intervalo (período) de tempo.
  - Contagem de eventos externos.
- Princípio de funcionamento:
  - como **um medidor de tempo**: **contagem** de *n* ciclos na base de tempo derivado do relógio do sistema de frequência f.

$$t = n \times \frac{1}{f}$$

  - como **um contador**: **contagem** de eventos externos de um sistema.

Revisão

- **Fonte de CLK:** base de tempo de contagem.
- ***Prescaler***: divisor de frequência do sinal da fonte.
- **Contador** crescente ou decrescente.
  - Crescente: 0 → referência.
  - Decrescente: referência → 0.
- ***Postscaler***: múltiplo de ciclos completos de contagem para geração de uma **interrupção por *overflow***.
- **Referência**: valor a ser carregado no contador (decrescente) ou a ser alcançado (crescente) nos ciclos de contagem.



Revisão

Relação de tempo *t* e a contagem *n*

- Frequência de CLK: *f*
- Divisor de frequência: Prescaler
- Contagem: *n*

$$t = \frac{Prescaler}{f} \times n$$

$$n = \frac{f}{Prescaler} \times t$$

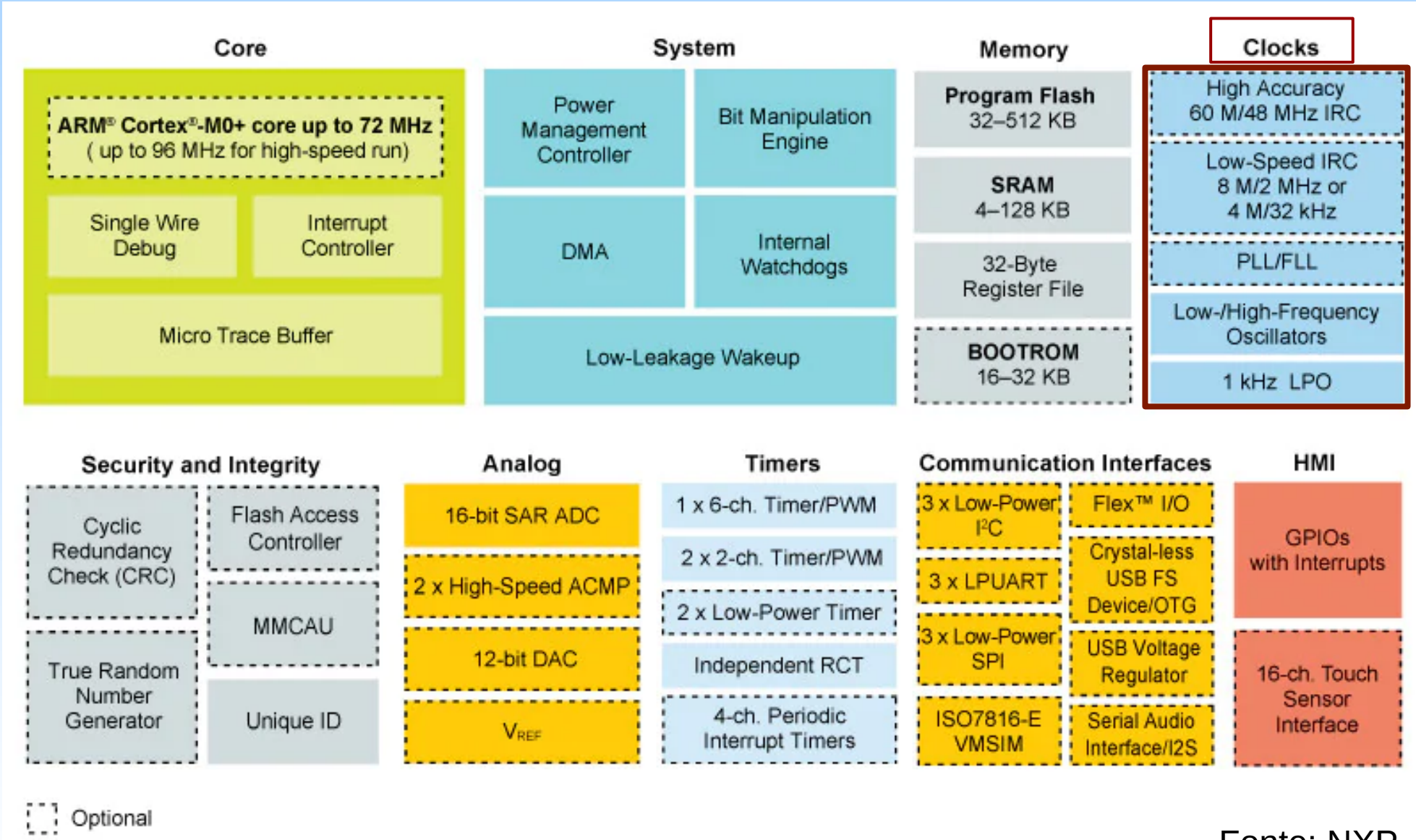
- *Overflow*: Referência * Postscaler

$$Período_{temporizador} = \frac{Prescaler}{f} \times Referência \times Postscaler$$

Revisão

- Eventos de *overflow* típicos:
  - **Contador decrescente**: quando a contagem atinge 0 (depois de *Postscaler* vezes),
    - valor de Referência é recarregado no Contador.
  - **Contador crescente**: quando a contagem atinge Referência (depois de *Postscaler* vezes),
    - valor no Contador pode ser resetado.

Kinetis KL25Z: Sinais de Relógio



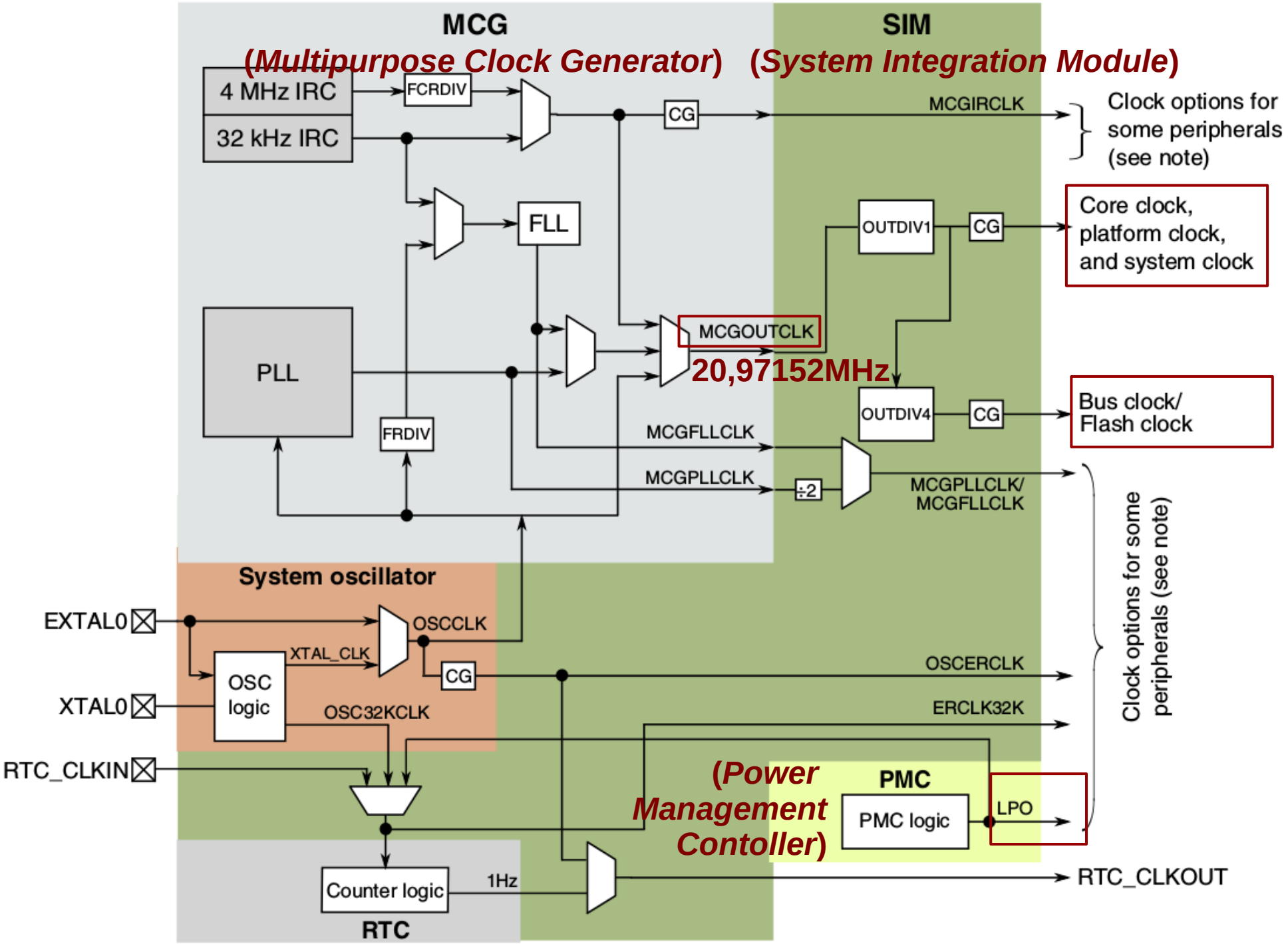
Fonte: NXP

Distribuição de Sinais de Relógio

Module	Bus interface clock	Internal clocks	I/O interface clocks
Core modules			
ARM Cortex-M0+ core	Platform clock	Core clock	—
NVIC	Platform clock	—	—
DAP	Platform clock	—	SWD_CLK
System modules			
DMA	System clock	—	—
DMA Mux	Bus clock	—	—
Port control	Bus clock	—	—
Crossbar Switch	Platform clock	—	—
Peripheral bridges	System clock	Bus clock	—
LLWU, PMC, SIM, RCM	Bus clock	LPO	—
Mode controller	Bus clock	—	—
MCM	Platform clock	—	—
Watchdog timer	Bus clock	LPO	—
Clocks			
MCG	Bus clock	MCGOUTCLK, MCGPLLCLK, MCGFLLCLK, MCGIRCLK, OSCERCLK	—
OSC	Bus clock	OSCERCLK	—
Memory and memory interfaces			
Flash Controller	Platform clock	Flash clock	—
Flash memory	Flash clock	—	—

