Systemy wbudowane

dr hab. inż. Dariusz Makowski, prof. uczelni

Katedra Mikroelektroniki i Technik Informatycznych

tel. 631 2720

dmakow@dmcs.pl

http://neo.dmcs.p.lodz.pl/sw

Układy do odmierzania czasu - timery procesora

Jak mierzyć czas w systemach mikroprocesorowych ?

- Odmierzenie określonego opóźnienia ?
- Generacja daty i godziny ?
- Pomiar długości impulsów ?
- Opóźnienia w systemach czasu rzeczywistego ?

Timery

Timer – urządzenie peryferyjne procesora przeznaczone do odmierzania określonych przedziałów czasu (zliczania elementarnych cykli zegarowych). Po odmierzeniu wymaganego okresu czasu timer zwykle generuje przerwanie. Timery wykorzystywane są do odmierzania czasu systemowego, przełączania wątków, generacji opóźnień.

Przykłady układów służących do odmierzania czasu:

Timer PIT (ang. Periodic Interval Timer, Programmable Interrupt Timer),

Timer Czasu Rzeczywistego RTT (ang. Real-Time Timer),

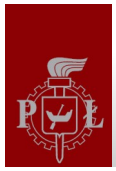
Timer PWM (ang. Pulse Width Modulation),

Timer uniwersalny TC (ang. Timer Counter),

Timer Watch-dog WDT.

Generatory sygnału zegarowego

- Kwarc z chemicznego punktu widzenia to związek zwany dwutlenkiem krzemu. Prawidłowo wycięty i zamontowany kryształ kwarcu można wprawić w drgania lub oscylacje za pomocą zmiennego prądu elektrycznego.
- Częstotliwość, z jaką kryształ oscyluje, zależy od jego kształtu i wielkości oraz od pozycji, w których umieszczone są na nim elektrody.
- Jeśli kryształ jest odpowiednio ukształtowany, będzie oscylował z pożądaną częstotliwością; w zegarach i zegarkach częstotliwość wynosi zwykle 32 768 Hz, ponieważ kryształ dla tej częstotliwości ma niewielkie rozmiary. Takie kryształy są zwykle używane w systemach cyfrowych.



Timery

Przykłady timerów mikrokontrolera STM32L4x6 MCU:

Basic Timers (TIM6, TIM7)

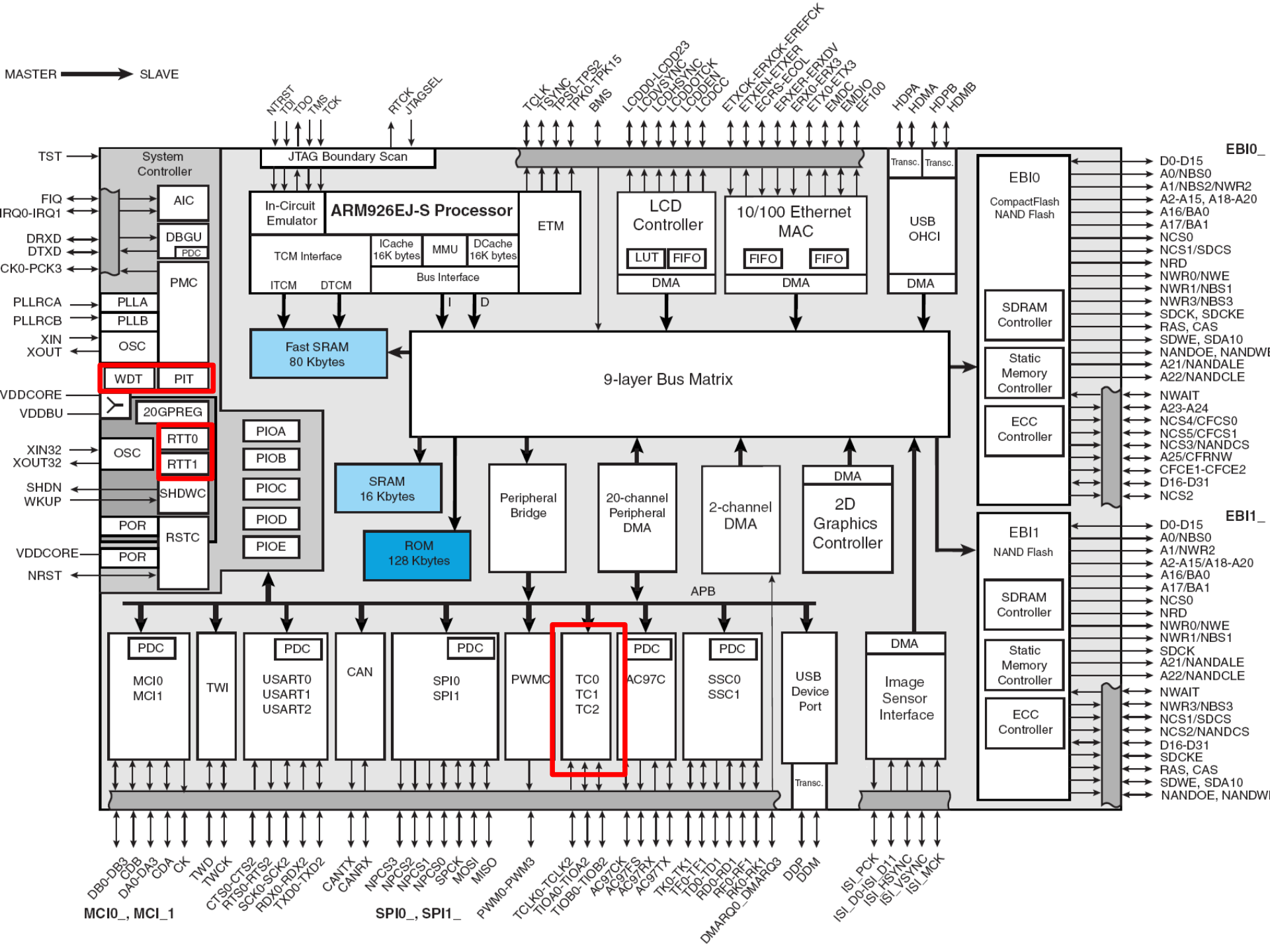
General-purpose Timers / PWM Timer (TIM2-TIM5, TIM15-TIM17)

