EA871

Temporizadores

Números em glifos

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

Revisão

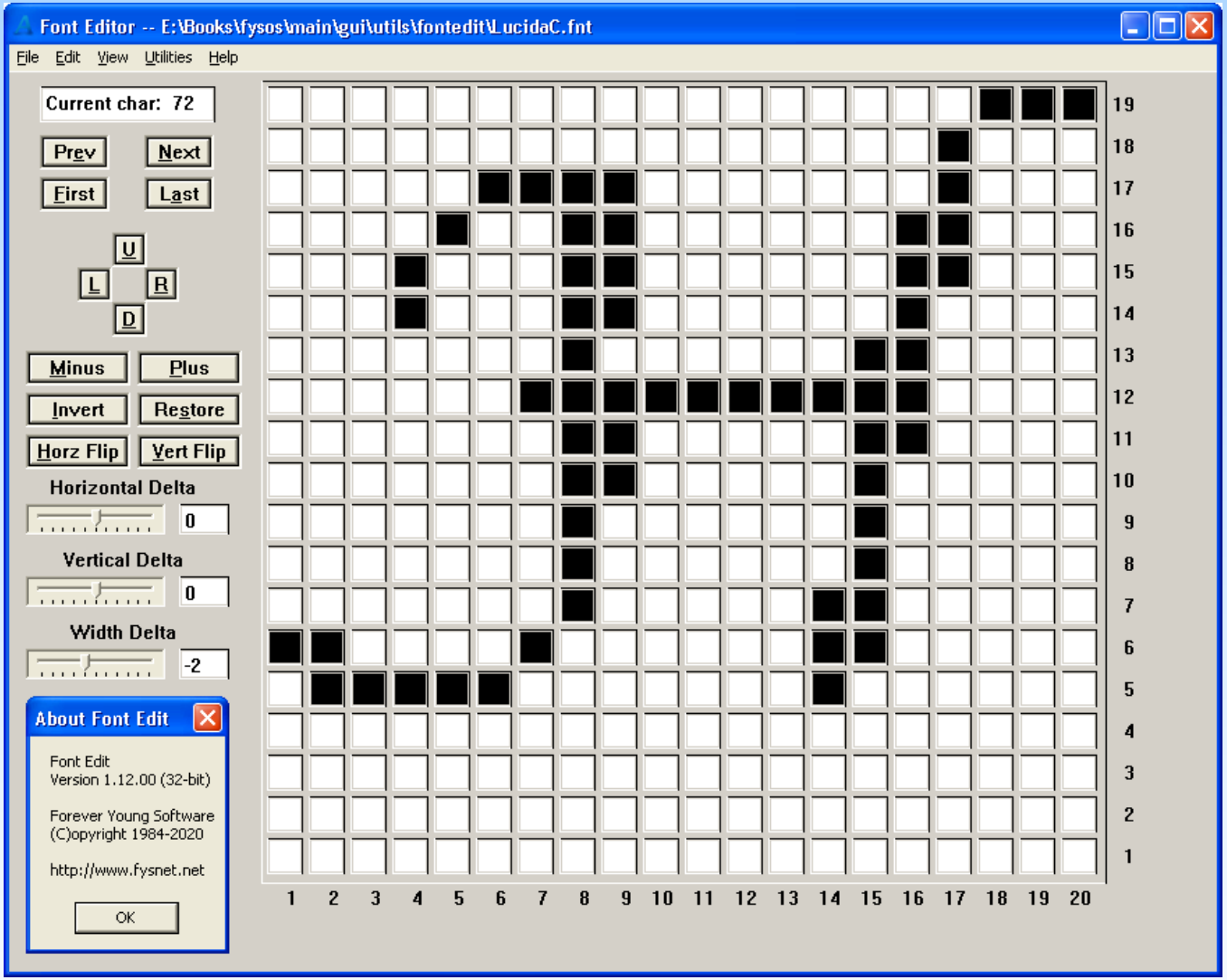
- Números: **sequências de algarismos**.
- **Notação posicional**: a posição ocupada por cada algarismo em um número modifica o seu valor de uma potência da base para cada casa à esquerda.
  - $453 = 4x10^2 + 5x10^1 + 3x10^0$
  - $0xA37C = Ax16^3 + 3x16^2 + 7x16^1 + Cx16^0$
- Algarismos são renderizados como glifos representáveis pelos códigos binários.

Revisão

- Códigos em ASCII (7 *bits*) para algarismos da base hexadecimal

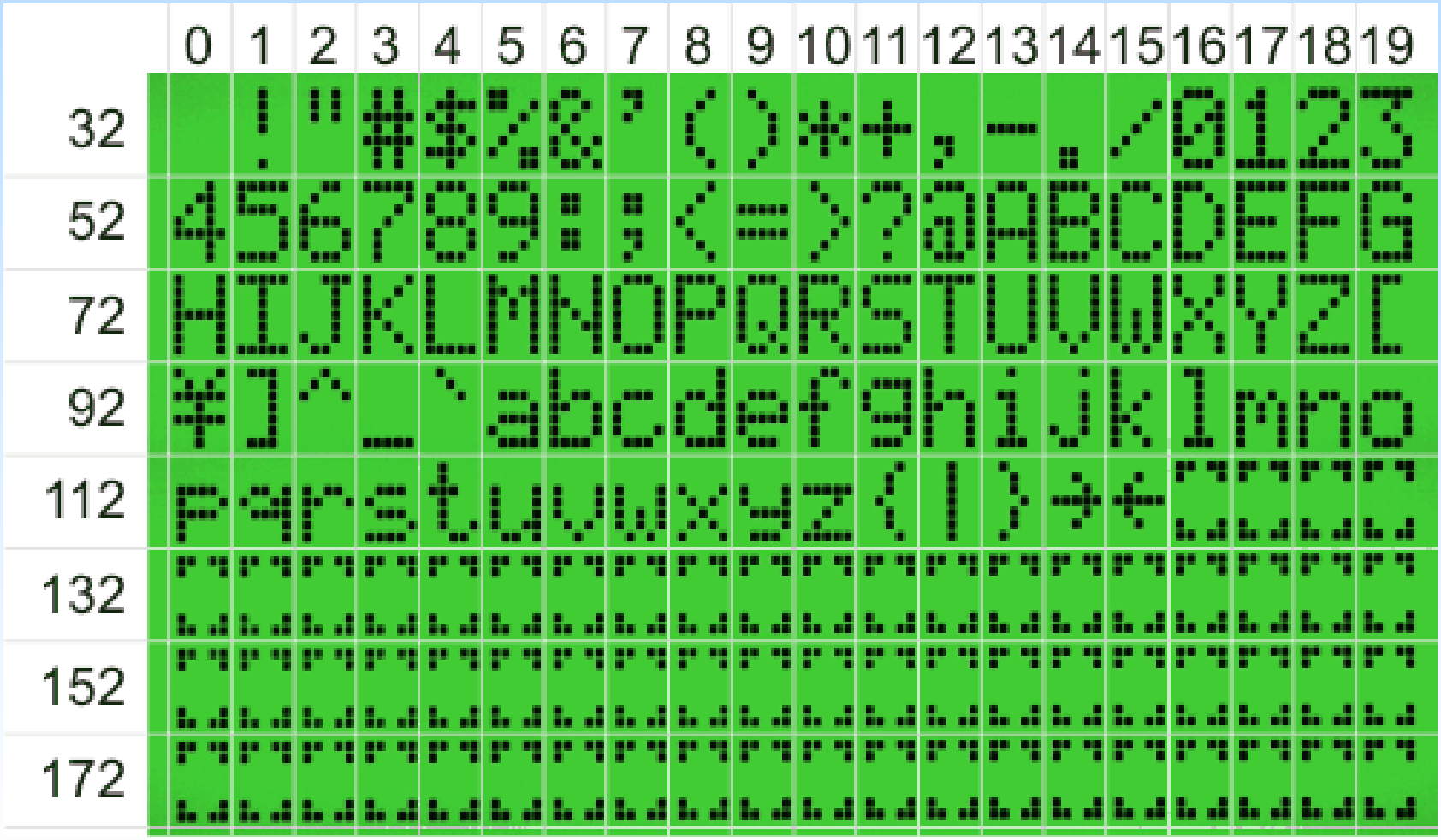
Algarismo	Código binário	glifo	Algarismo	Código binário	glifo
0	011_0000	‘0’	8	011_1000	‘8’
1	011_0001	‘1’	9	011_1001	‘9’
2	011_0010	‘2’	A/a	100_0001	‘A’
3	011_0011	‘3’	B/b	100_0010	‘B’
4	011_0100	‘4’	C/c	100_0011	‘C’
5	011_0101	‘5’	D/d	100_0100	‘D’
6	011_0110	‘6’	E/e	100_0101	‘E’
7	011_0111	‘7’	F/f	100_0110	‘F’

Glifo


Font Editor -- E:\Books\fysos\main\gui\utils\fontedit\LucidaC.fnt
File Edit View Utilities Help
Current char: 72
Prev
Next
First
Last
U
L
R
D
Minus
Plus
Invert
Restore
Horz Flip
Vert Flip
Horizontal Delta
0
Vertical Delta
0
Width Delta
-2
About Font Edit
Font Edit
Version 1.12.00 (32-bit)
Forever Young Software
(C)opyright 1984-2020
http://www.fysnet.net
OK
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

LUT

- Correspondência entre algarismo ↔ glifo: tabela *Look Up* (LUT).



Revisão

Representação simbólica

- Números ↔ sequência de glifos ↔ sequência de dados de 7 *bits* (ASCII) ↔ sequência de dados do tipo char ↔ *strings*.
- Códigos binários dos glifos dos algarismos e das letras seguem a mesma **ordem** numérica/alfabética
  - ‘0’ – ‘9’: Códigos ASCII 0x30 - 0x39
  - ‘A’ – ‘F’: Códigos ASCII 0x41 – 0x46
  - ‘a’ – ‘f’: Códigos ASCII 0x61 – 0x66

Números → *Strings*

- Tipos de dados numéricos
  - Inteiros sem sinal: sequências de algarismos.
  - Inteiros com sinal

    `-` `Inteiro sem sinal`
  - Pontos flutuantes: Composição de 2 inteiros, separados por um ponto/vírgula.

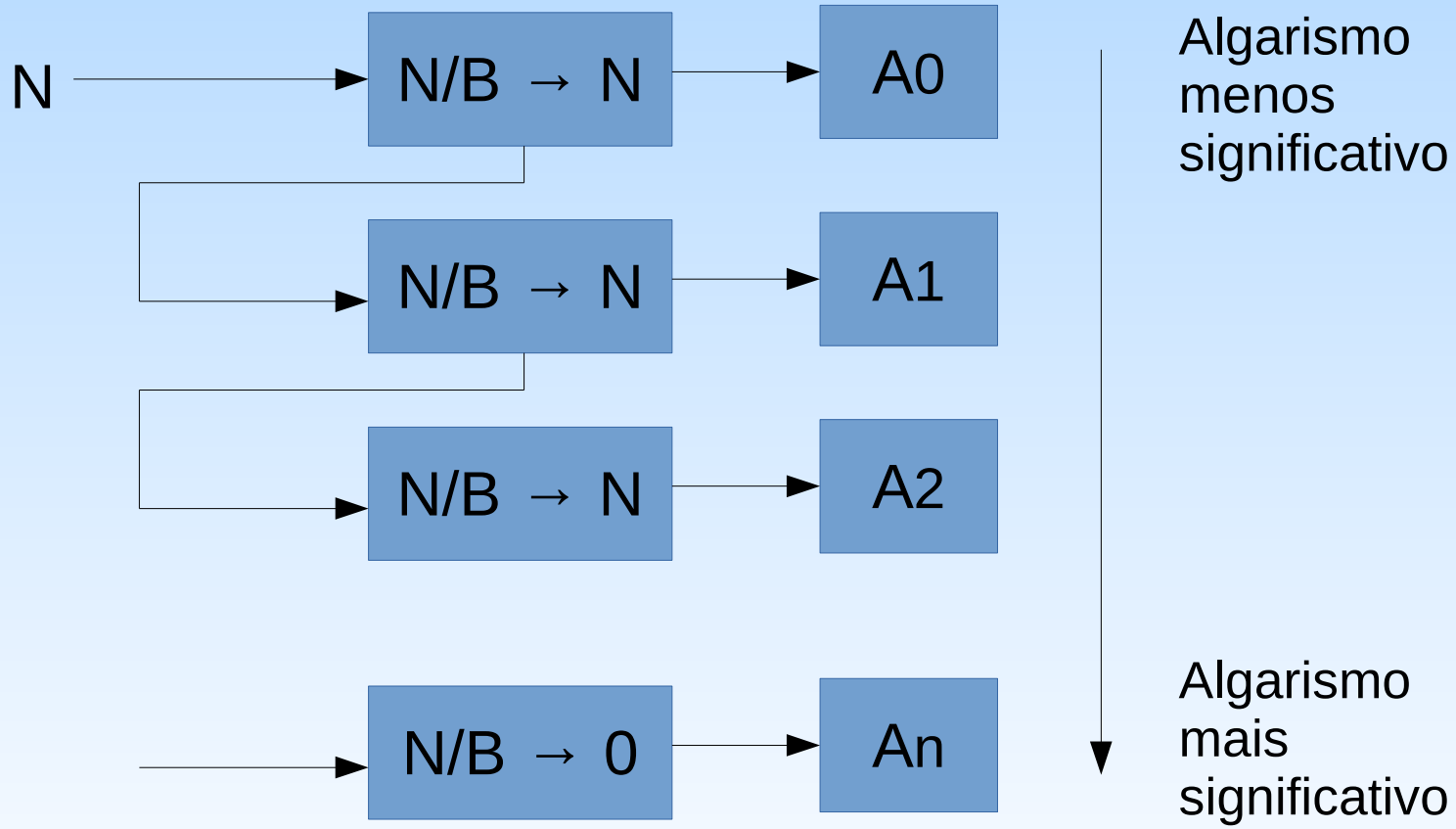
    `Parte inteira` `.` `Parte fracionária`

Inteiros sem sinal → *Strings*

- Extração da sequência de algarismos de um número representado na base B.
- Identificação do código ASCII de cada algarismo.