Distribuição de Sinais de Relógio

Module	Bus interface clock	Internal clocks	I/O interface clocks
ADC	Bus clock	OSCERCLK	—
CMP	Bus clock	—	—
DAC	Bus clock	—	—
Timers			
TPM	Bus clock	TPM clock	TPM_CLKIN0, TPM_CLKIN1
PIT	Bus clock	—	—
LPTMR	Bus clock	LPO, OSCERCLK, MCGIRCLK, ERCLK32K	—
RTC	Bus clock	ERCLK32K	RTC_CLKOUT
Communication interfaces			
USB FS OTG	System clock	USB FS clock	—
SPI0	Bus clock	—	SPI0_SCK
SPI1	System clock	—	SPI1_SCK
I^2C0	Bus clock	—	I2C0_SCL
I^2C1	Bus clock	—	I2C1_SCL
UART0	Bus clock	UART0 clock	—
UART1 , UART2	Bus clock	—	—
Human-machine interfaces			
GPIO	Platform clock	—	—
TSI	Bus clock	—	—



CG — Clock gate

Note: See subsequent sections for details on where these clocks are used.

Divisores de Frequência

SIM_CLKDIV1

Address: 4004_7000h base + 1044h offset = 4004_8044h

Bit	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
R	OUTDIV1				0									OUTDIV4			0															
W																																
Reset	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- OUTDIV1 field: The reset value depends on the FTF_FOPT[LPBOOT]. it is loaded with 0000 (divide by one), 0001 (divide by two), 0011 (divide by four), or 0111 (divide by eight).

Core, Platform, System Clock =

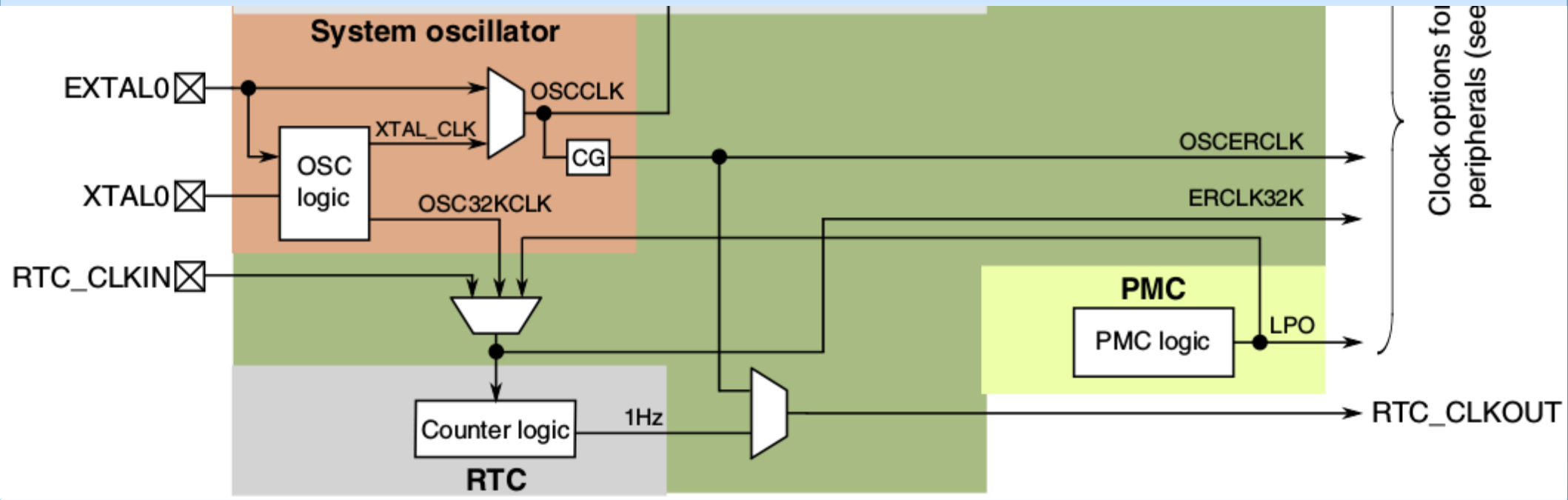
MCGOUTCLK/OUTDIV1 = 20,97152MHz

Bus, Flash Clock =

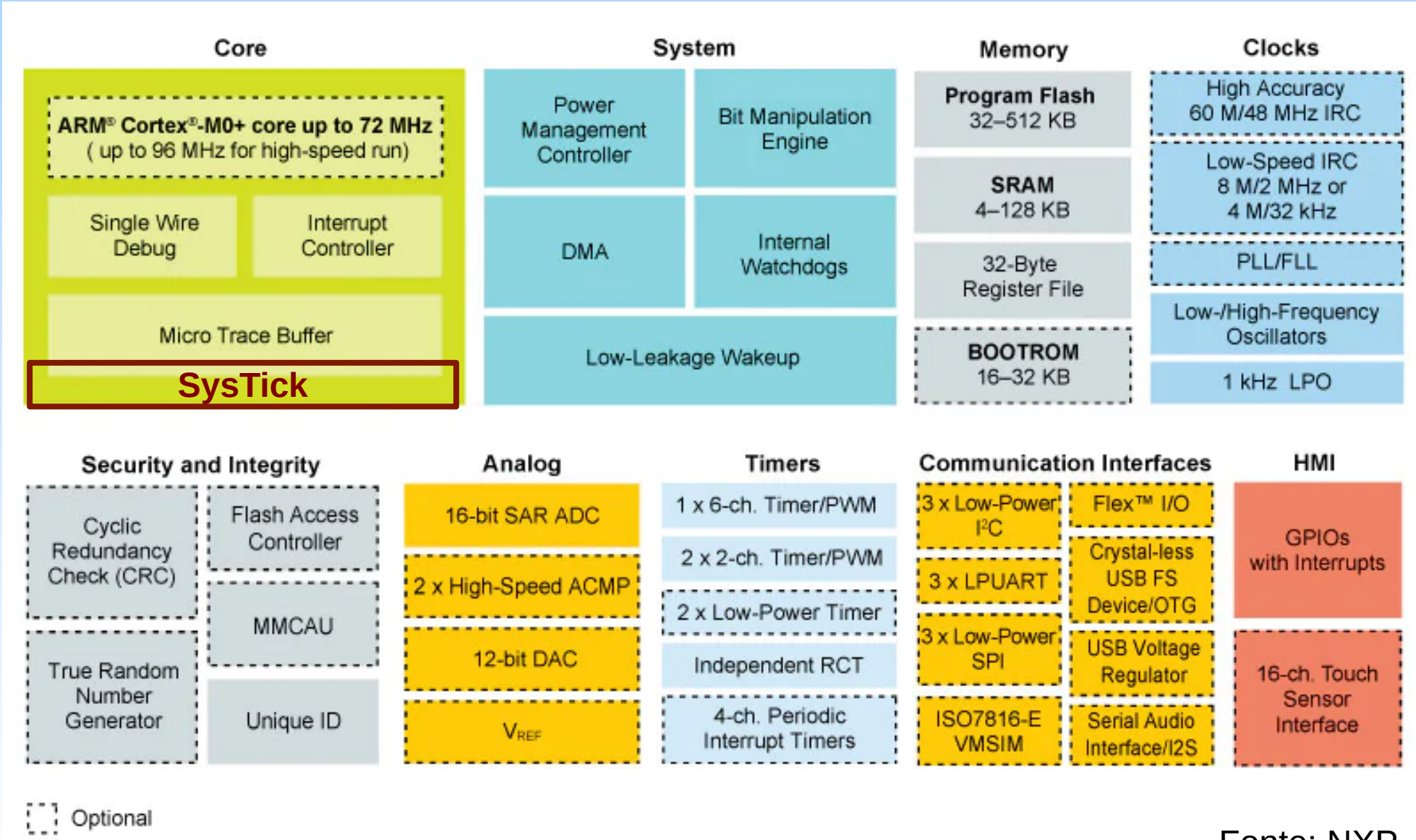
System Clock/OUTDIV4 = 10,48576MHz

LPO

- **Oscilador de baixa potência** *(Low Power Oscillator)*
  - 1kHz
  - gerado pelo *Power Management Controller* (PMC)
  - habilitado para quase todos os modos de baixa potência (WAIT, STOP, VLPR, VLPW, etc.)