Advanced-control Timers / PWM Timer (TIM1, TIM8)

Low-power Timer (LPTIM)

Watchdog Timers (IWDG, WWDG)

Real-time Timer (RTC)


MASTER
SLAVE
NTRST
TDI
TDO
TMS
TCK
RTCK
JTAGSEL
TCLK
TSYNC
TPS0-TPS2
TPK0-TPK15
BMS
LCDD0-LCDD23
LCDVSYNC
LCDHSYNC
LCDDOTCK
LCDDEN
LCDCC
ETXCK-ERXCK-EREFCK
ETXEN-ETXER
ECRS-ECOL
ERXER-ERXDV
ERX0-ERX3
ETX0-ETX3
EMDC
EMDIO
EF100
HDPA
HDMA
HDPB
HDMB
TST
System Controller
JTAG Boundary Scan
FIQ
IRQ0-IRQ1
DRXD
DTXD
CK0-PCK3
AIC
DBGU
PDC
PMC
PLLRCA
PLLRCB
XIN
XOUT
PLLA
PLLB
OSC
WDT
PIT
VDDCORE
VDDBU
20GPREG
RTT0
RTT1
XIN32
XOUT32
OSC
SHDN
WKUP
SHDWC
POR
RSTC
VDDCORE
POR
NRST
In-Circuit Emulator
ARM926EJ-S Processor
ETM
TCM Interface
ICache 16K bytes
MMU
DCache 16K bytes
Bus Interface
ITCM
DTCM
I
D
LCD Controller
LUT
FIFO
DMA
10/100 Ethernet MAC
FIFO
FIFO
DMA
Transc.
Transc.
USB OHCI
DMA
Fast SRAM 80 Kbytes
9-layer Bus Matrix
PIOA
PIOB
PIOC
PIOD
PIOE
SRAM 16 Kbytes
ROM 128 Kbytes
Peripheral Bridge
20-channel Peripheral DMA
2-channel DMA
DMA
2D Graphics Controller
APB
EBI0
CompactFlash NAND Flash
SDRAM Controller
Static Memory Controller
ECC Controller
EBI0_
D0-D15
A0/NBS0
A1/NBS2/NWR2
A2-A15, A18-A20
A16/BA0
A17/BA1
NCS0
NCS1/SDCS
NRD
NWR0/NWE
NWR1/NBS1
NWR3/NBS3
SDCK, SDCKE
RAS, CAS
SDWE, SDA10
NANDOE, NANDWE
A21/NANDALE
A22/NANDCLE
NWAIT
A23-A24
NCS4/CFCS0
NCS5/CFCS1
NCS3/NANDCS
A25/CFRNW
CFCE1-CFCE2
D16-D31
NCS2
EBI1
NAND Flash
SDRAM Controller
Static Memory Controller
ECC Controller
EBI1_
D0-D15
A0/NBS0
A1/NWR2
A2-A15/A18-A20
A16/BA0
A17/BA1
NCS0
NRD
NWR0/NWE
NWR1/NBS1
SDCK
A21/NANDALE
A22/NANDCLE
NWAIT
NWR3/NBS3
NCS1/SDCS
NCS2/NANDCS
D16-D31
SDCKE
RAS, CAS
SDWE, SDA10
NANDOE, NANDWE
PDC
MCI0
MCI1
TWI
PDC
USART0
USART1
USART2
CAN
PDC
SPI0
SPI1
PWMC
TC0
TC1
TC2
PDC
AC97C
PDC
SSC0
SSC1
USB Device Port
Transc.
DMA
Image Sensor Interface
DB0-DB3
CDB
DA0-DA3
CDA
CK
TWD
TWCK
CTS0-CTS2
RTS0-RTS2
SCK0-SCK2
RDX0-RDX2
TXD0-TXD2
CANTX
CANRX
NPCS3
NPCS2
NPCS1
NPCS0
SPCK
MOSI
MISO
PWM0-PWM3
TCLK0-TCLK2
TIOA0-TIOA2
TIOB0-TIOB2
AC97CK
AC97FS
AC97RX
AC97TX
TK0-TK1
TF0-TF1
TD0-TD1
RD0-RD1
RF0-RF1
RK0-RK1
DMARQ0_DMARQ3
DDP
DDM
ISI_PCK
ISI_D0-ISI_D11
ISI_HSYNC
ISI_VSYNC
ISI_MCK
MCI0_, MCI_1
SPI0_, SPI1_