Inteiros sem sinal → *Strings*

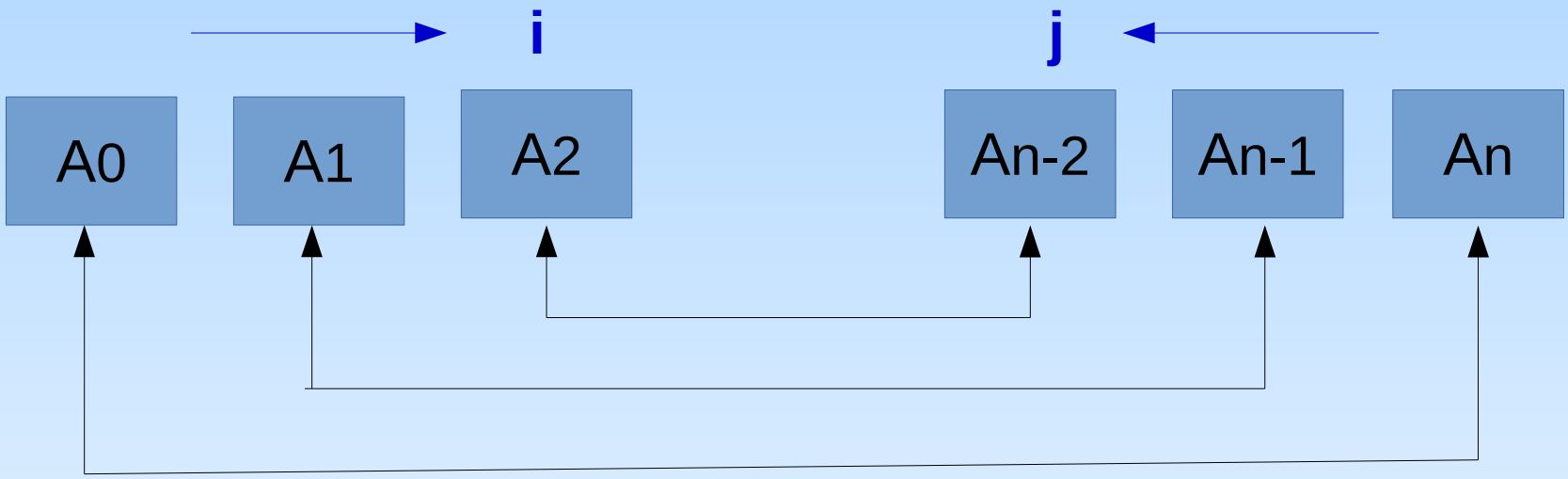
- Extração da sequência de algarismos de N representado na base B.



Sequência invertida!

Inteiros sem sinal → *Strings*

- Reversão da sequência



Troca, par a par, até j < i

Inteiros sem sinal → *Strings*

- Substituição por código ASCII:

0-9:
0x30+valor

10-15:
0x41+(valor-10)
0x61+(valor-10)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Pseudocódigo: itoa(N, B)

Inteiros *N* de base *B*→ *String*

Se N > 0 então M=N; sinal = 0;

senão M=abs(N); sinal = 1;

Inicializa i = 0;

Enquanto M != 0

Extração do algarismo A de cada posição i: A=M%B;

Conversão para ASCII: Se A > 9 então string[i+sinal] = (A-10)+'A';

Senão string[i+sinal] = A+’0’;

M=M/B; i++;

string[i+sinal] = ‘\0’;

- Reversão da sequência entre i=sinal e j=strlen(string)
  - Enquanto j > i

    Troca entre string[i] e string[j];

    I++; j--
- Se sinal == 1 então string[0] = ‘-’;

Pseudocódigo atoi(*string*,B)

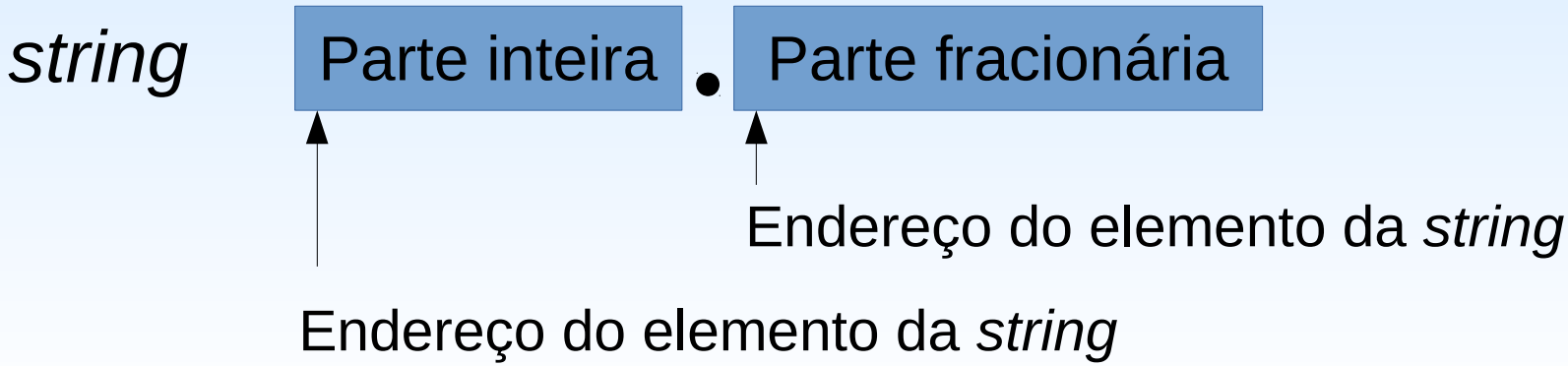
Strings* → Inteiros (sem sinal) *N* de base *B

- Extração do valor numérico de cada caractere da posição i da *string*:
  - Se C < 0x40 então A_i = C-0x30.
  - Senão A_i=C-0x41+10.
- Atribuição do peso do valor (potência da base).
  - A_i *B^i
- Adição de todas as parcelas
  - $resultado \leftarrow \sum_{i=0} A_i B^i$

Disponível na biblioteca de C: stdlib.h

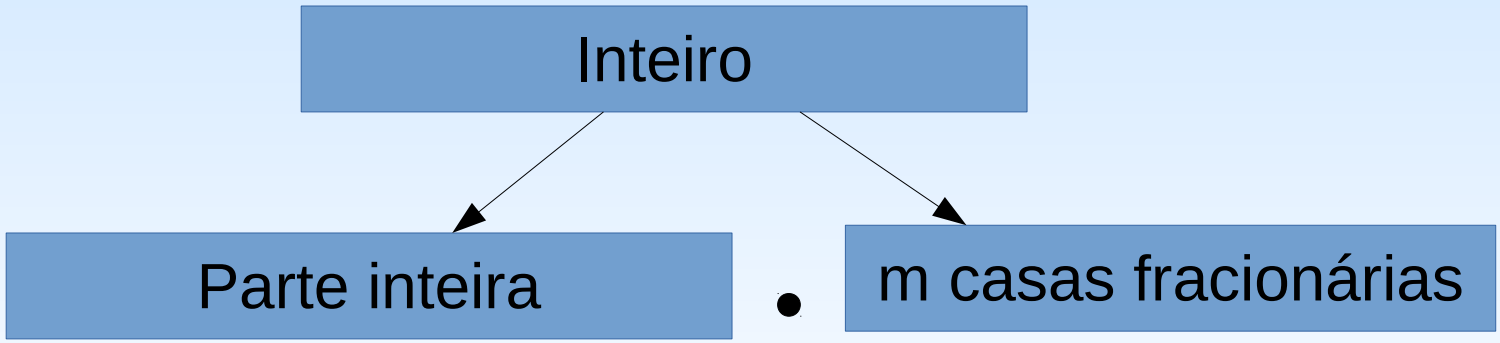
Pontos Flutuantes → *Strings*

- Separação parte inteira e parte fracionária de N
  - Parte Inteira: N/1
  - Parte fracionária = N – Parte inteira
- Aproximação da parte fracionária para a parte inteira.
- Conversão da parte inteira para *string.*
- Inserção de ponto decimal.
- Inserção de ‘0’ até algarismo diferente de 0 na parte fracionária
- Conversão da parte fracionária para *string*.



Pontos Flutuantes → *Strings*

- Multiplicação de N por B^m, onde m é a quantidade de casas fracionárias desejada.
- Descarte da parte fracionária de $N*B^m$ transformando-o num inteiro.
- Conversão do inteiro para *string*.
- Inserção do ponto/vírgula antes da m-ésima casa do inteiro.



Strings → Pontos Flutuantes

- atof (char *str)
- Disponível na biblioteca de C
  - Protótipo em stdlib.h

CodeWarrior IDE Development Suite

Informações Adicionais

- C library function – atoi()

  https://www.tutorialspoint.com/c_standard_library/c_function_atoi.htm
- Write your own atoi()

  https://www.geeksforgeeks.org/write-your-own-atoi/
- atol(), atoll and atof functions in C/C++

  https://www.geeksforgeeks.org/atol-atoll-and-atof-functions-in-c-c/
- Implement your own itoa()

  https://www.geeksforgeeks.org/implement-itoa/
- Convert a floating point number to string in C

  https://www.geeksforgeeks.org/convert-floating-point-number-string/

Informações Adicionais

- Converting an Integer to a String in Any Base

  https://runestone.academy/runestone/books/published/pythonds/Recursion/pythondsConvertinganIntegertoaStringinAnyBase.html

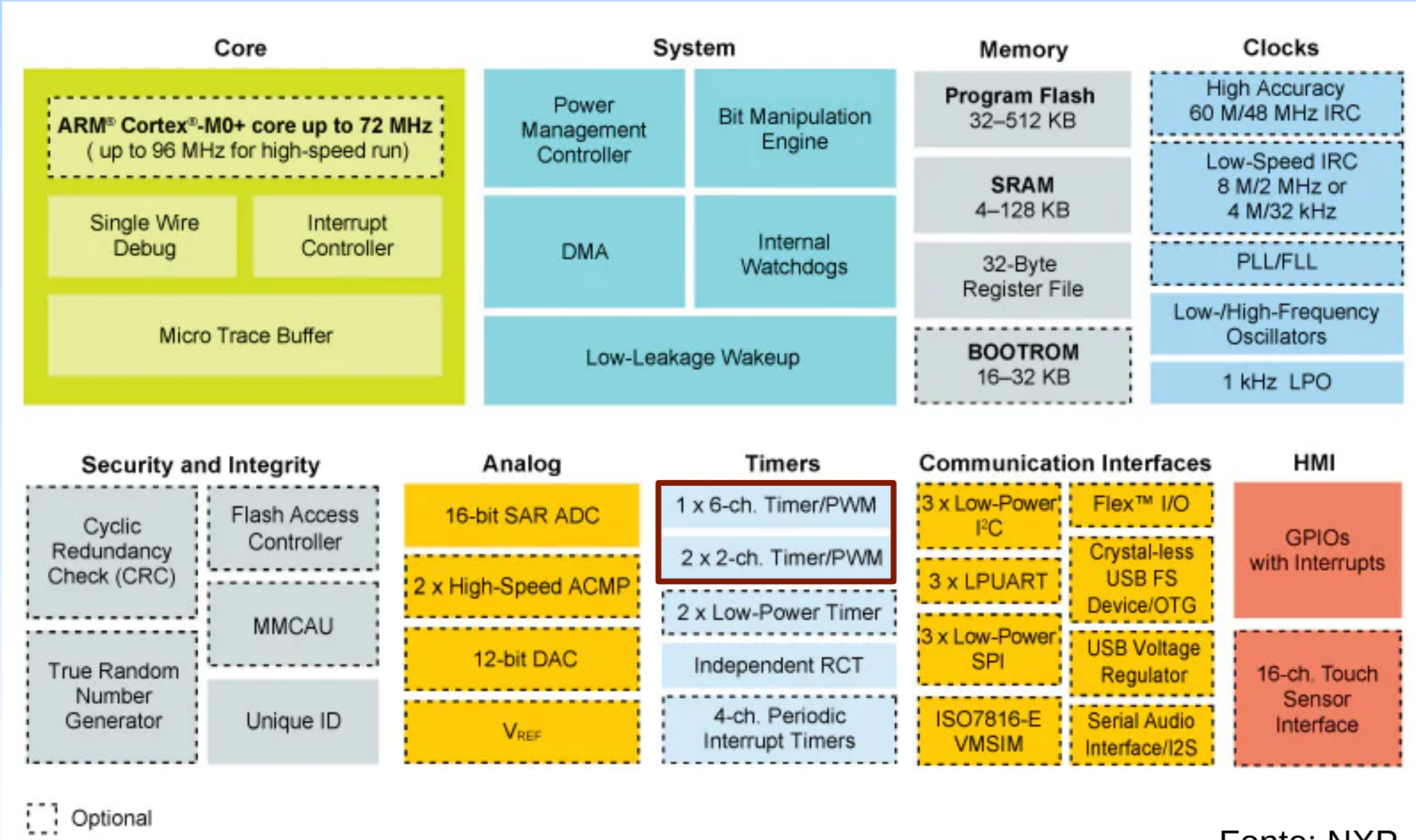
EA871

Temporizadores

TPM (*Timer - PWM*)

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

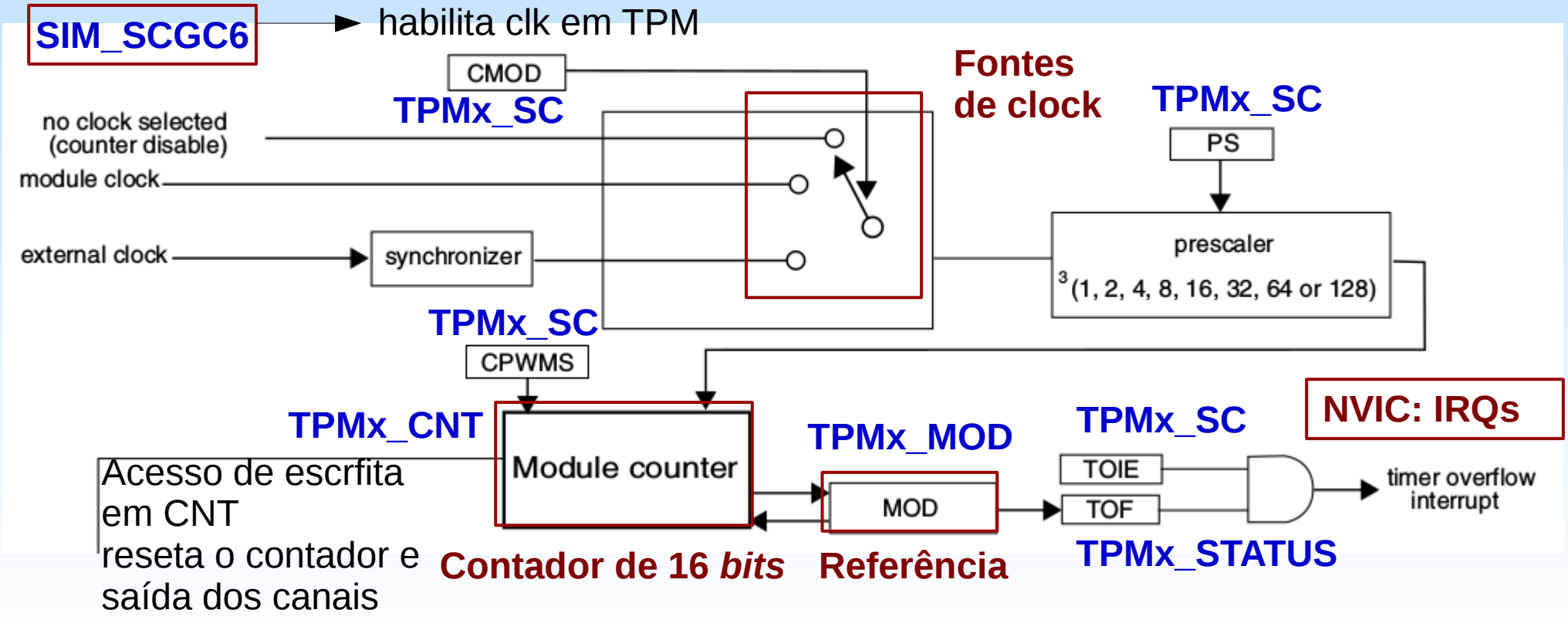
Microcontrolador Kinetis KL25Z



Fonte: NXP

TPM

- **Módulo *Timer/PWM*** (*Pulse Width Modulation*): um temporizador de 16 *bits*, com um contador configurável para contagem crescente ou crescente-decrescente, e 3 funções integradas – *input capture* (captura da entrada), *output compare* (comparação da saída) e geração de sinais PWM.