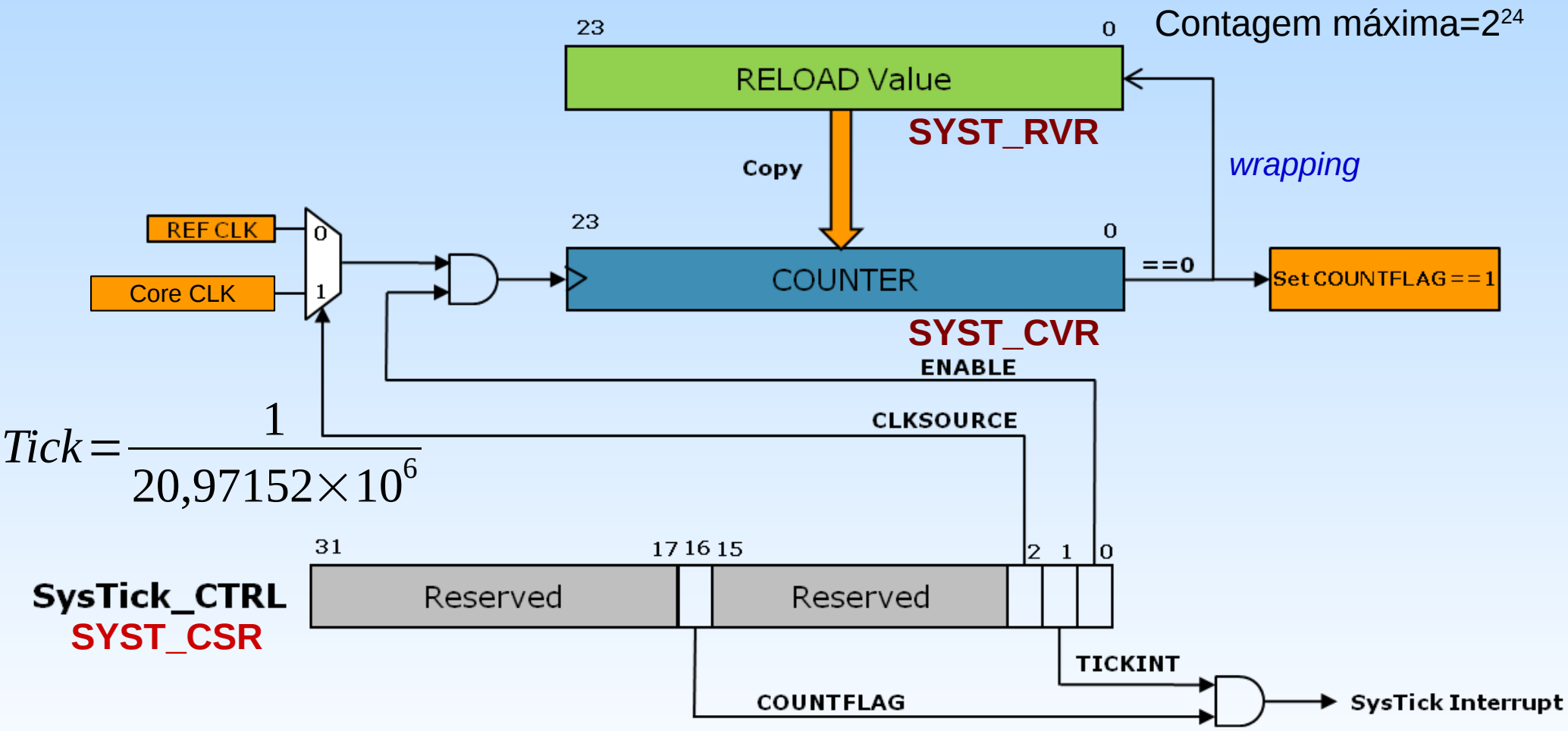
Microcontrolador Kinetis KL25Z



Fonte: NXP

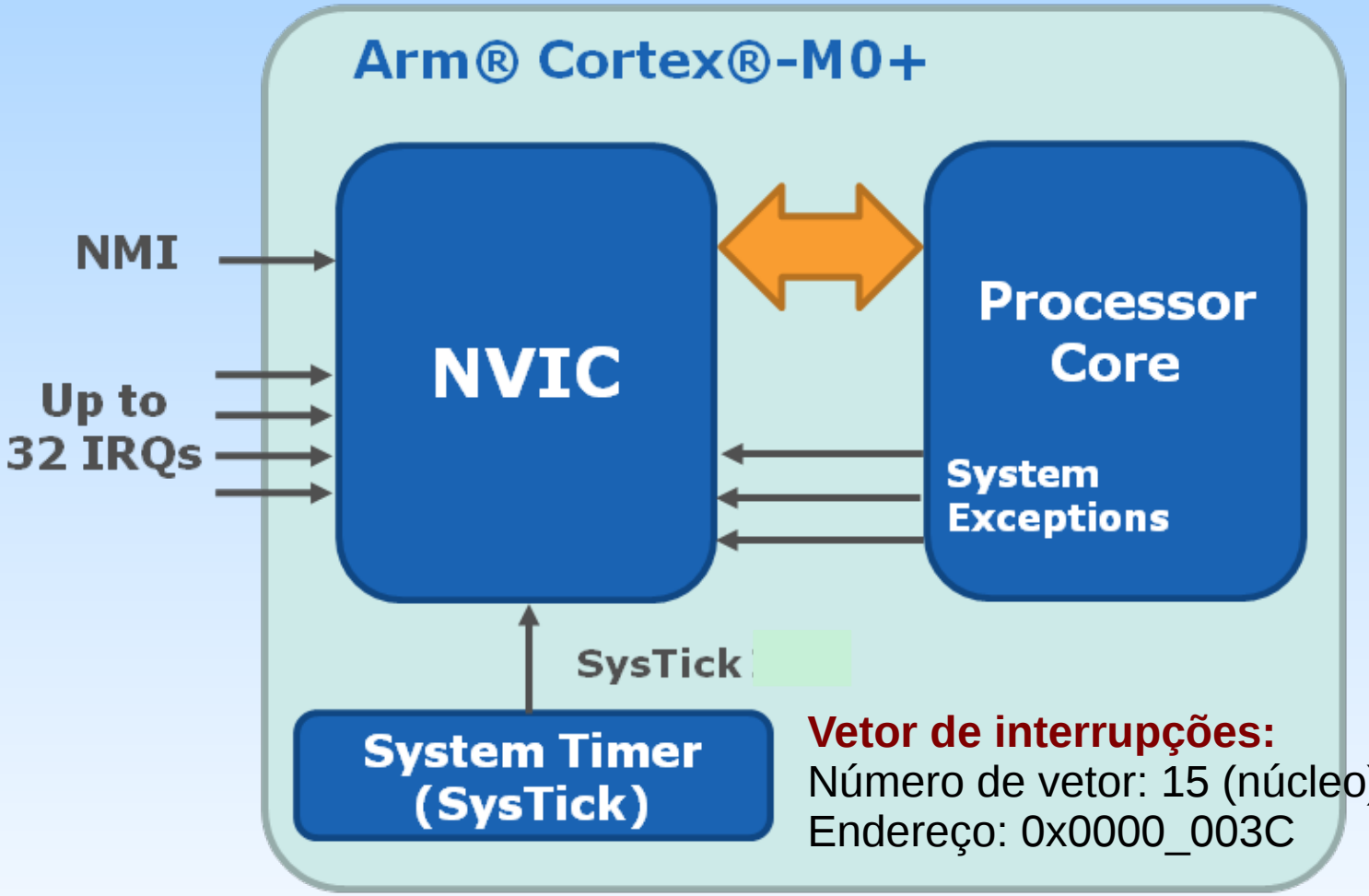
SysTick

- Um temporizador de núcleo com um contador de 24 *bits*, de **contagem decrescente**.