Basic Timers
(TIM6, TIM7, LPTIM)
Chapter 33

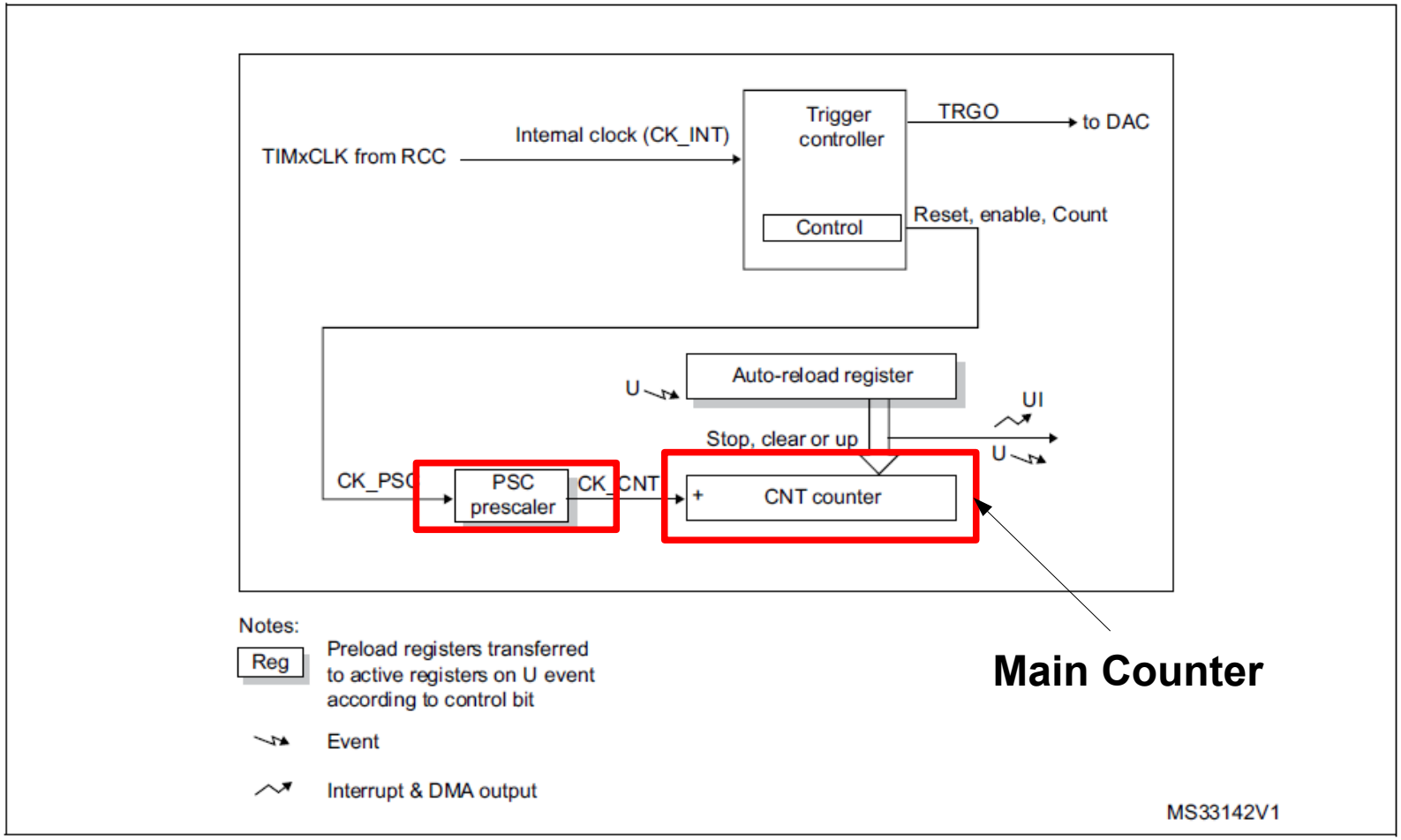
Basic timers (TIM6/TIM7)