TPMx_SC

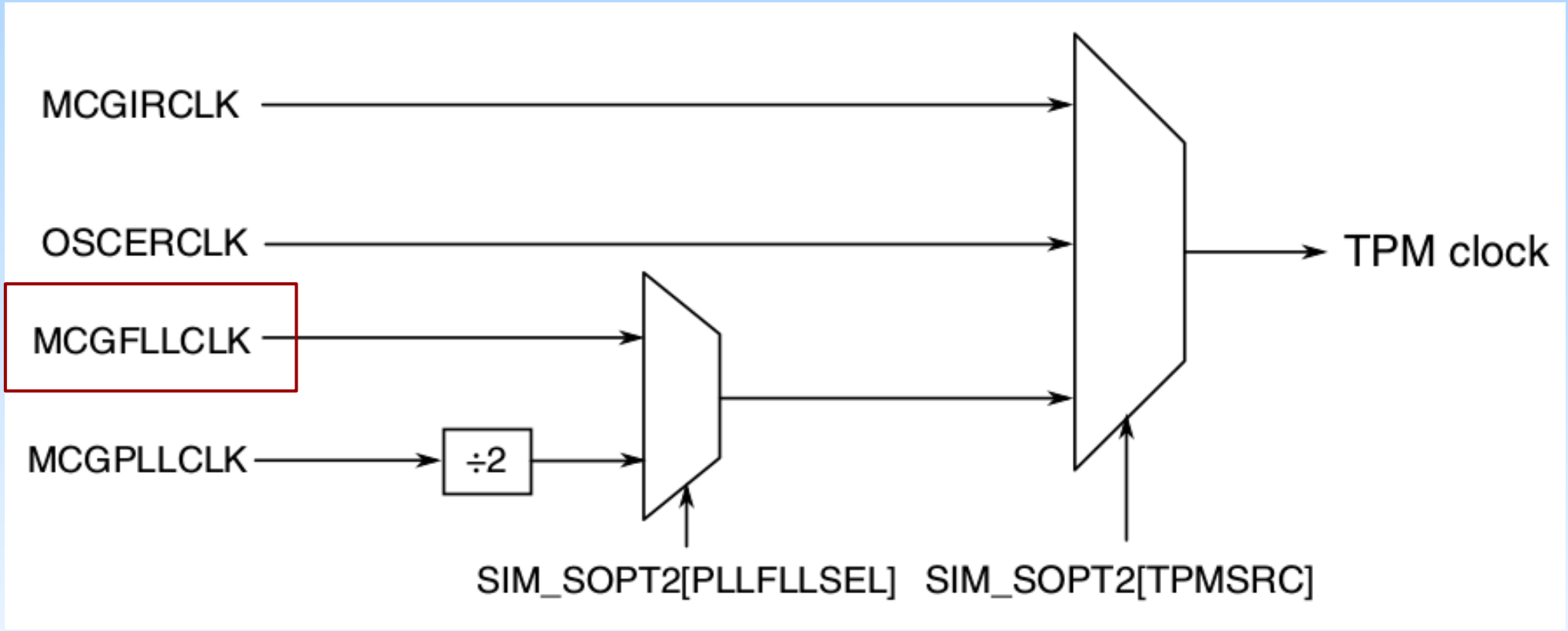
Address: Base address + 0h offset

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

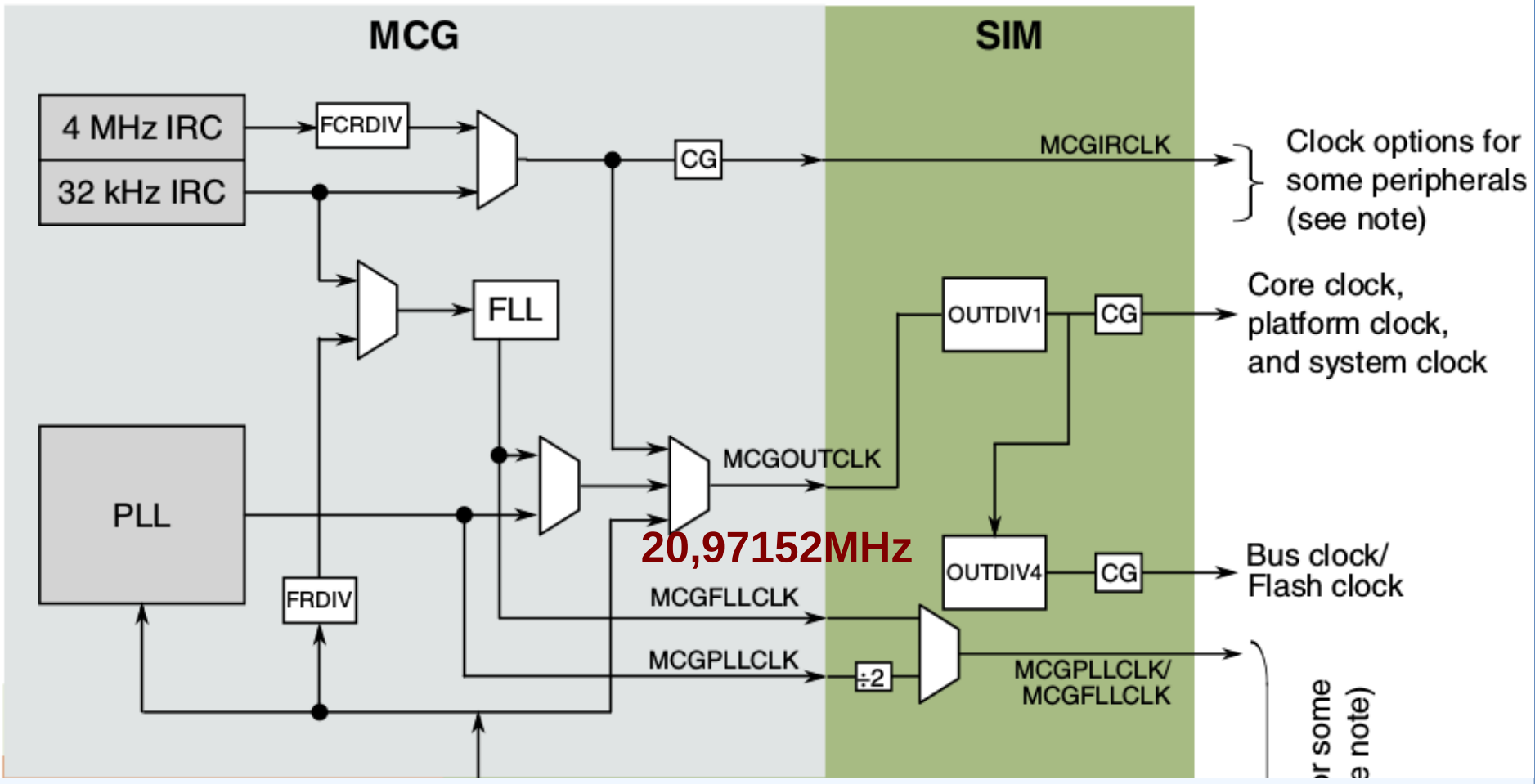
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							DMA	TOF	TOIE	CPWMS	CMOD		PS		
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1-128

Fontes de CLK



MCGFLLCLK


MCG
SIM
4 MHz IRC
32 kHz IRC
FCRDIV
CG
MCGIRCLK
Clock options for some peripherals (see note)
FLL
OUTDIV1
CG
Core clock, platform clock, and system clock
PLL
MCGOUTCLK
20,97152MHz
FRDIV
OUTDIV4
CG
Bus clock/ Flash clock
MCGFLLCLK
MCGPLLCLK
÷2
MCGPLLCLK/ MCGFLLCLK

IRQs

0x0000_0084	33	17	4	TPM0	
0x0000_0088	34	18	4	TPM1	
0x0000_008C	35	19	4	TPM2	

Address: Base address + 50h offset

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							TOF	0		CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W								w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPMx_STATUS field descriptions

Canais

Table 3-37. TPM configuration

TPM instance	Number of channels	Features/usage
TPM0	6	Basic TPM,functional in Stop/VLPS mode
TPM1	2	Basic TPM,functional in Stop/VLPS mode
TPM2	2	Basic TPM,functional in Stop/VLPS mode

Address: Base address + 84h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				TRGSEL				0					CROT	CSOO	CSOT
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						GTBEEN	0	DBGMODE		DOZEEN	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPMx_CONF field descriptions

Canais

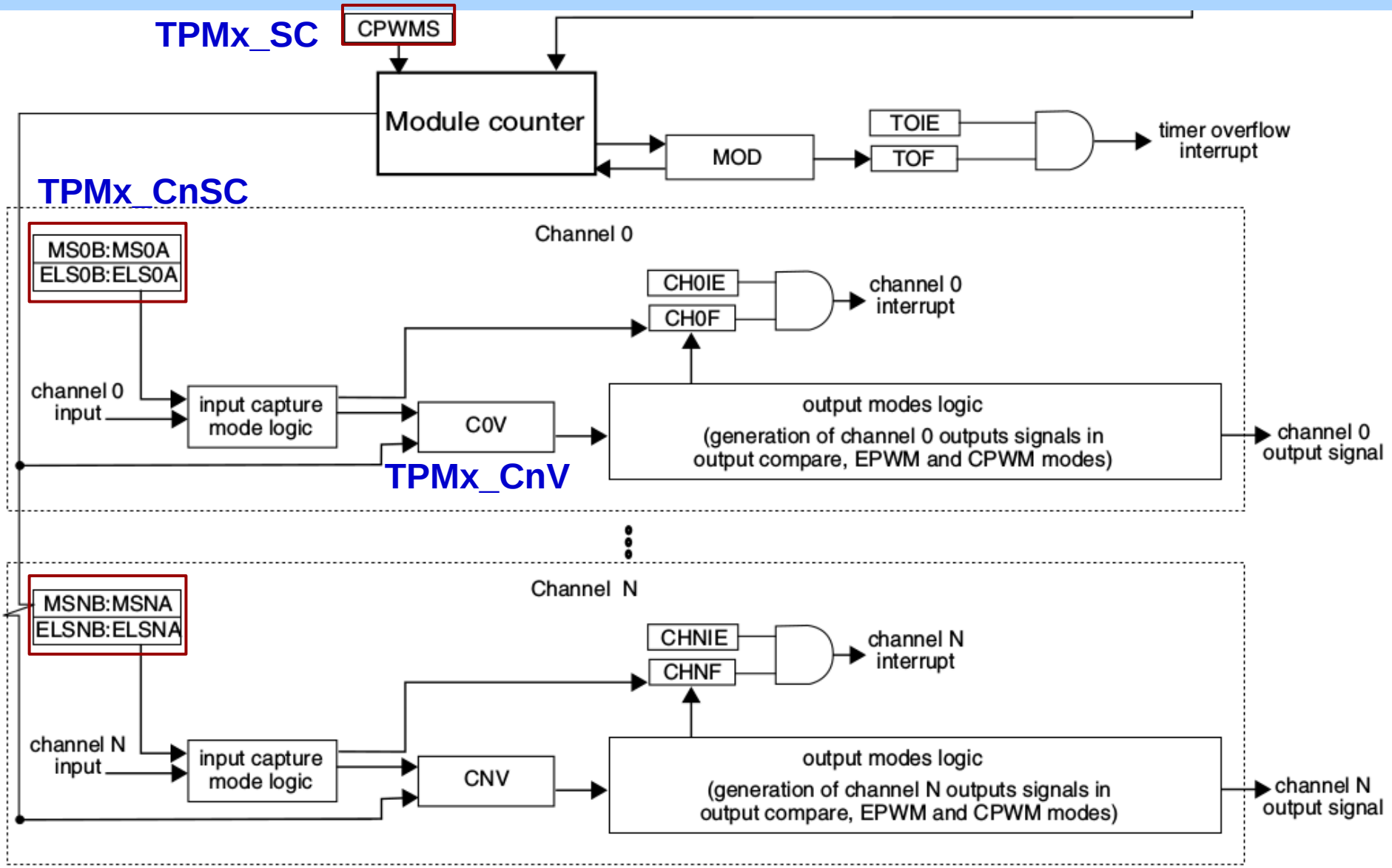

TPMx_SC
CPWMS
Module counter
MOD
TOIE
TOF
timer overflow interrupt
TPMx_CnSC
Channel 0
MS0B:MS0A
ELS0B:ELS0A
CH0IE
CH0F
channel 0 interrupt
channel 0 input
input capture mode logic
C0V
output modes logic
(generation of channel 0 outputs signals in output compare, EPWM and CPWM modes)
channel 0 output signal
TPMx_CnV
Channel N
MSNB:MSNA
ELSNB:ELSNA
CHNIE
CHNF
channel N interrupt
channel N input
input capture mode logic
CNV
output modes logic
(generation of channel N outputs signals in output compare, EPWM and CPWM modes)
channel N output signal