Adaptado: https://microchipdeveloper.com/32arm:samd21-systick

SysTick: Exceção



Vetor de interrupções:
Número de vetor: 15 (núcleo)
Endereço: 0x0000_003C

Adaptado: https://microchipdeveloper.com/32arm:samd21-systick

Registradores

- SYST_CSR: registrador de controle e de estado
  - CLK do núcleo
- SYST_RVR: registrador de recarga
- SYST_CVR: contador

SysTick: SYST_CSR

Bits	TYPE	Name	Function
[31:17]	-	-	Reserved.
[16]	RO	COUNTFLAG	Indicates whether the counter has counted to 0 since the last read of this register: **0** timer has not counted to 0. **1** timer has counted to 0. COUNTFLAG is set to 1 by a count transition from 1 to 0. COUNTFLAG is cleared to 0 by a read of this register, and by any write to the Current Value register.
[15:3]	-	-	Reserved.
[2]	RW	CLKSOURCE	Indicates the SysTick clock source: **0** SysTick uses the optional external reference clock. **1** SysTick uses the processor clock. If no external clock is provided, this bit reads as one and ignores writes.
[1]	RW	TICKINT	Indicates whether counting to 0 causes the status of the SysTick exception to change to pending: **0** count to 0 does not affect the SysTick exception status. **1** count to 0 changes the SysTick exception status to pending. Changing the value of the counter to 0 by writing zero to the SysTick Current Value register to 0 never changes the status of the SysTick exception.
[0]	RW	ENABLE	Indicates the enabled status of the SysTick counter: **0** counter is disabled. **1** counter is operating.

SysTick: Programação

- Inicialização

  SYST_RVR = SysTick_RVR_RELOAD(recarga);

  SYST_CVR = SysTick_CVR_CURRENT(0); //reseta COUNTFLAG e recarga

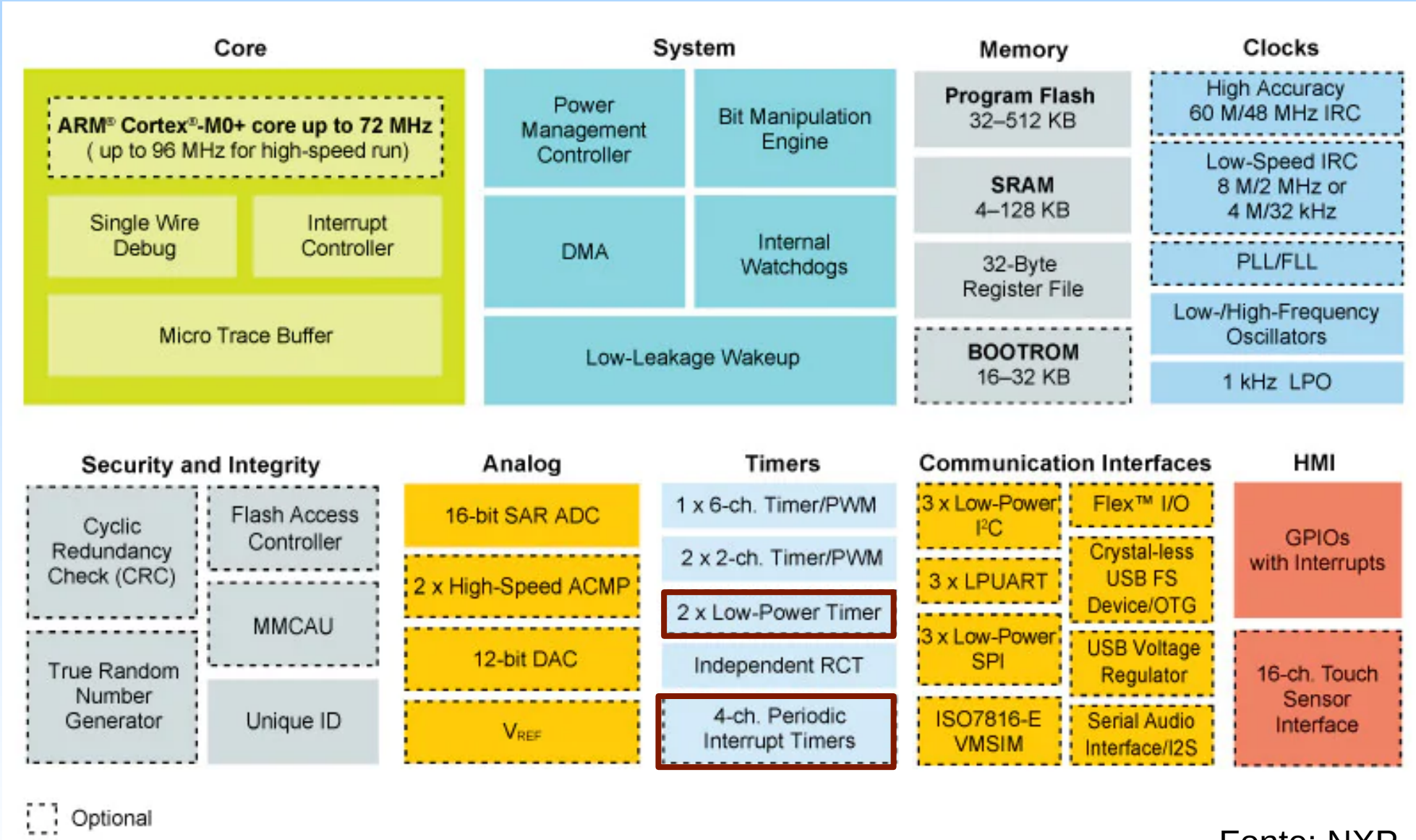
  SYST_CSR |= (SysTick_CSR_CLKSOURCE_MASK | // CLK
  SysTick_CSR_ENABLE_MASK); // habilita contador

- Processamento de Exceção (Núcleo)

  SYST_CSR |= SysTick_CSR_TICKINT_MASK;

  - Definir a rotina de serviço **SysTick_Handler()**

Microcontrolador Kinetis KL25Z



Fonte: NXP

Sinais de Relógio dos Temporizadores

Module	Bus interface clock	Internal clocks	I/O interface clocks
ADC	Bus clock	OSCERCLK	—
CMP	Bus clock	—	—
DAC	Bus clock	—	—
Timers			
TPM	Bus clock	TPM clock	TPM_CLKIN0, TPM_CLKIN1
PIT	Bus clock	—	—
LPTMR	Bus clock	LPO, OSCERCLK, MCGIRCLK, ERCLK32K	—
RTC	Bus clock	ERCLK32K	RTC_CLKOUT
Communication interfaces			
USB FS OTG	System clock	USB FS clock	—
SPI0	Bus clock	—	SPI0_SCK
SPI1	System clock	—	SPI1_SCK
I^2C0	Bus clock	—	I2C0_SCL
I^2C1	Bus clock	—	I2C1_SCL
UART0	Bus clock	UART0 clock	—
UART1 , UART2	Bus clock	—	—
Human-machine interfaces			
GPIO	Platform clock	—	—
TSI	Bus clock	—	—

PIT *(Periodic Interrupt Timers)*

- 2 temporizadores com contadores de 32 *bits*, de **contagem decrescente**, encadeáveis em um contador de 64 *bits* e capazes de gerarem pulsos (triggers) periódicos.