Basic timer (TIM6/TIM7) features include:

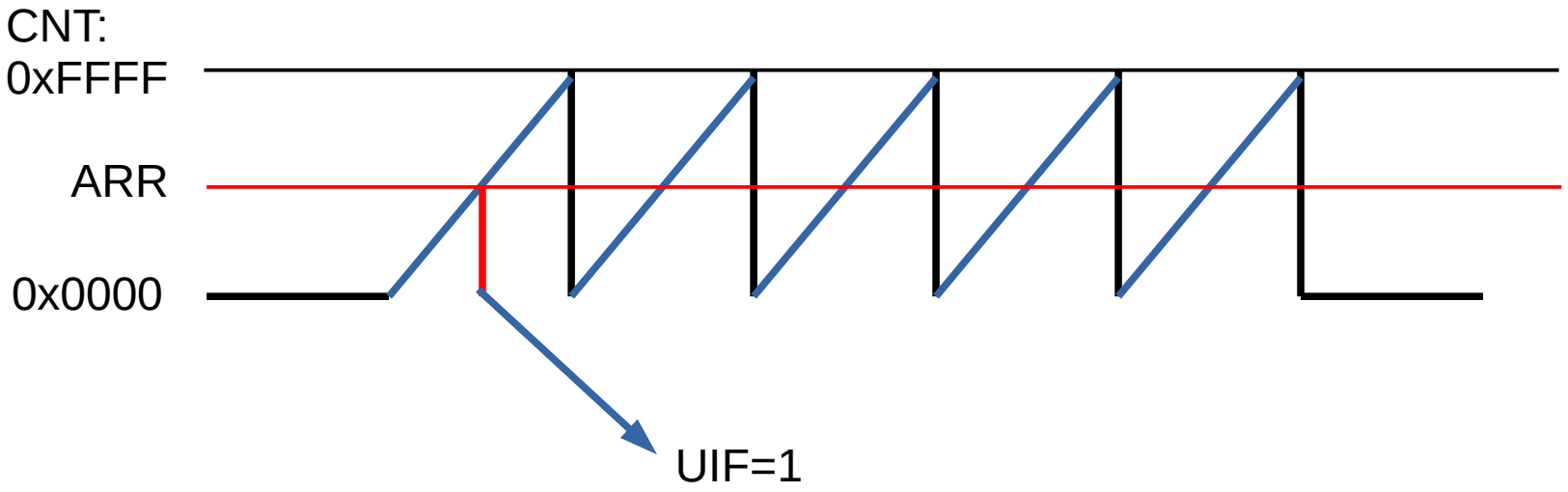
- 16-bit auto-reload up-counter
- Additional 16-bit programmable prescaler used to divide the counter clock frequency (1 to 65535)
- Synchronization circuit to trigger other peripheral devices (DAC)
- Interrupt/DMA generation on the update event: counter overflow