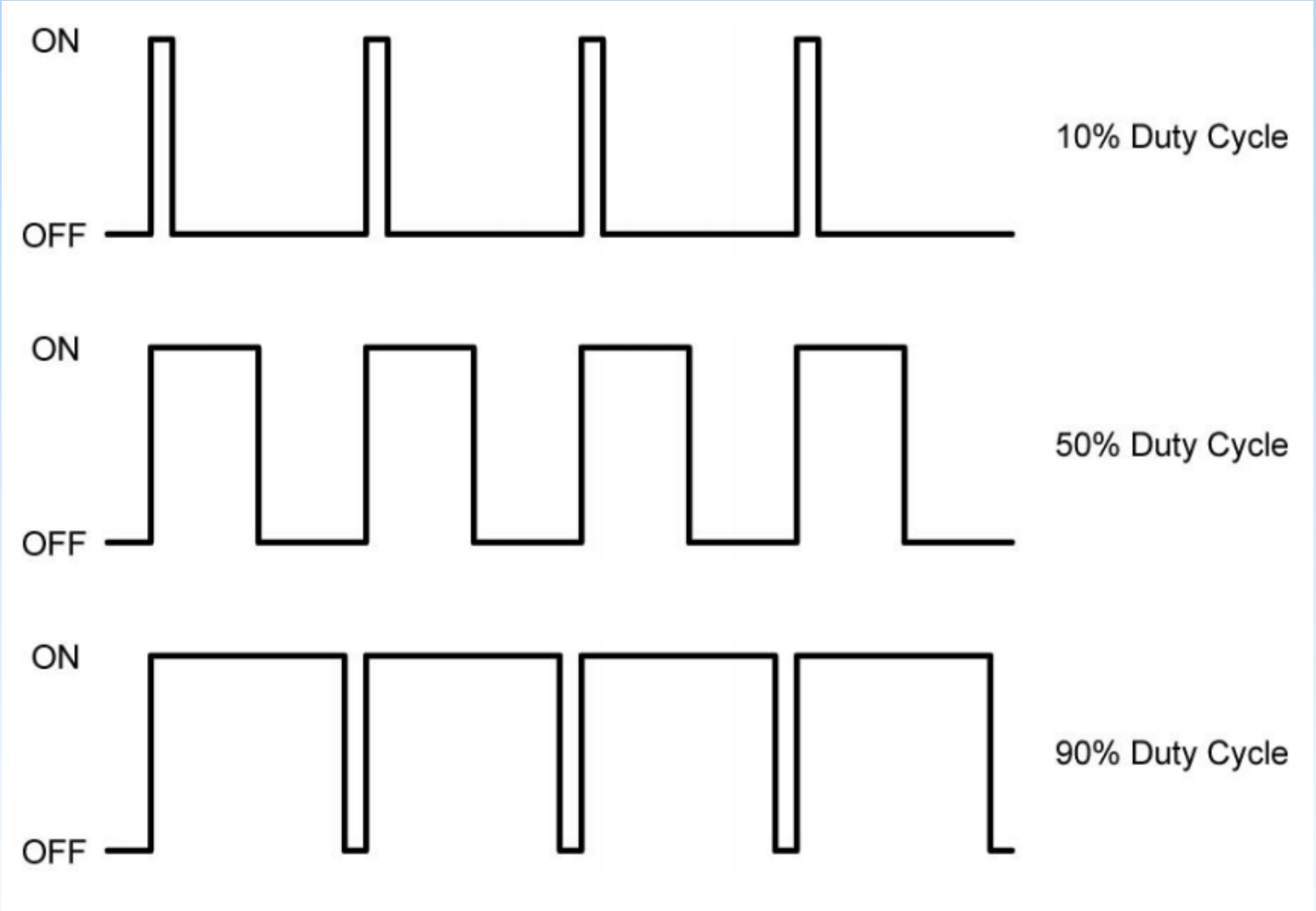
Table 31-34. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	00	00	None	Channel disabled
X	01/10/11	00	Software compare	Pin not used for LPTPM
0	00	01	Input capture	Capture on Rising Edge Only
		10		Capture on Falling Edge Only
		11		Capture on Rising or Falling Edge
	01	01	Output compare	Toggle Output on match
		10		Clear Output on match
		11		Set Output on match
	10	10	Edge-aligned PWM	High-true pulses (clear Output on match, set Output on reload)
		X1		Low-true pulses (set Output on match, clear Output on reload)
	11	10	Output compare	Pulse Output low on match
		X1		Pulse Output high on match
1	10	10	Center-aligned PWM	High-true pulses (clear Output on match-up, set Output on match-down)
		X1		Low-true pulses (set Output on match-up, clear Output on match-down)

Revisão

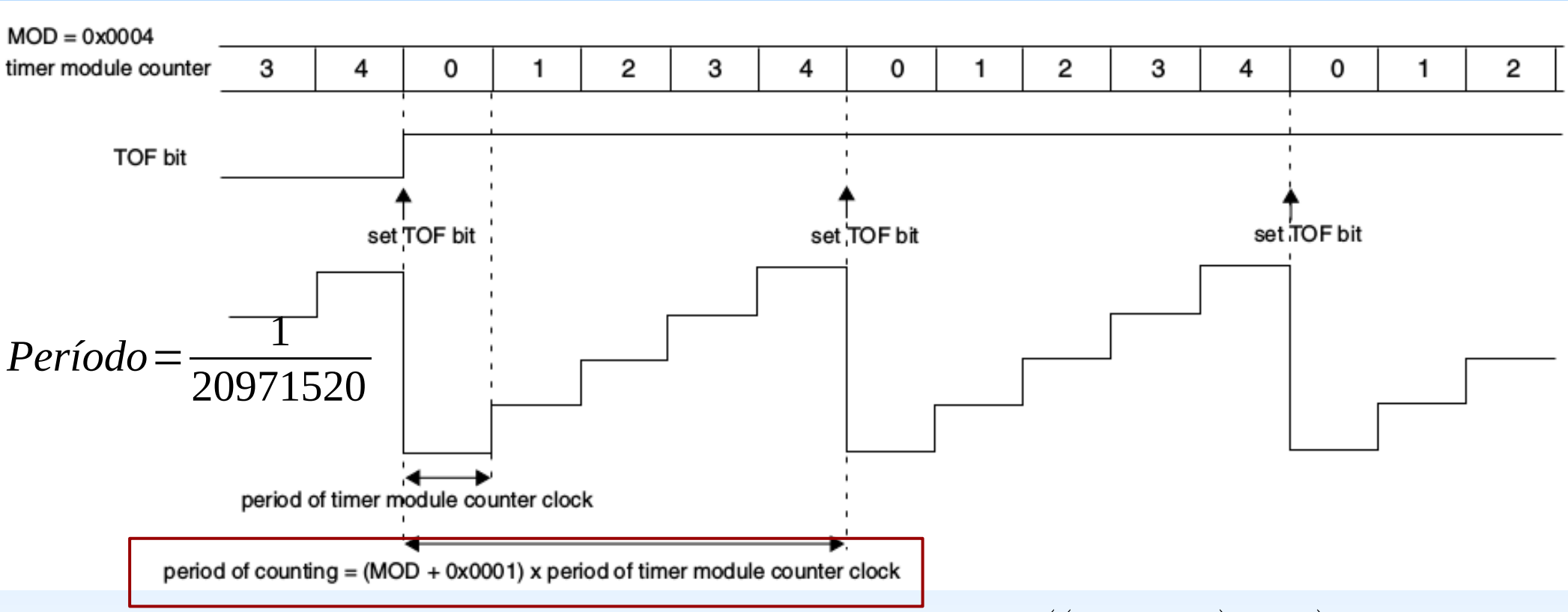
- **Modulação por Largura de Pulso** (*Pulse Width Modulation*): pulsos de larguras variadas modulados por intervalos de tempo regulares, a uma frequência portadora da modulação.
- **Ciclo de trabalho** (*duty cycle*): razão cíclica do tempo em que uma carga/um circuito está ON em comparação com o período de modulação.
- **Aplicações:**
  - Transferência de informação: valor de dado mapeado em largura de pulso.
  - Transferência de potência: valor médio de potência/velocidade mapeado em largura de pulso.
  - Regulação de tensão: nível de tensão mapeado em largura de pulso.

Pulsos de Diferentes Larguras



TPM: Implementação de PWM

- Contagem Crescente (*Up*): CPWMS = 0



$$Período = \frac{1}{20971520}$$

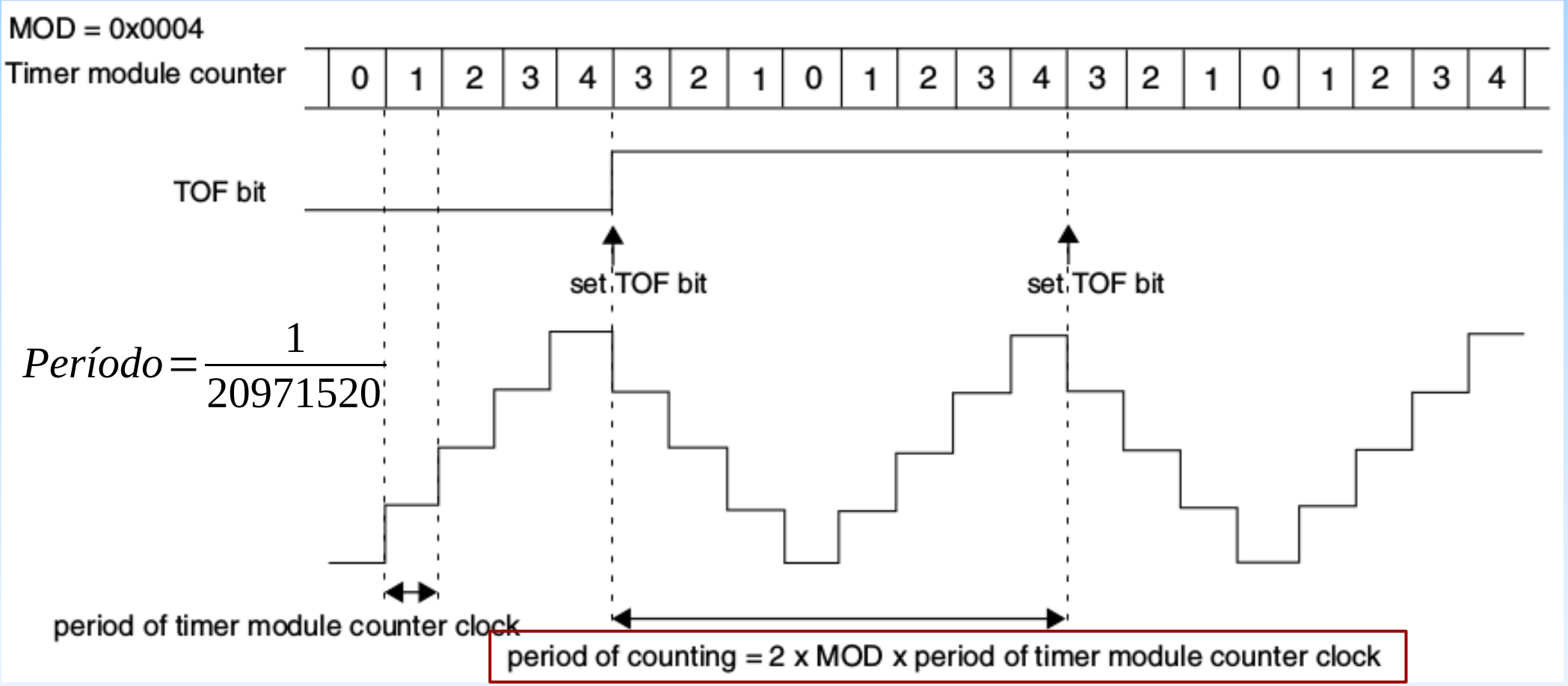
$MOD = 1023$

$PS = 32$ $\quad Período\ de\ contagem = \frac{((1023+1)\times 32)}{20971520} = 1{,}5625\ ms$

$PS = 128$ $\quad Período\ de\ contagem = \frac{((1023+1)\times 128)}{20971520} = 6{,}25\ ms$

TPM: Implementação de PWM

- Contagem Crescente-Decrescente (*Up-Down*): CPWMS=1



$$Período = \frac{1}{20971520}$$

$$MOD = 1023$$

$$PS = 32 \qquad Período\ de\ contagem = \frac{((2 \times 1023) \times 32)}{20971520} = 3{,}122\ ms$$

$$PS = 128 \qquad Período\ de\ contagem = \frac{((2 \times 1023) \times 128)}{20971520} = 12{,}488\ ms$$

TPM: Implementação de PWM

- Contagem Crescente (*Up*)

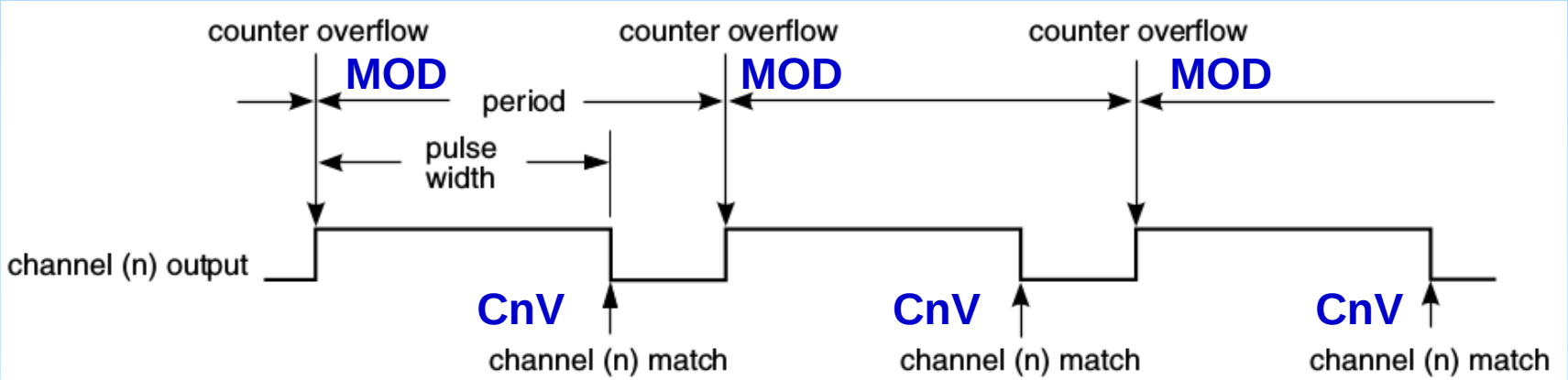


Figure 31-85. EPWM period and pulse width with ELSnB:ELSnA = 1:0

- Contagem Crescente-Decrescente (*Up-Down*)