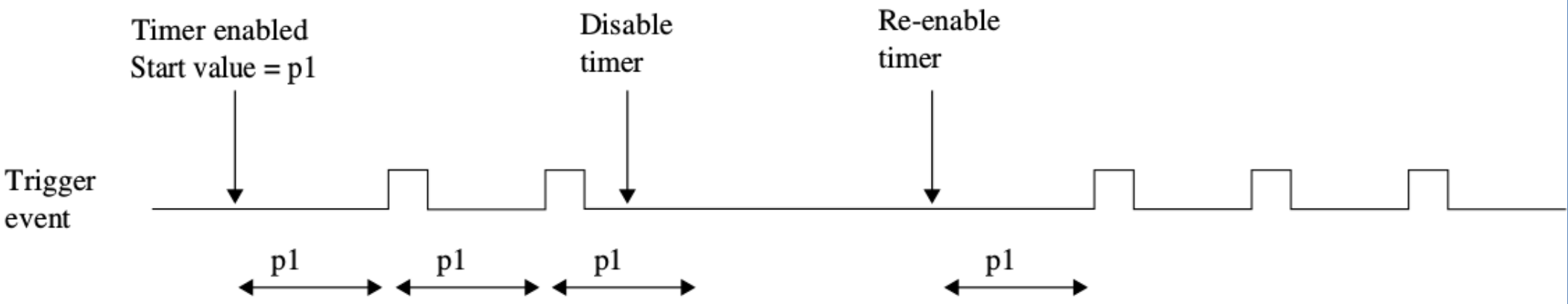
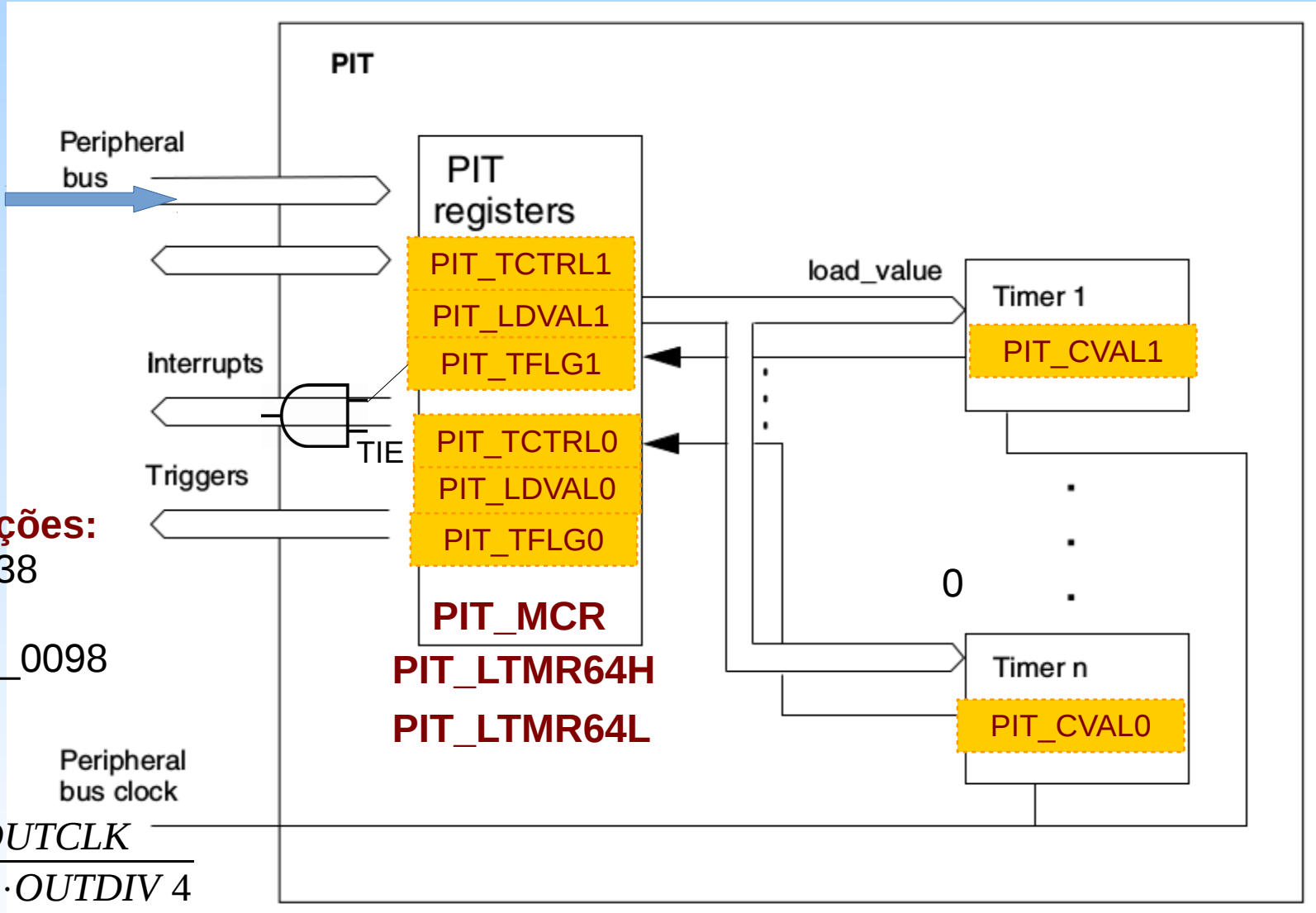


Figure 32-17. Stopping and starting a timer

PIT: Fluxo de Dados

SIM_SCGC6[23]



Vetor de interrupções:
Número de vetor: 38
IRQ: 22
Endereço: 0x0000_0098

PIT: PIT_MCR

Address: 4003_7000h base + 0h offset = 4003_7000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													Reserved	MDIS	FRZ
W															Module Disable	Freeze
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

PIT_MCR field descriptions

PIT_TCTRLn

32.3.6 Timer Control Register (PIT_TCTRL*n*)

These register contain the control bits for each timer.

Address: 4003_7000h base + 108h offset + (16d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TCTRL*n* field descriptions

PIT: PIT_TFLGn

Address: 4003_7000h base + 10Ch offset + (16d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TFLG*n* field descriptions

Timer Interrupt Flag

Uma nova interrupção é gerada se a anterior não ter sido removida.

Registradores

- SIM_CLKDIV1: registrador de divisor de frequência
- SIM_SCGC6: registrador para habilitar CLK

- PIT_MCR: registrador de controle de módulo
- PIT_LTMR64H, PIT_LTMR64L : contador de 64 *bits*
- PIT_LDVALn: registrador de recarga
- PIT_CVALn: contador
- PIT_TCTRLn: registrador de controle
- PIT_TFLGn: registrador de estado

- NVIC_ISER
- NVIC_ICER
- NVIC_ISPR
- NVIC_ICPR
- NVIC_IPR5

PIT: Programação

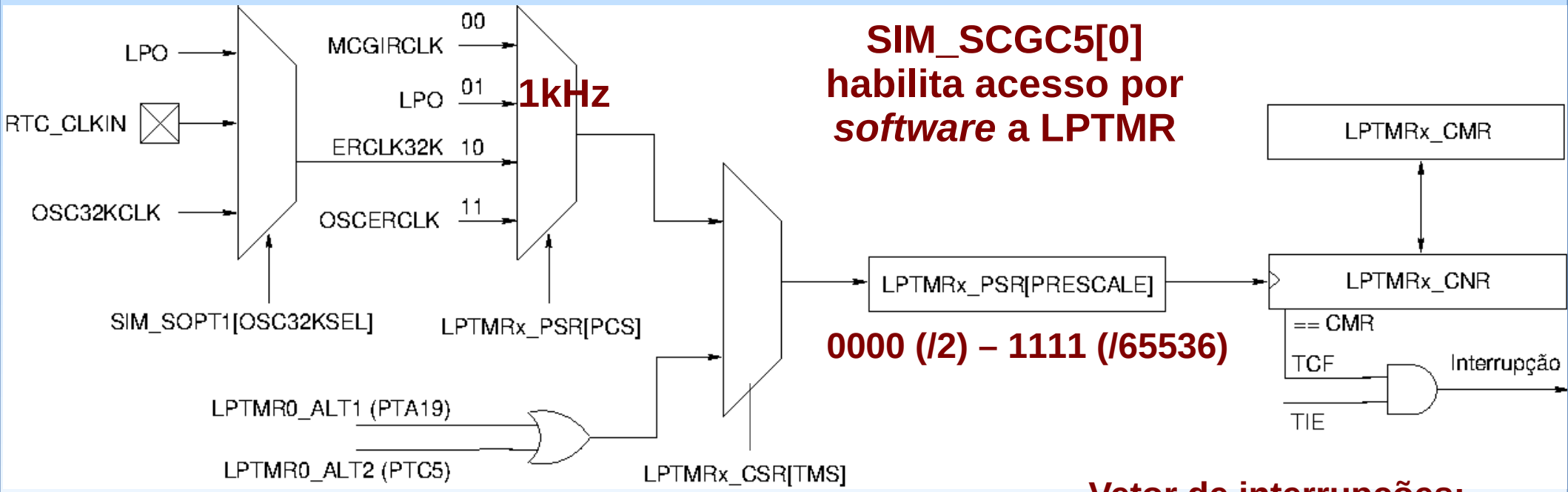
- Inicialização

```
SIM_SCGC6 |= SIM_SCGC6_PIT_MASK;  //habilita clock
SIM_CLKDIV1 &= ~SIM_CLKDIV1_OUTDIV4(0b111);  //divisor 1

PIT_TCTRL0 &= ~PIT_TCTRL_CHN_MASK;  // não encadeados
PIT_LDVAL0 = PIT_LDVAL_TSV(periodo); //recarga

PIT_TFLG0 |= PIT_TFLG_TIF_MASK; // w1c
PIT_TCTRL0 |= (PIT_TCTRL_TEN_MASK); //habilita timer
PIT_MCR &= ~PIT_MCR_MDIS_MASK;     //habilita modulo
```

- Processamento de Interrupções
  - PIT_TCTRL0 |= PIT_TCTRL_TIE_MASK;
  - NVIC – IRQ22 (Número de exceção 38)
  - Definir a rotina de serviço **PIT_IRQHandler**

LPTMR (*Low Power Timer*)

- Um contador **crescente** de 16 *bits*, de baixa potência, configurável para contagem de instantes de tempo ou para **contagem** de pulsos. Fonte de sinal de CLK pode ser externa.