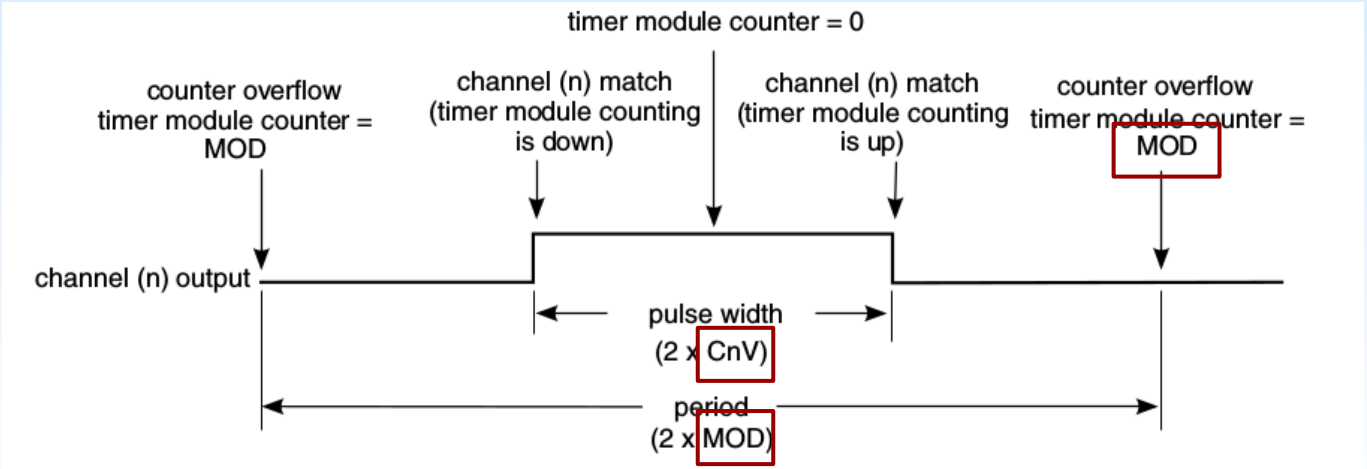


Figure 31-88. CPWM period and pulse width with ELSnB:ELSnA = 1:0

TPM: Canal PWM

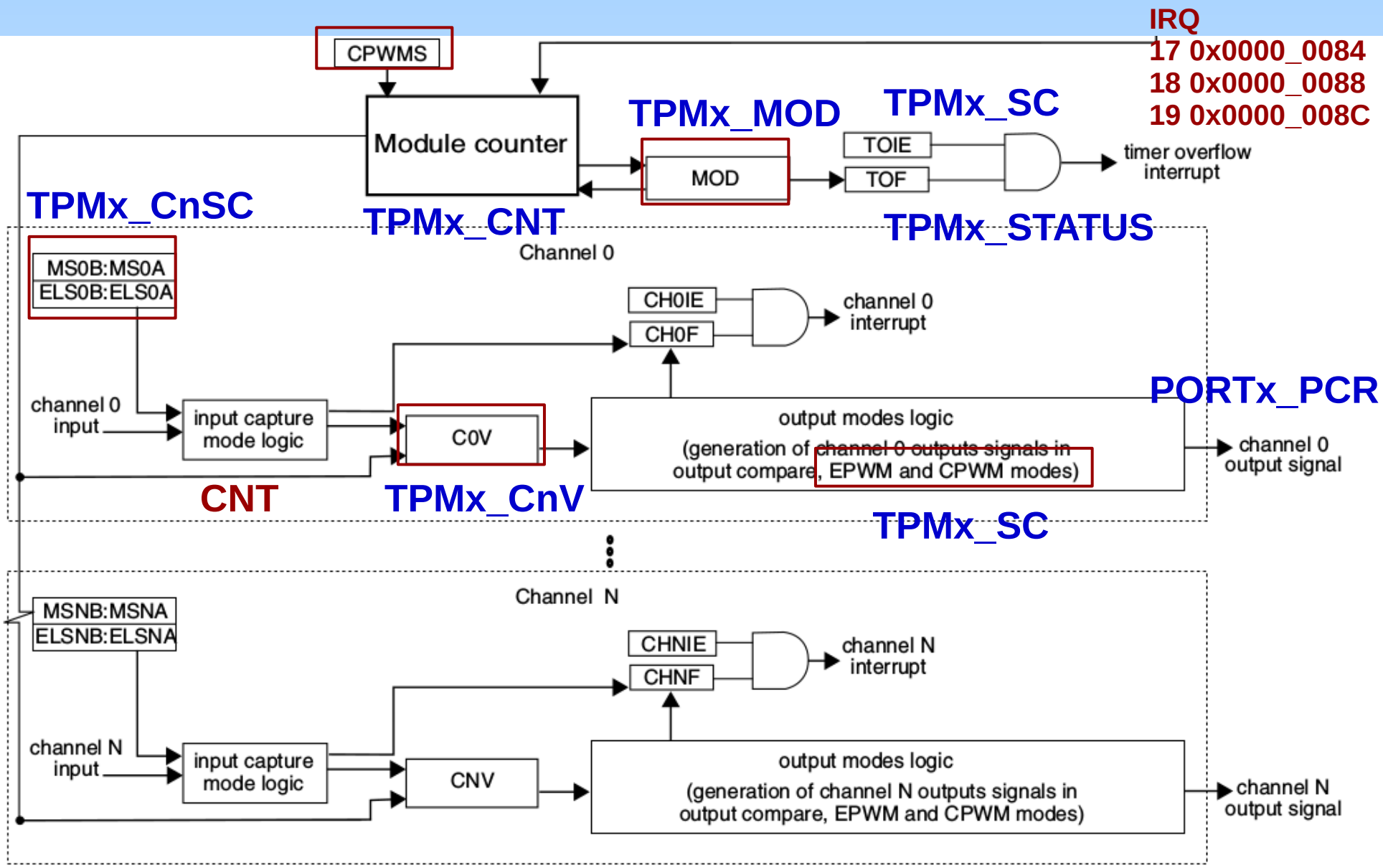


Table 31-34. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	00	00	None	Channel disabled
X	01/10/11	00	Software compare	Pin not used for LPTPM
0	00	01	Input capture	Capture on Rising Edge Only
		10		Capture on Falling Edge Only
		11		Capture on Rising or Falling Edge
	01	01	Output compare	Toggle Output on match
		10		Clear Output on match
		11		Set Output on match
	10	10	Edge-aligned PWM	High-true pulses (clear Output on match, set Output on reload)
		X1		Low-true pulses (set Output on match, clear Output on reload)
	11	10	Output compare	Pulse Output low on match
		X1		Pulse Output high on match
1	10	10	Center-aligned PWM	High-true pulses (clear Output on match-up, set Output on match-down)
		X1		Low-true pulses (set Output on match-up, clear Output on match-down)

PWM: Polaridade

- *High-true pulse* na contagem crescente (CPWMS=0)

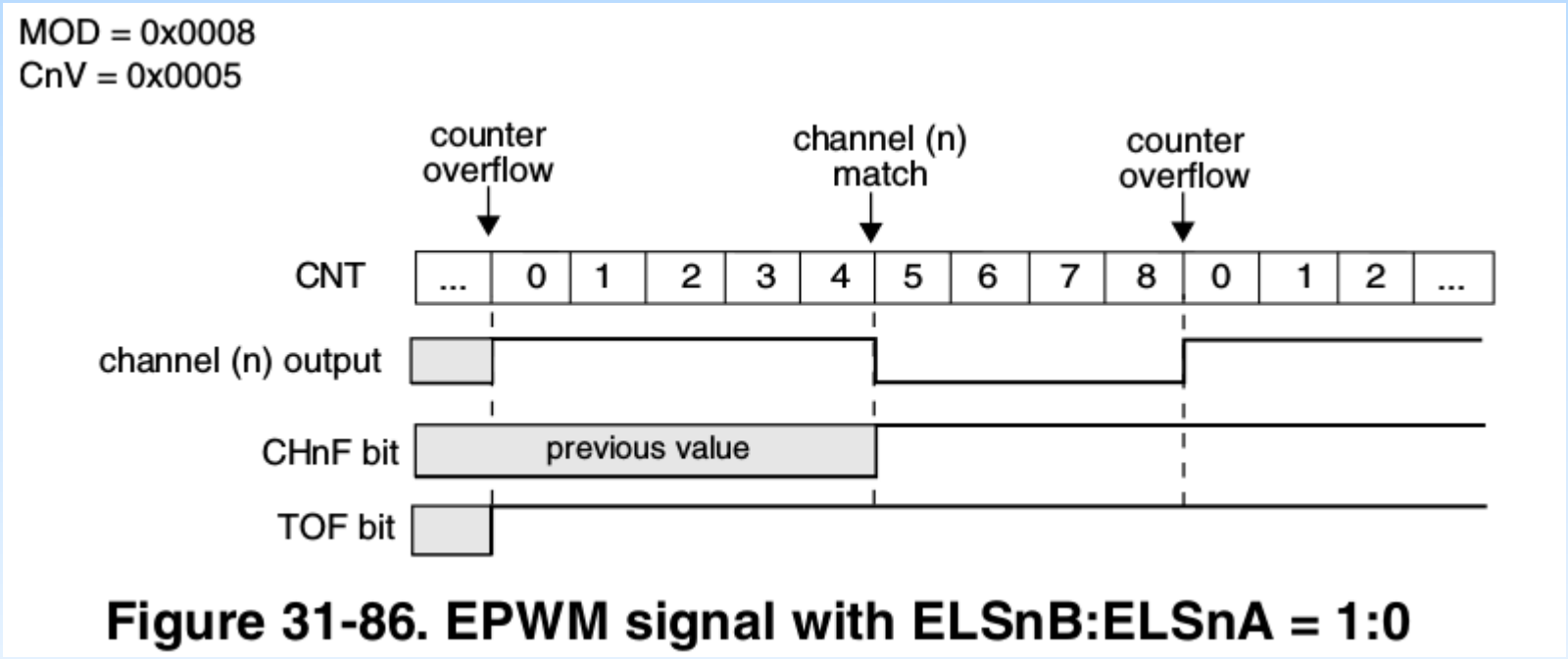


Figure 31-86. EPWM signal with ELSnB:ELSnA = 1:0

PWM: Polaridade

- *Low-true pulse* na contagem crescente (CPWMS=0)

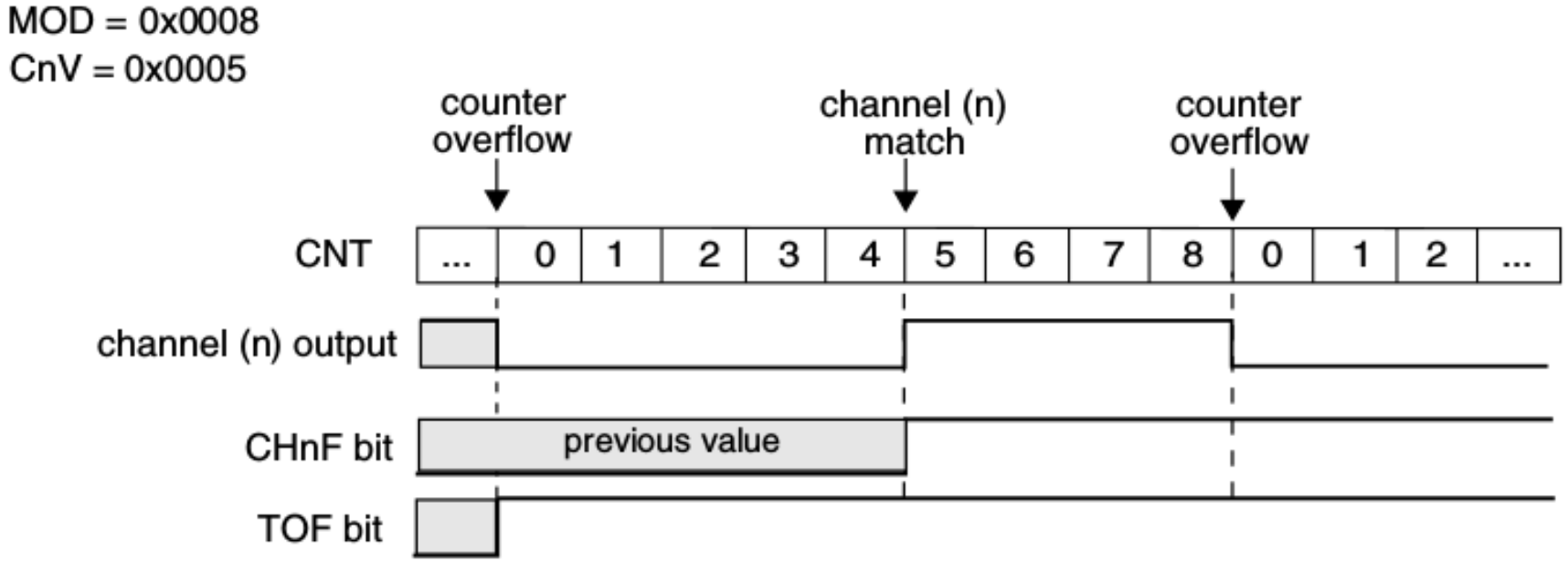


Figure 31-87. EPWM signal with ELSnB:ELSnA = X:1

PWM: Polaridade

- *High-true pulse* na contagem UP-DOWN (CPWMS=1)

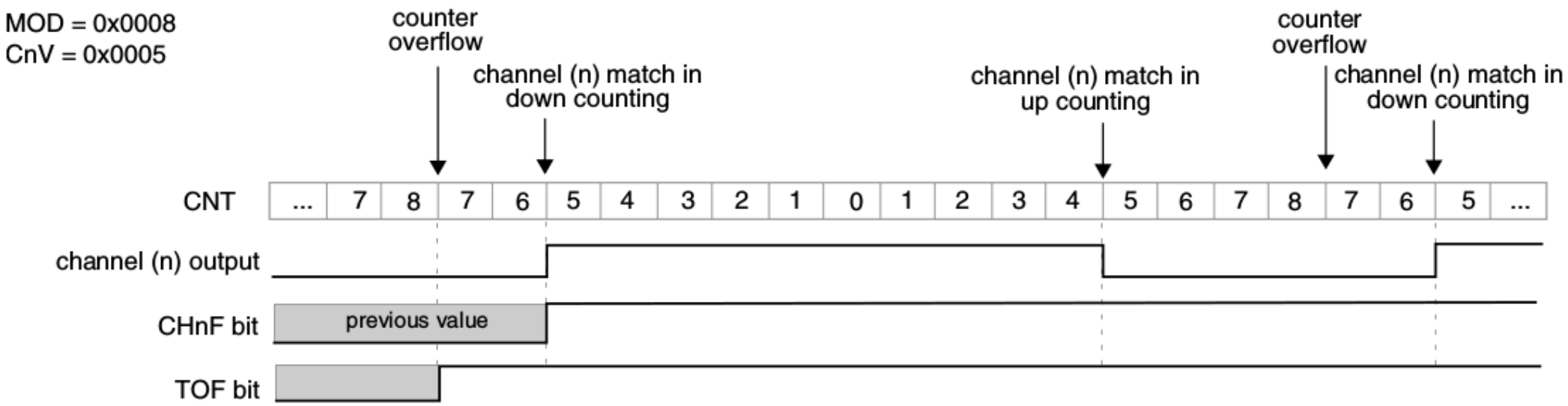


Figure 31-89. CPWM signal with ELSnB:ELSnA = 1:0

PWM: Polaridade

- *Low-true pulse* na contagem UP-DOWN (CPWMS=1)

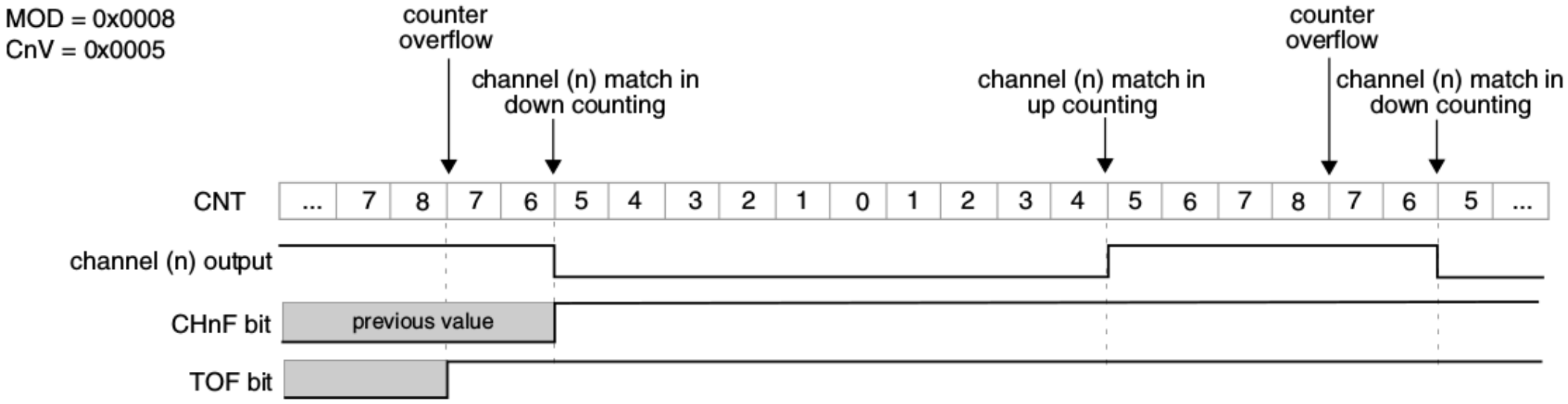


Figure 31-90. CPWM signal with ELSnB:ELSnA = X:1

Registradores

- SIM_SOPT2: registrador de seleção de fonte de clk
- SIM_SCGC6: registrador para habilitar clk do módulo TPM
- SIM_SCGC5: registrador para habilitar porta dos pinos de saída

- PORTx_PCRn: configuração do pino n da PORTx

- TPMx_SC: registrador de controle e de estado do módulo x
- TPMx_MOD: registrador de referência (valor máximo de contagem ou módulo de contagem)
- TPMx_CNT: contador
- TPMx_CnSC: registrador de estado e de configuração do canal n para PWM
- TPMx_CnV: registrador de largura de pulso
- TPMx_STAUTS: registrador de todos os estados
- TPMx_CONF: registrador de configuração de modo de operação do TPM em diferentes estados

Programação de PWM

- **Inicialização do TPM**

```
//Configurar e habilitar fonte de clk
SIM_SOPT2 |= SIM_SOPT2_TPMSRC(0b01);
SIM_SOPT2 &= ~SIM_SOPT2_PLLFLLSEL_MASK;
SIM_SCGC6 |= SIM_SCGC6_TPM2_MASK;
//Configurar TPM
TPM2_SC &= ~(TPM_SC_DMA_MASK | TPM_SC_TOIE_MASK |
TPM_SC_CPWMS_MASK);                    //modo de contagem
TPM2_MOD &= TPM_MOD_MOD(modulo);    // periodo/modulo
TPM2_CNT |= TPM_CNT_COUNT(0x0);        // resetar
TPM2_SC |= (TPM_SC_CMOD(0x1) |         // habilitar modo de clocking
TPM_SC_PS(prescaler) )                 // com prescaler especificado
```

Programação de PWM

- **Inicialização do canal com função PWM**

```
//Inicializar pino de saida
SIM_SCGC5 |= SIM_SCGC5_PORTx_MASK;
 PORTx_PCRn |= (PORT_PCR_ISF_MASK |
                        PORT_PCR_MUX(ALT) );
//Configurar canal para PWM
TPMx_CnSC &= ~(TPM_CnSC_MSB_MASK |  TPM_CnSC_MSA_MASK
| TPM_CnSC_ELSB_MASK | TPM_CnSC_ELSA_MASK );
TPMx_CnSC |= (TPM_CnSC_CHF_MASK);
TPMx_CnSC &= ~(TPM_CnSC_CHIE_MASK |
TPM_CnSC_DMA_MASK);
TPMx_CnV = TPM_CnV_VAL(largura);
TPMx_CnSC |= (TPM_CnSC_MSB_MASK | TPM_CnSC_ELSB_MASK );
```

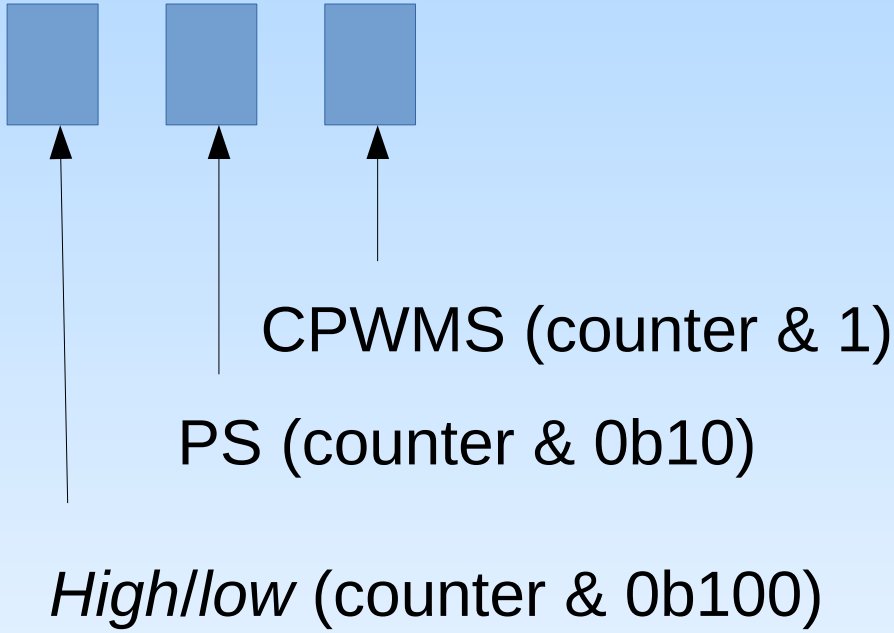
Projeto-Exemplo

- Controlar ciclicamente a luminosidade do *led* azul de resolução 1024, aplicando todas as possíveis combinações de polaridade*, contagem crescente e contagem crescente-decrescente, e* prescaler 32 e 128.
  - *Led* azul: PORTE_PCR23 (PTE23/TPM2_CH1)
  - *Ciclos de trabalho*: nas razões de 0 a 1,1 num espaçamento de 0.2
  - *MOD* (Referência): 1023

Técnicas de Programação

- Uma variável de controle para 3 estados binários:

counter



Técnicas de Programação

```
typedef enum counting_mode_tag {
    UP,
    UP_DOWN
} counting_mode_type;

typedef enum plarity_tag {
    LOW,
    HIGH
} polarity_type;
```

Técnicas de Programação

- Modularização de códigos

  TPM2_initLedB()

  TPM2_pulseLedB (logic_level_type level)

  TPM2_cpwmsLedB (counting_mode_type mode)

  TPM2_psLedB (char ps)

  TPM2_valLedB(unsigned short valor, polarity_type mode);

Pseudocódigo

TPM2_initLedB();

counter = 0;

Laço:

Se (counter & 0b10) TPM2_psLedB(0b111);

senão TPM2_psLedB(0b101);

TPM2_cpwmsLedB(counter & 1);

Laço de diferentes valores:

TPM2_valLedB(valor, (counter &0b100 >> 2))

counter++.

CodeWarrior IDE Development Suite

Informações Adicionais

- KL25 Sub-Family Reference Manual

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

  - Chip Configuration: Capítulo 3 (página 84)
  - Clock Distribution: Capítulo 5 (página 124)
  - PORT: Capítulo 11 (página 184)
  - TPM: Capítulo 31 (página 547)

EA871

Temporizadores

TPM (*Input Capture* e *Output Compare*)

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

TPM

- **Módulo *Timer/PWM*** (*Pulse Width Modulation*): um temporizador de 16 *bits*, com um contador configurável para contagem crescente ou crescente-decrescente, e 3 funções integradas – *input capture* (captura da entrada), *output compare* (comparação da saída) e geração de sinais PWM.

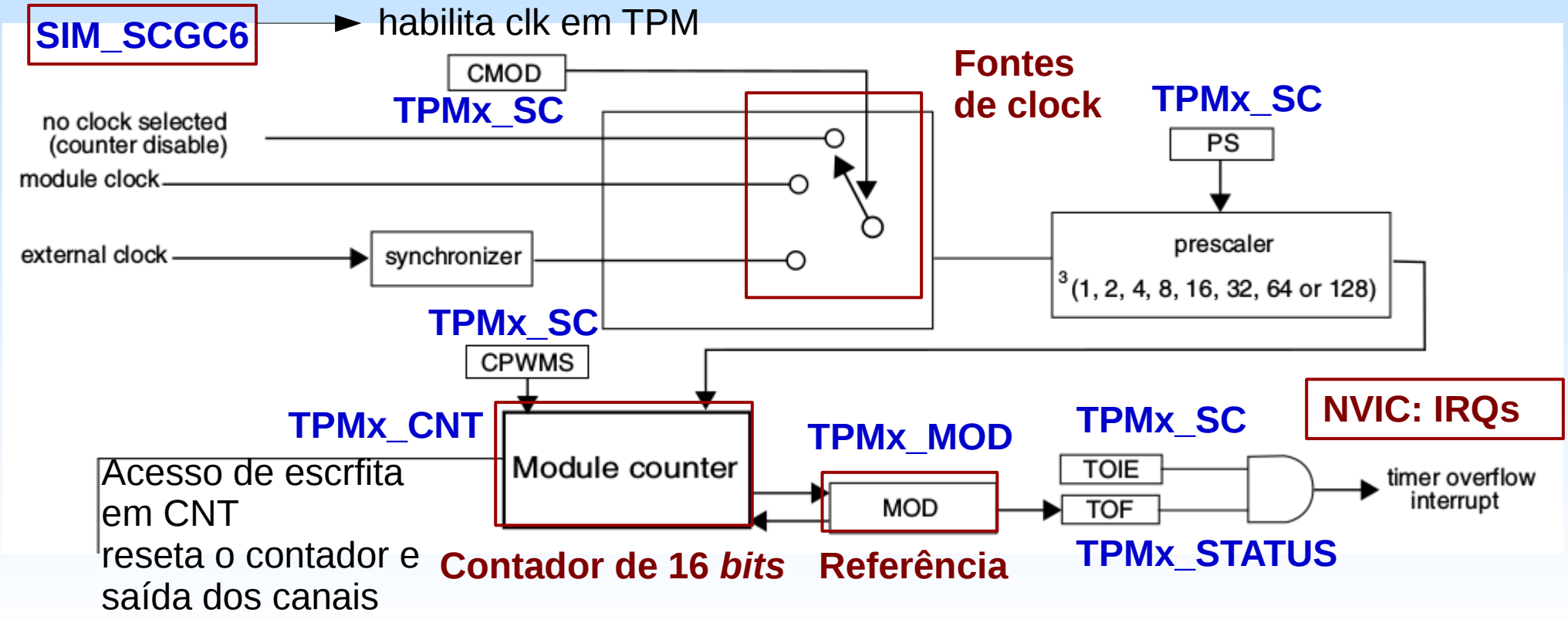
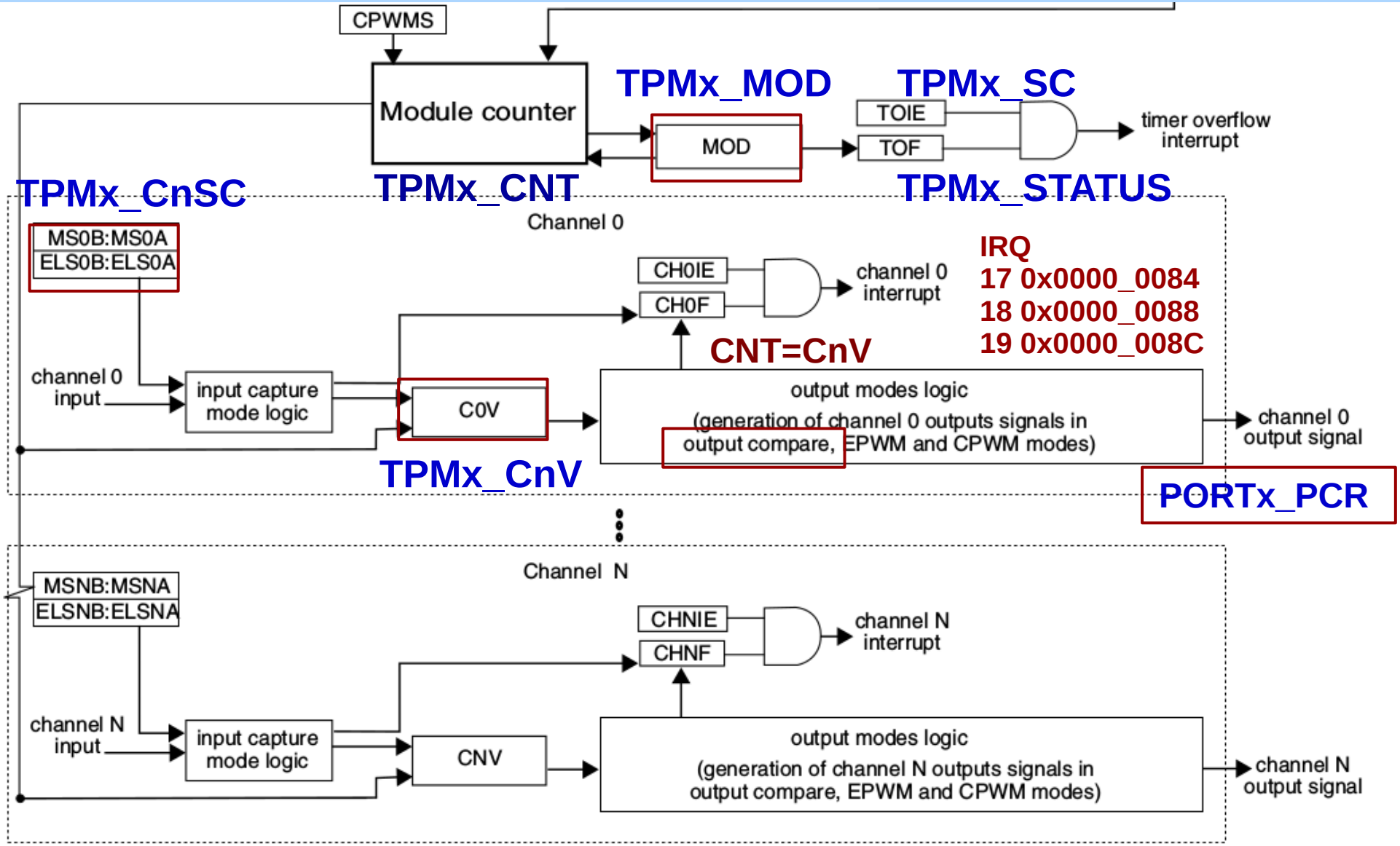