SIM_SCGC5[0] habilita acesso por *software* a LPTMR

Vetor de interrupções:
Número de vetor: 44
IRQ: 28
Endereço: 0x0000_00B0

LPTMR: LPTMRx_CSR

Address: 4004_0000h base + 0h offset = 4004_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								TCF	TIE	TPS		TPP	TFC	TMS	TEN
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LPTMRx_CSR field descriptions

Timer Free Running Counter

Timer Mode Select

Timer Enable

Vetor de interrupções:
Número de vetor: 44
IRQ: 28
Endereço: 0x0000_00B0

Registradores

- SIM_SOPT1: registrador de seleção da fonte de CLK de 32kHz
- SIM_SCGC5: registrador para habilitar acesso por *software*

- PORTA_PCR19, PORTC_PCR5: registradores de controle dos pinos de eventos externos

- LPTMR0_CSR: registrador de controle e de estado
  LPTMR0_PSR: registrador de Prescale
- LPTMR0_CMR: registrador de comparação
- LPTMR0_CNR: contador

- NVIC_ISER
- NVIC_ICER
- NVIC_ISPR
- NVIC_ICPR
- NVIC_IPR7

LPTMRx: Programação

- Inicailização

```
SIM_SCGC5 |= SIM_SCGC5_LPTMR_MASK;
LPTMR0_CSR &= ~(LPTMR_CSR_TFC_MASK  // CNR = CMR
                | LPTMR_CSR_TMS_MASK);  // contagem por tempo
LPTMR0_PSR |= LPTMR_PSR_PCS(0b01);    // LPO
LPTMR0_PSR |= LPTMR_PSR_PRESCALE(0b0000); // LPO/2
LPTMR0_CMR = LPTMR_CMR_COMPARE(valor);
```

  - LPTMR0_CSR != LPTMR_CSR_TCF_MASK; //w1c
  - LPTMR0_CSR |= LPTMR_CSR_TEN_MASK; // habilita timer

- Processamento de Interrupção
  - LPTMR0_CSR |= LPTMR_CSR_TIE_MASK;
  - NVIC – IRQ28 (Número de exceção 44)
  - Definir a rotina de serviço **LPTimer_IRQHandler**

Projeto-Exemplo

- Frequência de piscadas do *led* RGB: 2Hz (0.25s)
  - *led* R: controlada pelo SysTick (vetor de exceção: 15)

$$Referência\left(RVR\right)=0{,}25\times 20{,}97152\times 10^{6}=5242880$$

  - *led* G: controlada pelo PIT (OUTDIV4 = 1, IRQ=22)

$$Referência\left(LDVAL\,0\right)=0{,}25\times 20{,}97152\times 10^{6}=5242880$$

  - *led* B: controlada pelo LPTMR0 (LPO = 1KHz)

$$Referência\left(CMR\right)=0{,}25\times\frac{1000}{2}=125$$

Rotinas de Serviço

```
void SysTick_Handler() {
    GPIO_PTOR = 1<<21;
}
void PIT_IRQHandler() {
    GPIO_PTOR = 1<<22;
    PIT_TFLG0 |= PIT_TFLG_TIF_MASK;   //baixa a bandeira
}
void LPTimer_IRQHandler() {
    GPIO_PTOR = 1<<23;
    LPTMR0_CSR |= LPTMR_CSR_TCF_MASK;  //baixa a bandeira
}
```

Técnica de Programação

- Modualização de códigos

  Inicialização do módulo SysTick

  SysTick_init();

  Inicialização do Timer 0 do módulo PIT

  PIT_Timer0(unsigned int periodo);

  Habilitação da interrupção

  PIT_enableNVICInterrupt(char priority);

  Inicialização do LPTMR

  LPTimer_init(unsigned short valor);

  Habilitação da Interrupção

  LPTimer_enableNVICInterrupr(char priority);

Pseudocódigo do fluxo principal

GPIO_initLedRGB();

SysTick_init (5242880); //5242880/20971520=0.25s
PIT_initTimer0(5242880); //5242880/20971520=0.25s
PIT_enableInterrupt(3);
LPTimer_init(125); //2*125/1000=0.25s
LPTimer_enableInterrupt(3);
Laço de espera:
    (pelos eventos de interrupção)

CodeWarrior IDE Development Suite

Informações Adicionais

- Timer/Counter Module – A Controller Independet Guide

  https://embedjournal.com/timer-modules-guide/

- Using Low Power modes on Kinetis family

  https://os.mbed.com/media/uploads/GregC/an4470-using_low-power_modes_with_kinetis_mcus.pdf

- ARMv6-M Architecture Reference Manual

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/ARMv6-M.pdf

  - SysTick: Seção B3.3 (página 275)

Informações Adicionais

- PIT – Periodic Interrupt Timer para FRDM-KL25Z

  https://www.embarcados.com.br/pit-periodic-interrupt-timer-para-frdm-kl25z/

- KL25 Sub-Family Reference Manual

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf

  - NVIC: Capítulo 3 (página 51)
  - Clock Distribution: Capítulo 5 (página 115)
  - PIT: Capítulo 32 (página 573)
  - LPTMR: Capítulo 3 (página 89), Capítulo 33 (página 587)

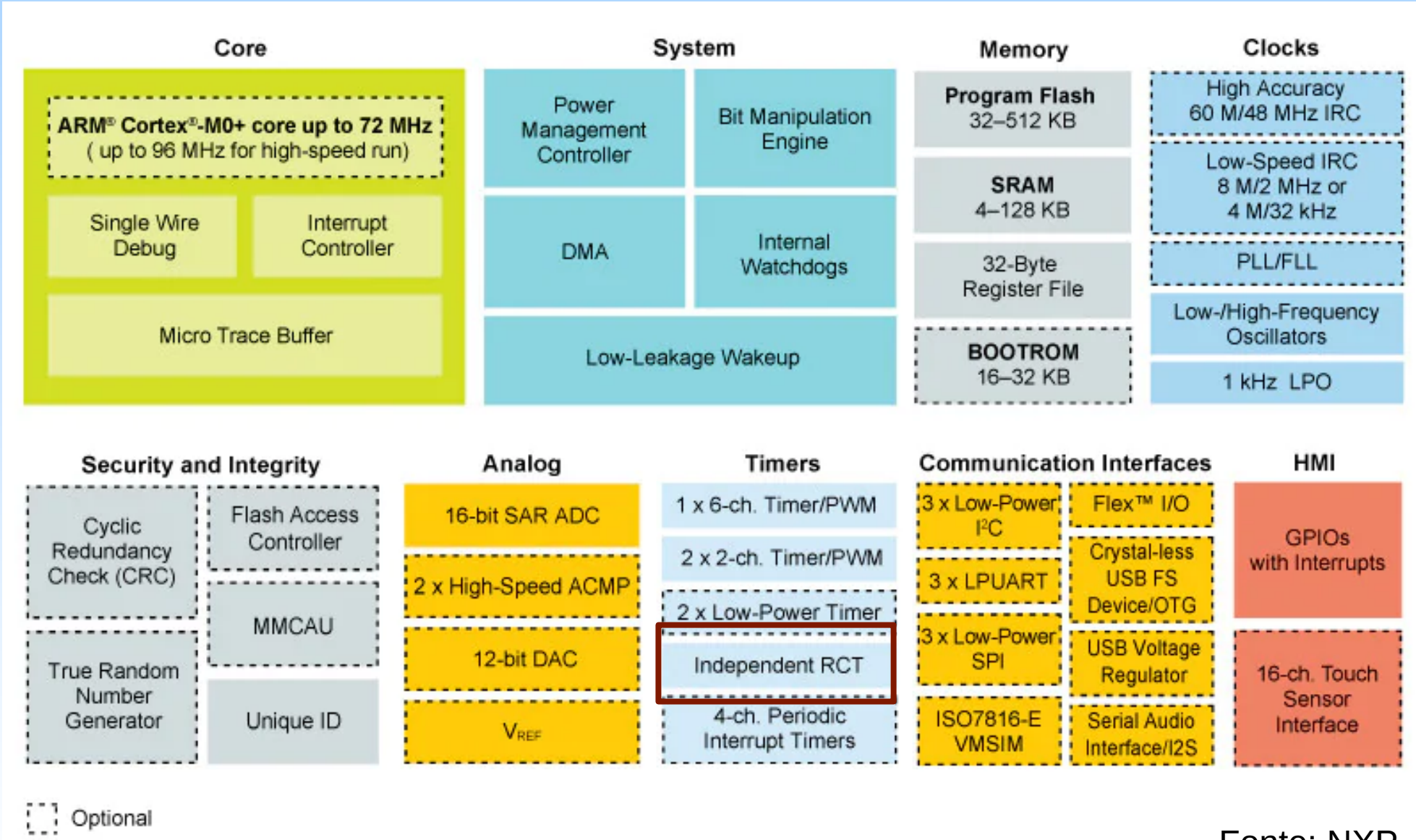
EA871

Temporizadores

RTC

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Microcontrolador Kinetis KL25Z