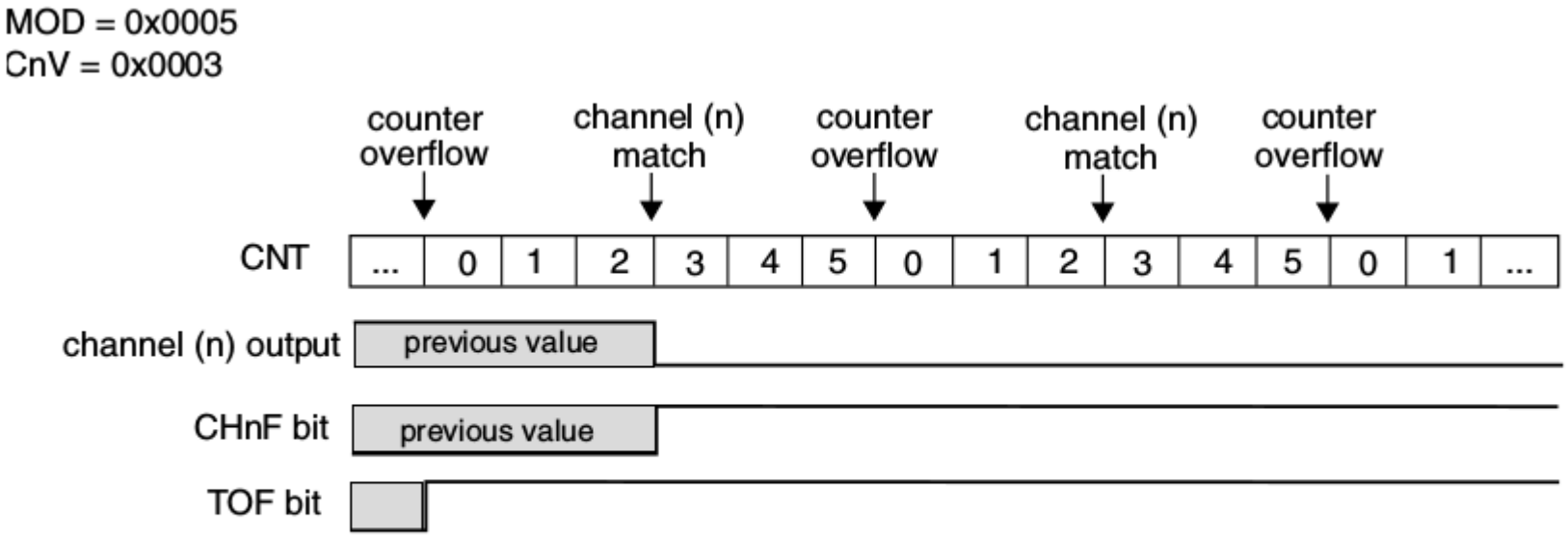
Table 31-34. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	00	00	None	Channel disabled
X	01/10/11	00	Software compare	Pin not used for LPTPM
0	00	01	Input capture	Capture on Rising Edge Only
		10		Capture on Falling Edge Only
		11		Capture on Rising or Falling Edge
	01	01	Output compare	Toggle Output on match
		10		Clear Output on match
		11		Set Output on match
	10	10	Edge-aligned PWM	High-true pulses (clear Output on match, set Output on reload)
		X1		Low-true pulses (set Output on match, clear Output on reload)
	11	10	Output compare	Pulse Output low on match
		X1		Pulse Output high on match
1	10	10	Center-aligned PWM	High-true pulses (clear Output on match-up, set Output on match-down)
		X1		Low-true pulses (set Output on match-up, clear Output on match-down)

Output Compare

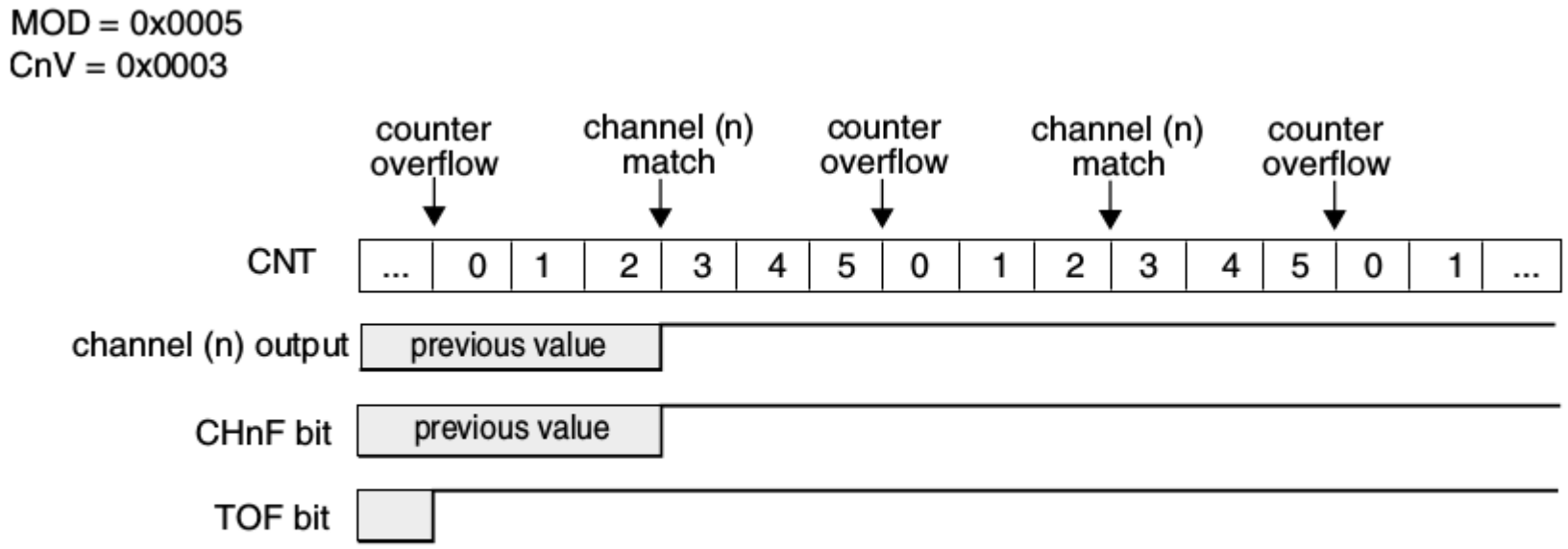


TOC: Saída 0



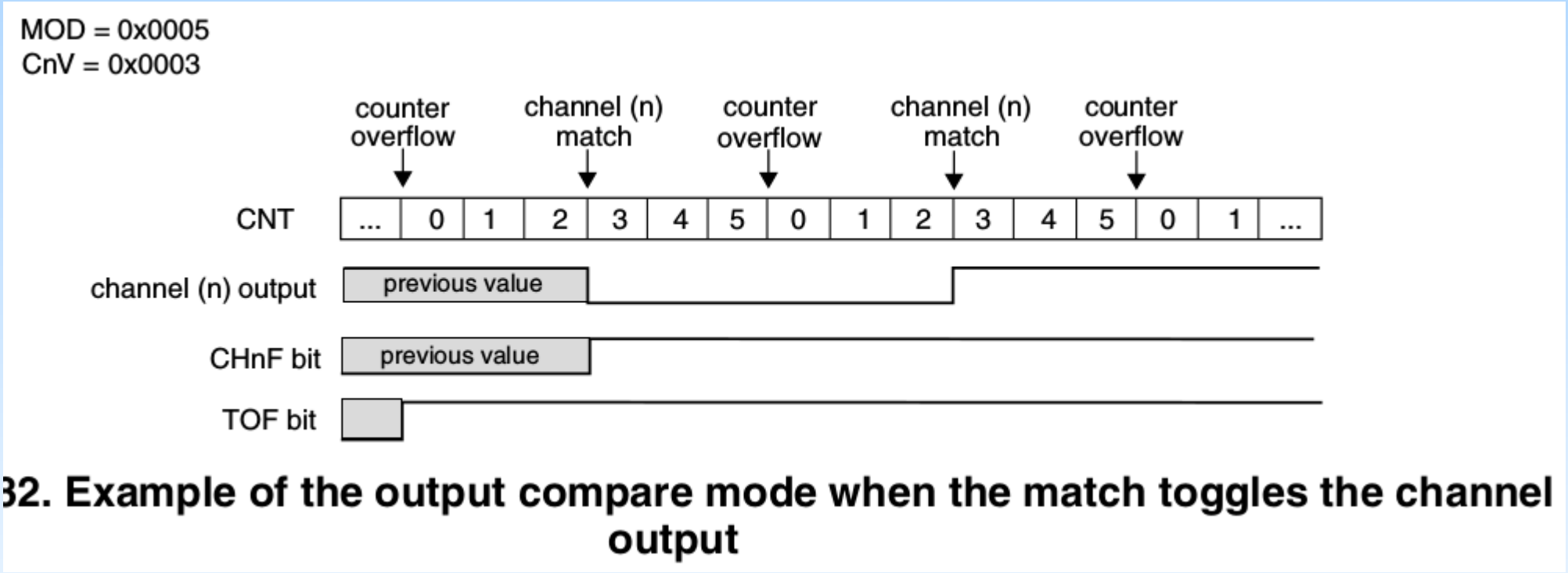
-83. Example of the output compare mode when the match clears the channel output

TOC: Saída 1



31-84. Example of the output compare mode when the match sets the channel output

TOC: Saída Alternante



32. Example of the output compare mode when the match toggles the channel output

TOC: Pino de Saída

- *Reset*: sinal de saída no nível lógico 1.
- Canal desabilitado: “*previous state*”?

Como configurar nível lógico do pino em estado “desabilitado”?

- Uma solução: multiplexar a função do pino entre GPIO e TPM, de forma que no estado “desabilitado” do TPM o estado desejado do pino seja controlado pelo GPIO.

Input Capture

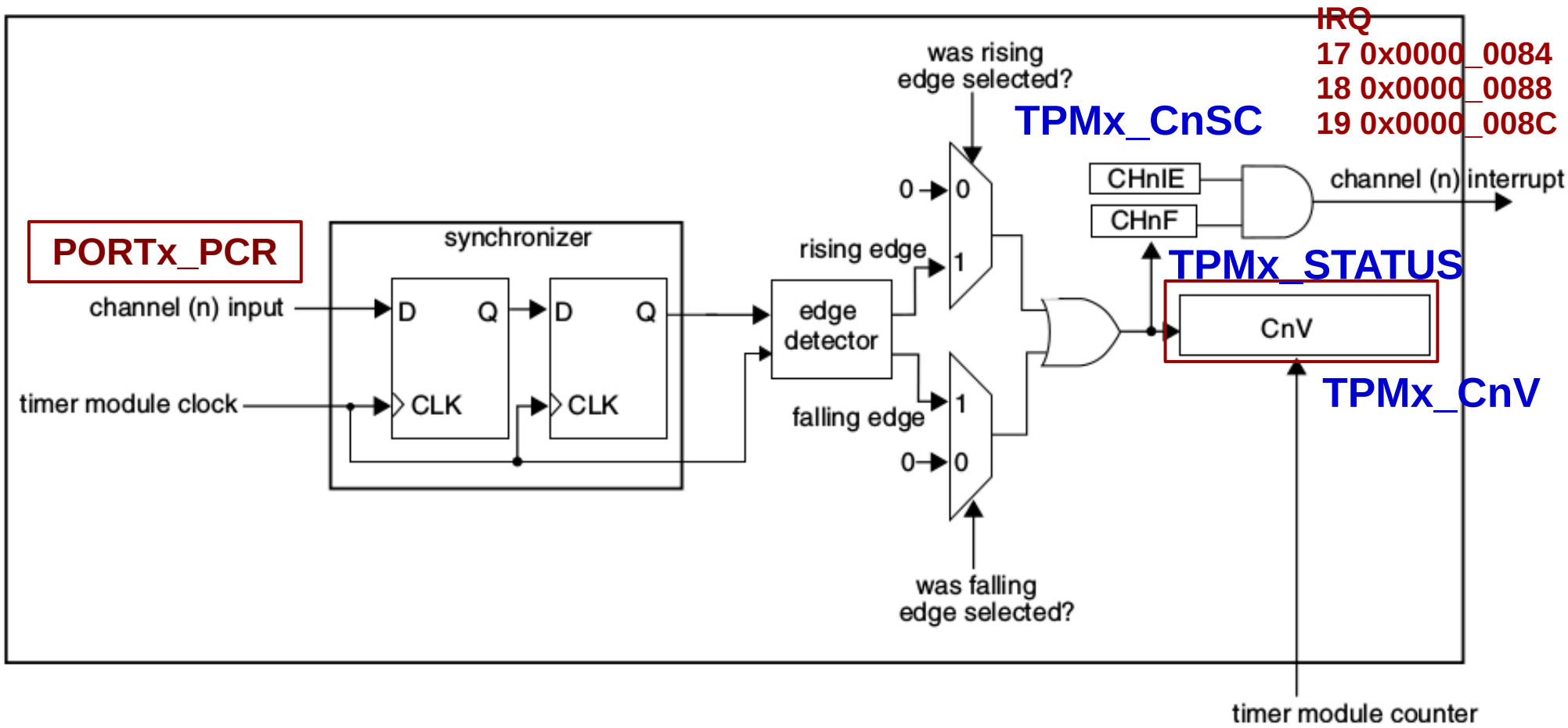
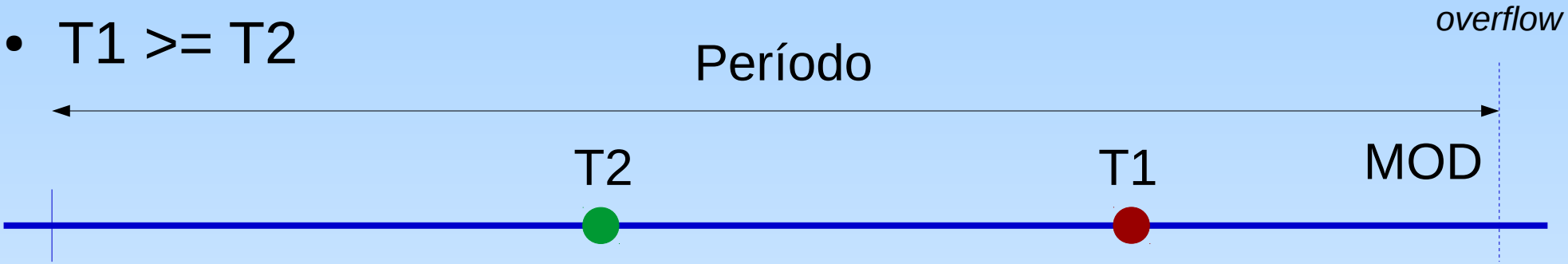