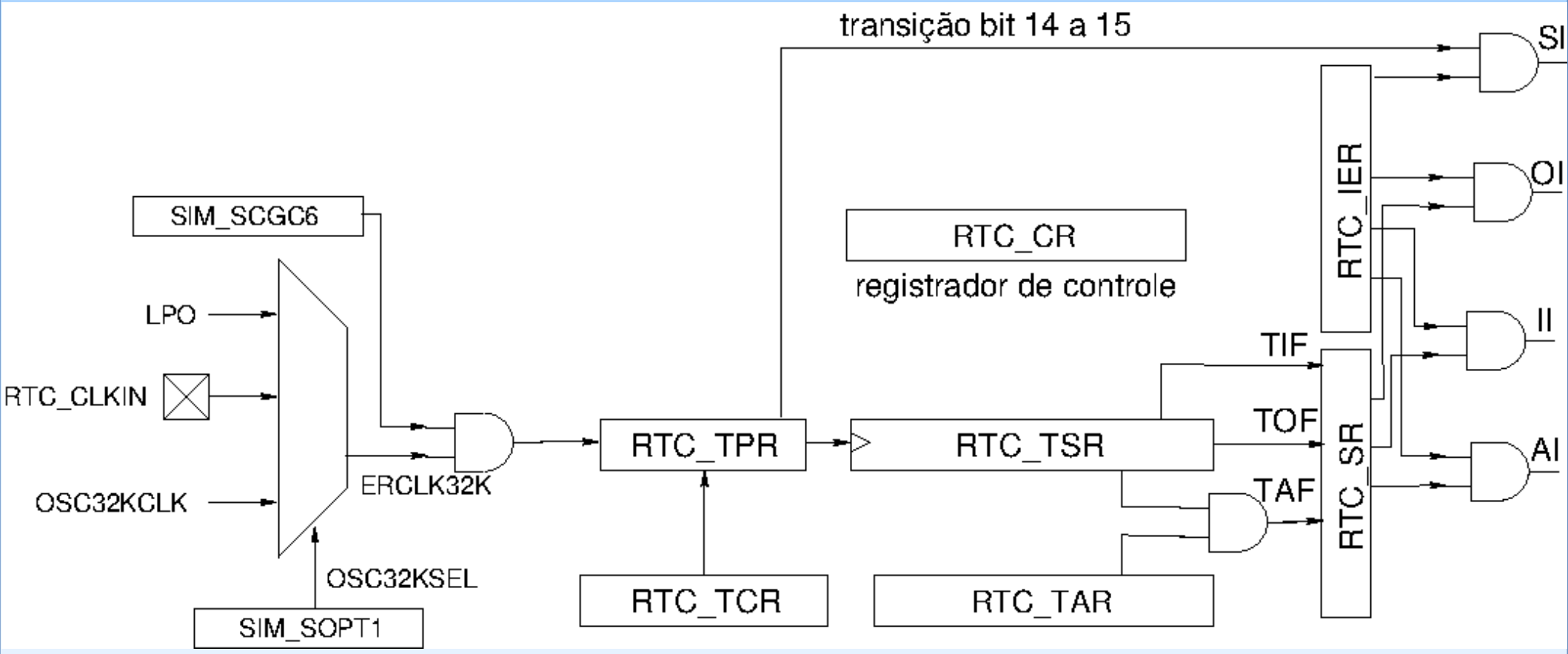


Fonte: NXP

RTC (*Real Time Clock*)

- É um temporizador com 1 contador de 32 *bits*, de **contagem crescente**, dedicado para manter o controle do tempo presente.
  - Projetado para operar com frequência 32,768kHz.
  - Três possíveis fontes de clock.
  - Gera uma onda quadrada de 1Hz.

RTC: Fluxo de Dados



SI:
Número de vetor: 37
IRQ: 21
Endereço: 0x0000_0094

OI, II, AI:
Número de vetor: 36
IRQ: 20
Endereço: 0x0000_0090

Modo de Operação

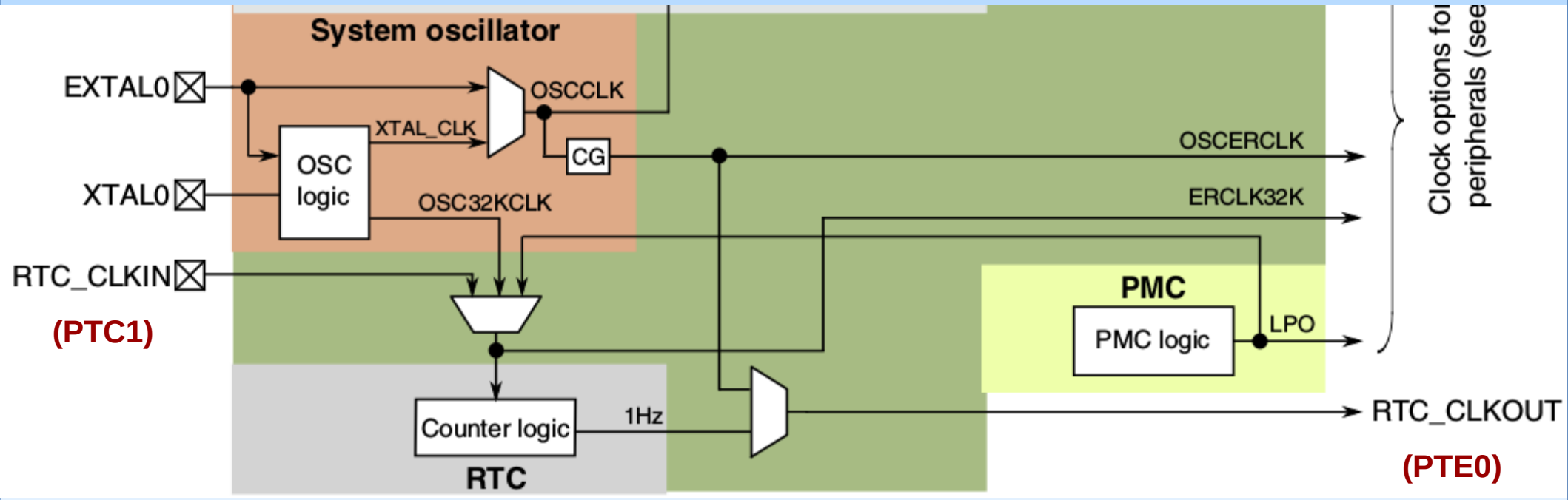
- RTC_TPR (16 *bits*): contador opera a frequência da fonte de CLK
- RTC_TSR (32 *bits*): contador opera a frequência da transição do bit 14 para 15.
  - a cada 2^{15} *ticks* da fonte de CLK, o *bit* 15 do RTC_TPR muda para 1, o contador RTC_TSR é incrementado de 1.

$$Período_{TPR} = \frac{1}{f_{CLK}} \times 2^{15}$$

$$f_{CLK} = 32.768\,Hz \rightarrow Incremento_{TSR} = 1\,s$$

- RTC_TCR: compensação de erros
  - 0.12 ppm (0.01s/dia,0.06min/ano)
  - 3906 ppm (337,48s/dia, 2052,99min/ano)

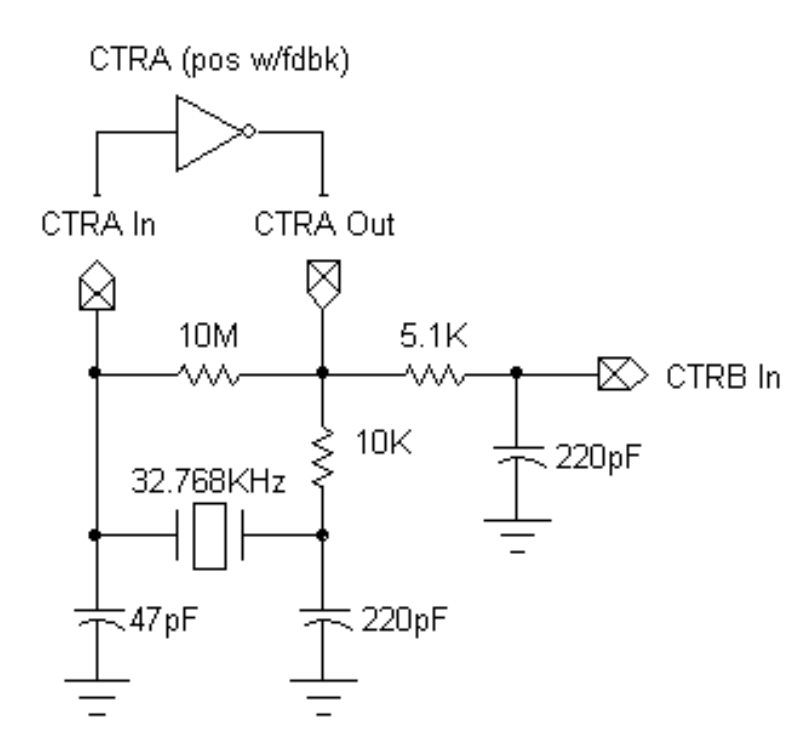
Fontes de CLK



$$f_{CLK} = 32{,}768\,kHz$$

Fonte de CLK: OSC32KCLK

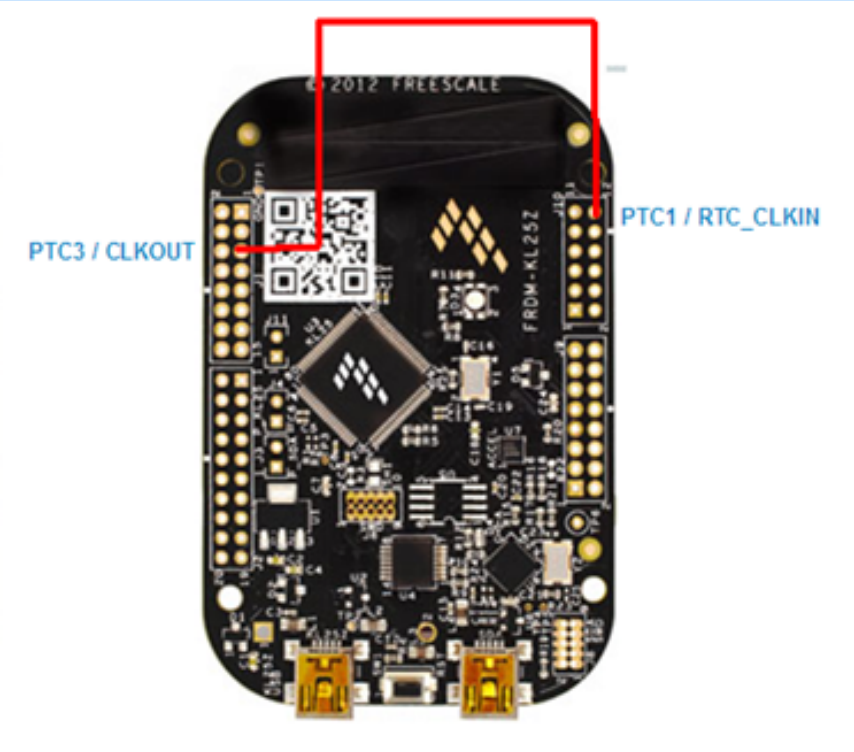
- Circuito de oscilador a cristal



Fonte: https://forums.parallax.com/discussion/download/80585/32KHz_osc.gif

Fonte de CLK: RTC_CLKIN

- Sinal de frequência 32,768kHz externo.
  - É possível usar o sinal gerado pelo microcontrolador e acessível pelo pino PTC3.



Fonte de CLK: LPO

- *Low Power Oscillator*, a frequência de 1kHz
  - Intervalo de incremento do RTC_TSR

$$Período_{TPR} = \frac{1}{f_{CLK}} \times 2^{15}$$

$$f_{CLK} = 1000\,Hz \rightarrow Incremento_{TSR} = 32{,}768\,s$$

  - Tempo presente

$$t = TSR \times \frac{32768}{1000} + \frac{TPR}{1000}$$

Curso é focado na programação
PTC1 e PTC3 alocados para LCD → Usaremos a fonte LPO.

Registradores

- SIM_SOPT1: selecionar a fonte de CLK
- SIM_SCGC6: registrador para habilitar CLK

- RTC_TSR: contador
- RTC_TPR: prescaler
- RTC_TCR: registrador de compensação de erros nos *ticks* dos tempos
- RTC_CR: registrador de controle
- RTC_SR: registrador de estado
- RTC_IER: registrador de habilitação de interrupções
- RTC_LR: registrador de trava dos estados do RTC

- NVIC_ISER
- NVIC_ICER
- NVIC_ISPR
- NVIC_ICPR
- NVIC_IPR5

RTC: RTC_SR

34.2.6 RTC Status Register (RTC_SR)

Address: 4003_D000h base + 14h offset = 4003_D014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											TCE	0	TAF	TOF	TIF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

RTC_SR field descriptions

RTC: RTC_IER

Address: 4003_D000h base + 1Ch offset = 4003_D01Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								WPON	Reserved		TSIE	Reserved	TAIE	TOIE	TIIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

RTC_IER field descriptions

RTC com fonte LPO: Programação

- Inicailização

```
SIM_SCGC6 |= SIM_SCGC6_RTC_MASK;
SIM_SOPT1 |= SIM_SOPT1_OSC32KSEL (0b11);
RTC_CR |= RTC_CR_SWR_MASK;        //resetar os registradores
RTC_CR &= ~RTC_CR_SWR_MASK;
RTC_TCR=RTC_TCR_CIR(0x01) |
                RTC_TCR_TCR(0xFF);
RTC_TSR = 0;
RTC_IER &= ~(RTC_IER_TAIE_MASK |
                    RTC_IER_TOIE_MASK |
                   RTC_IER_TSIE_MASK);
RTC_SR |= RTC_SR_TCE_MASK;
```

- Processamento de Interrupção de alarme
  - RTC_IER |= RTC_IER_TAIE_MASK;
  - NVIC – IRQ20 (Número de exceção 36)
  - Definir a rotina de serviço **RTC_Alarm_IRQHandler**
  - Limpar a bandeira: fazer acesso de escrita a RTC_TAR

Projeto-Exemplo

- Relógio digital que mostra as horas no formato HH:MM:SS na primeira linha, alinhada com a margem esquerda, do LCD. Tem um alarme que “toca” a cada 60s aproximadamente.
  - Fonte de CLK: 1kHz (LPO).
  - O tempo é inicializado em 0.
  - O toque é implementado como “piscada” da mensagem “ALARME!!!” na segunda linha do visor do LCD.

Cômputo de tempo em segundos

- RTC_TSR e RTC_TPR

  Segundos ← (RTC_TSR*32768)/1000+(RTC_TPR/1000);

- Segundos → RTC_TSR e RTC_TPR

  RTC_TPR ← (segundos * 1000)%32768;

  RTC_TSR ← (segundos * 1000)/32768;

- Um segundo corresponde a 1000 *ticks* no registrador RTC_TPR, e não 32768!

Técnicas de Programação

- Sem poder contar com o evento de interrupção de Segundos, como atualizar os horários na base de segundos?
  - Pela técnica *polling*, amostrando periodicamente o tempo corrente.
- Como fazer atualizações no LCD somente quando houver variação nos horários?
  - Comparação entre o valor corrente (seconds) e o valor anterior (seconds_bak).

Controle do alarme

- O registrador RTC_TAR foi projetado para trabalhar com unidade em segundos e ter RTC_TSR como referência.
  - 60s na base de tempo de 1/1000 correspondem a
    - RTC_TPR = 27232
    - RTC_TSR = 1
  - Aproximação: RTC_TAR = 1 (32,768s) ou RTC_TAR = 2 (65,536s)
- Geração de alarme repetitivo em cada 60s
  - Setar em cada i vez de entrada na rotina RTC_IRQHandler (i+1) vezes de (60s) em RTC_TAR: (segundos * 1000 * (i+1))/32768.
- Estado de alarme processado fora de RTC_IRQHandler
  - Passagem do estado de alarme para o fluxo de controle principal

Técnicas de Programação

- Como preservar o número de vezes de entrada numa rotina de serviço durante o tempo de execução de um programa, sem que este número seja visível por outras funções?
  - Uso do qualificador **static** na declaração da variável

    static unsigned int i;
- Como passar a informação do estado de alarme detectado por uma rotina de serviço para o fluxo de controle principal?
  - Uso do qualificador **extern**.

Pseudo-Código da Rotina de Serviço

- **RTC_Alarm_IRQHandler()**
  - Se RTC_SR_TAR estiver levantada
    - Declarar variável i com qualificador static e Alarm, extern
    - Alarm ← Habilitado;
    - Setar alarme para tocar a ~60s novamente

      RTC_TAR ← (60***(++i)***1000)/32768, com i inicializado em 1 .

Técnicas de Programação

- Uso de **operador de incremento e decremento pré-fixado e pós-fixado**
  - Incremento pré-fixado (++i): variável i é incrementada antes de usá-la numa instrução;
  - Incremento pós-fixado (i++): variável i é incrementada após o seu uso numa instrução;
  - Decremento pré-fixado (--i): variável i é decrementada antes de usá-la numa instrução;
  - Decremento pós-fixado (i--): variável i é decrementada após o seu uso numa instrução

Técnicas de Programação

- Modularização de códigos

  Inicialização do módulo RTC

  RTC_init()

  Habilitação de interrupção por eventos AI, OI, II

  RTC_enableNVICAlarm (char priority)

  Ativação da Interrupção por AI

  RTC_enableInterruptAlarm()

  Desativação da Interrupção por AI

  RTC_disableInterruptAlarm()

  Leitura do tempo em segundos

  getTime (unsigned int *seconds)

  Escrita do tempo em segundos

  setTime (unsigned int seconds)

  Escrita do horário de alarme em segundos

  setAlarm (unsigned int seconds)

Pseudo-Código do Fluxo Principal

```
GPIO_initLCD();
initLCD();

RTC_init();
RTC_enableNVICAlarm();

setTime(0);
mostre o horário no visor;
getTime (&seconds);
alarm=DESABILITA;
seconds_bak = seconds;  // salvar o estado anterior
```

Pseudo-Código do Fluxo Principal

Laço:

Se alarm==HABILITA

pisque no visor 5 vezes “ALARME!!!”

Alarm = DESABILITA;

getTime(&seconds);

Se (seconds != seconds_bak)

atualiza o horário no visor;

seconds_bak = seconds;

CodeWarrior IDE Development Suite

Informações Adicionais

- Using RTC module on FRDM-KL25Z

  https://community.nxp.com/t5/Kinetis-Microcontrollers/Using-RTC-module-on-FRDM-KL25Z/ta-p/1127476

- KL25 Sub-Family Reference Manual

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf

  - NVIC: Capítulo 3 (página 51)
  - Clock Distribution: Capítulo 5 (página 123)
  - SIM: Capítulo 12 (página 193)
  - RTC: Capítulo 34 (página 597)