Figure 31-81. Input capture mode

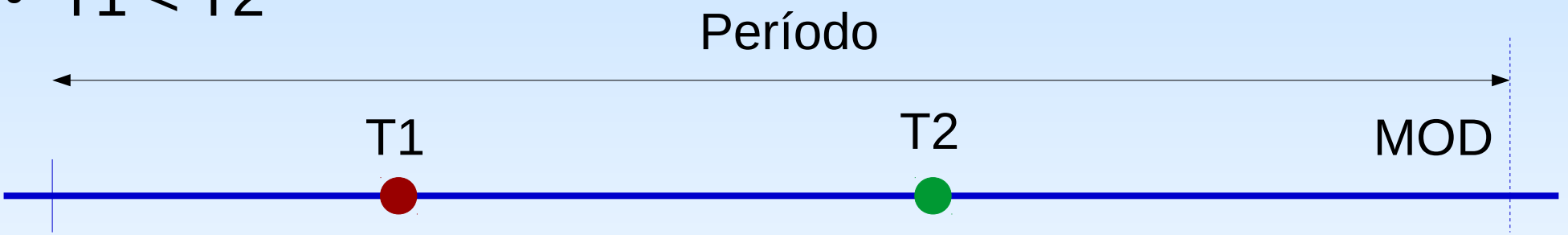
Intervalo de Tempo entre 2 Capturas

- T1 >= T2



$$t = \left(\frac{\left(MOD + T2 - T1\right)}{MOD} + \left(N - 1\right)\right) \times Período$$

- T1 < T2



$$t = \left(\frac{\left(T2 - T1\right)}{MOD} + N\right) \times Período$$

N = qtde de *overflows*

Registradores

- SIM_SOPT2: registrador de seleção de fonte de clk
- SIM_SCGC6: registrador para habilitar clk do módulo TPM
- SIM_SCGC5: registrador para habilitar porta dos pinos de saída/entrada

- PORTx_PCRn: configuração do pino n da PORTx
- GPIOx_*: registradores para configurar o estado do pino de saída enquanto TPM estiver desabilitado

- TPMx_SC: registrador de controle e de estado do módulo x
- TPMx_MOD: registrador de referência (valor máximo de contagem ou módulo de contagem)
- TPMx_CNT: contador
- TPMx_CnSC: registrador de estado e de configuração do canal n
- TPMx_CnV: registrador do valor de contagem para comparação
- TPMx_STAUTS: registrador de todos os estados
- TPMx_CONF: registrador de configuração de modo de operação do TPM em diferentes estados

Registradores

- NVIC_ISER
- NVIC_ICER
- NVIC_ISPR
- NVIC_ICPR
- NVIC_IPR3

Programação do Módulo

- **Inicialização do TPM**

```
//Configurar e habilitar fonte de clk
SIM_SOPT2 |= SIM_SOPT2_TPMSRC(0b01);
SIM_SOPT2 &= ~SIM_SOPT2_PLLFLLSEL_MASK;
SIM_SCGC6 |= SIM_SCGC6_TPM2_MASK;
//Configurar TPM
TPM2_SC &= ~(TPM_SC_DMA_MASK | TPM_SC_TOIE_MASK |
TPM_SC_CPWMS_MASK);       //modo de contagem
TPM2_MOD &= TPM_MOD_MOD(modulo);    // periodo/modulo
TPM2_CNT |= TPM_CNT_COUNT(0x0);     // resetar
TPM_SC_PS(prescaler) ) // com prescaler especificado
TPM2_SC |= (TPM_SC_CMOD(0x1) |        // habilitar modo de clocking
```

Programação de Interrupções

- **TPMx e canal Cn**
  - TPMx_CnSC |= TPM_CnSC_CHIE_MASK;
  - TPMx_SC |= TPM_SC_TOIE_MASK;
  - NVIC

    TPM0: IRQ17 (Número de exceção 33) - 0x0000_0084

    TPM1: IRQ18 (Número de exceção 34) - 0x0000_0088

    TPM2: IRQ19 (Número de exceção 35) - 0x0000_008C
  - Definir a rotina de serviço **FTMx_IRQHandler**

Programação de TOC

- **Inicialização do canal com função TOC**

```
//Inicializar pino de saida (Opcional)
SIM_SCGC5 |= SIM_SCGC5_PORTx_MASK;
 PORTx_PCRn |= (PORT_PCR_ISF_MASK |
                        PORT_PCR_MUX(ALT) );
//Inicializar os registradores de GPIOx_*
//Configurar canal para TOC
TPMx_CnSC &= ~(TPM_CnSC_MSB_MASK |  TPM_CnSC_MSA_MASK |
TPM_CnSC_ELSB_MASK | TPM_CnSC_ELSA_MASK );
TPMx_CnSC |= (TPM_CnSC_CHF_MASK);
TPMx_CnSC &= ~( TPM_CnSC_DMA_MASK);
TPMx_CnV = TPM_CnV_VAL(comparação);
TPMx_CnSC |= (TPM_CnSC_MSA_MASK | TPM_CnSC_ELSA_MASK );
```

Programação de TIC

- **Inicialização do canal com função TIC**

```
//Inicializar pino de entrada
SIM_SCGC5 |= SIM_SCGC5_PORTx_MASK;
 PORTx_PCRn |= (PORT_PCR_ISF_MASK |
                        PORT_PCR_MUX(ALT) );
//Configurar canal para TIC
TPMx_CnSC &= ~(TPM_CnSC_MSB_MASK |  TPM_CnSC_MSA_MASK |
TPM_CnSC_ELSB_MASK | TPM_CnSC_ELSA_MASK );
TPMx_CnSC |= (TPM_CnSC_CHF_MASK);
TPMx_CnSC &= ~(TPM_CnSC_CHIE_MASK | TPM_CnSC_DMA_MASK);
TPMx_CnSC |= TPM_CnSC_ELSB_MASK;     //borda de descida
TPM0_C1SC |= TPM_CnSC_CHIE_MASK;
```

Projeto-Exemplo

- Acionamento do *led* vermelho D4 do *shield* FEEC871 pela botoeira NMI, de modo que o *led* acenda 2 segundos depois do acionamento da chave e fique piscando por ~2 segundos para então voltar para o estado apagado. A resolução do contador é 2^{16}.
  - NMI: PORTA_PCR4 (PTA4/TPM0_CH1)
    IRQ 17
  - *Led* vermelho: PORTC_PCR4 (PTC4/TPM0_CH3)
  - MOD (Referência): 65536
  - Prescaler (PS): 128
  - Fonte de clk: MCGFLLCLK (20.971.520 Hz)

$$Ciclo\,completo\left(Período\right)=\frac{\left(65536\times128\right)}{20971520}=0.4\,s\Rightarrow2\,s=5\,ciclos\,completos$$

Análise da Compatibilidade Funcional

- *Led* Vermelho D4 conectado ao pino PTC4 via um *latch* 74573, junto como outros 7 *leds* D0-D3, D5-D7
  - Estado apagado D0-D3, D5-D7:
    - Manter DO-D3 e D5-D7 em 0.
  - *Led* vermelho responsivo aos sinais do microcontrolador
    - Habilitar o *latch* (LE = 1): PTC10

Pseudocódigo do fluxo de controle

TPM0_init ();

TPM0_enableNVICInterrupt(2);

TPM0_initSwitchNMI ();

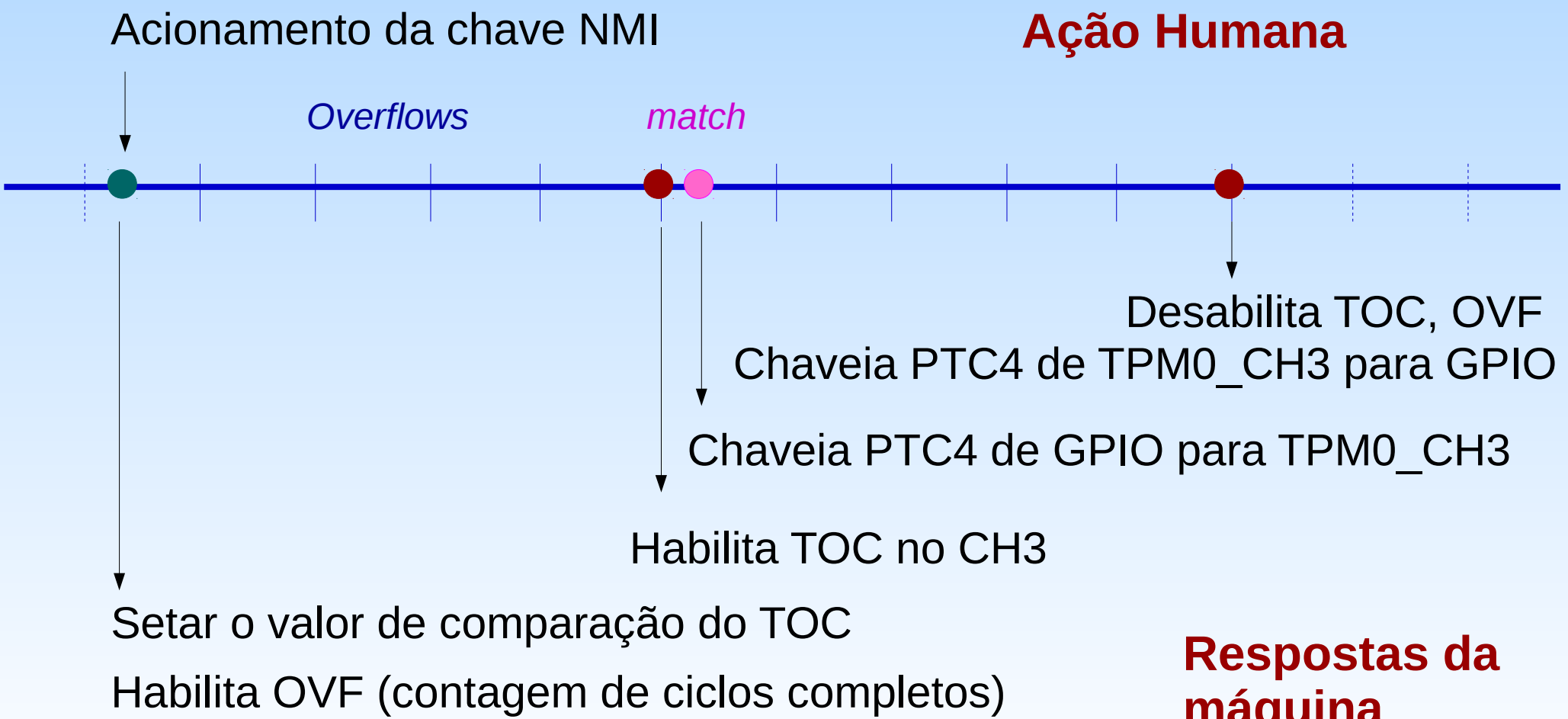
GPIO_initLatch74573();

TPM0_initLedPTC4 ();

Laço de espera:

Análise Temporal

- **Eventos referentes ao “acionamento” da chave NMI**



Pseudocódigo de rotina de serviço

FTM0_IRQHandler:

Se a chave levantou a bandeira, então

setar o valor capturado no canal do *led*;

habilita a interrupção por *overflow*;

resetar o contador de *overflows*: contador=0;

baixar a bandeira da chave;

Se o *led* levantou a bandeira, então

multiplexar o pino PTC4 para o canal TPM0_CH3;

baixar a bandeira do *led*;

Pseudocódigo de rotina de serviço (cont.)

Se o *overflow* levantou a bandeira, então

incrementar contador;

se contador == 5 então

habilitar o canal TPM0_CH3 para modo TOC;

habilitar a interrupção do canal TPM0_CH3;

se contador == 10 então

multiplexar o pino PTC4 para GPIO;

desabilitar a interrupção do canal TPM0_CH3;

desabilitar a interrupção por *overflow*;

baixar a bandeira de *overflow*.

Técnica de Programação

- Controle de quantidade de ciclos de *Overflow*
  - Uso de uma variável com qualificador *static*

    static unsigned int contador;

Técnicas de Programação

- **Modularização de códigos**

  TPM0_init ()

  TPM0_enableNVICInterrupt(2): habilitar IRQ17 no NVIC

  TPM0_initSwitchNMI (): habilitar evento de interrupção do canal 0

  GPIO_initLatch74573(): inicializar *latch* 74573

  TPM0_initLedPTC4 (): inicializar *led* Vermelho para canal 3

  TPM0_flagOnSwitchNMI(): estado de solicitação do NMI

  TPM0_flagOnLedPTC4(): estado de solicitação do *led* Vermelho

  TPM0_setCnVLedPTC4 (valor): setar o valor de comparação

  TPM0_enableOVFInterrupt(): habilitar evento de *Overflow*

Técnicas de Programação

- **Modularização de códigos**

  TPM0_w1cFlagSwitchNMI(): limpara solicitação do NMI

  TPM0_GPIO2TPMLedPTC4(): chavear de GPIO para TPM o pino PTC4

  TPM0_w1cFlagLedPTC4(): limpar a solicitação do *led* Vermelho

  TPM0_toggleLedPTC4 (): estado piscante do led Vermelho

  TPM0_enableLedPTC4Interrupt(): habilitar o evento de interrupção do led Vermelho

  TPM0_disableLedPTC4(): desabilitar o canal 3 que led Vermelho serve

  TPM0_TPM2GPIOLedPTC4 (): chavear de TPM para GPIO o pino PTC4

  TPM0_disableLedPTC4Interrupt(): desabilitar a interrupção do *led* Vermelho

  TPM0_disableOVFInterrupt(): desabilitar a interrupção por *Overflow*

  TPM0_w1cFlagOVF(): limpara a solicitação por *Overflow*

CodeWarrior IDE Development Suite

Informações Adicionais

- KL25 Sub-Family Reference Manual

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

  - Chip Configuration: Capítulo 3 (página 84)
  - Clock Distribution: Capítulo 5 (página 124)
  - PORT: Capítulo 11 (página 184)
  - TPM: Capítulo 31 (página 547)