Block Diagram of Basic Timer

Figure 368. Basic timer block diagram



Automatic Reload of Timer



Period of generated events:

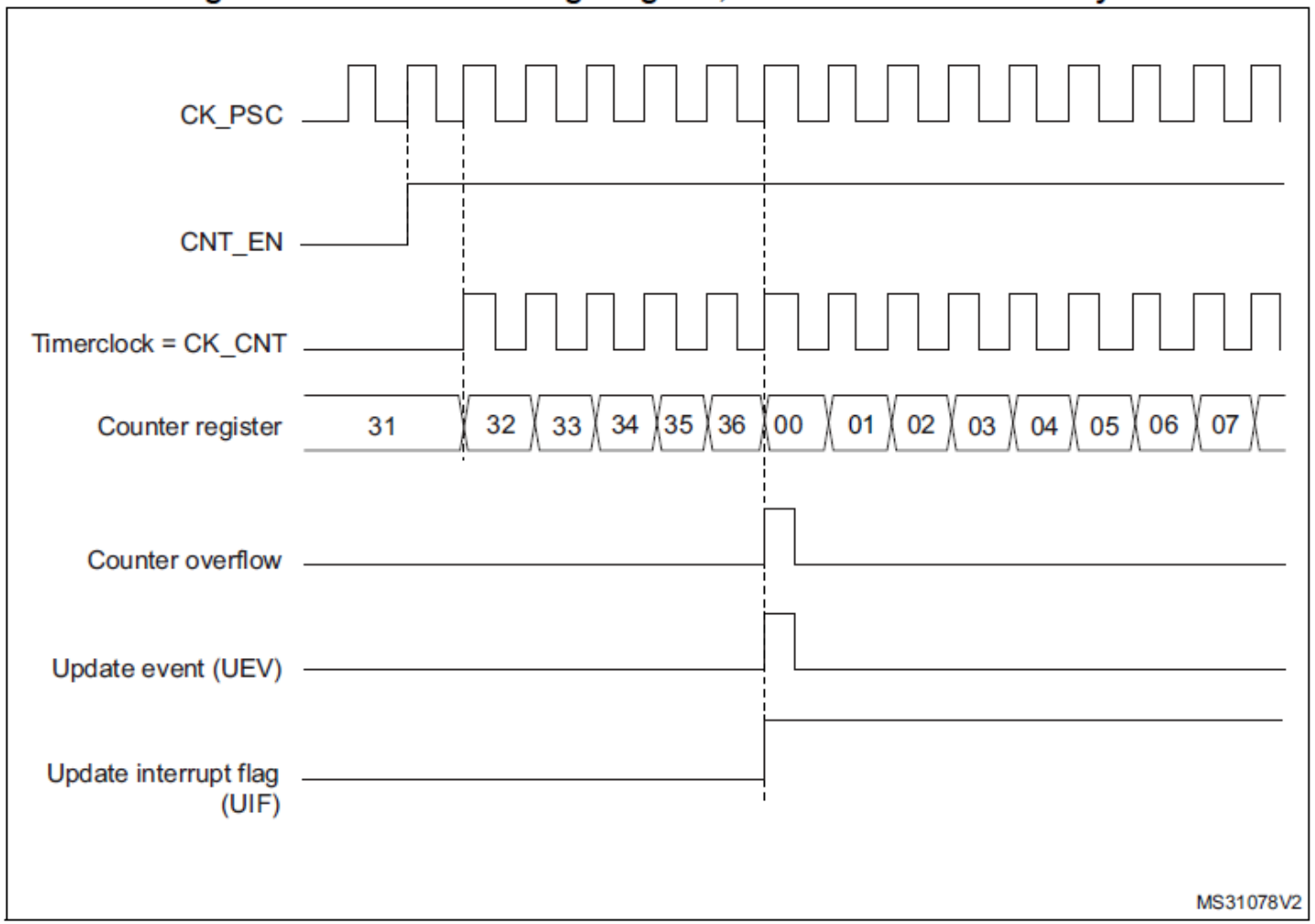
t_{TIM} = (TIM6_ARR+1) * (TIM6_PSC+1) / Clk => **ARR = ….**

Clk = 4 MHz, TIM6_ARR = 1999, TIM6_PSC = 3999 => t_{TIM} = 2000 ms

(**please provide detailed calculations during lab**)

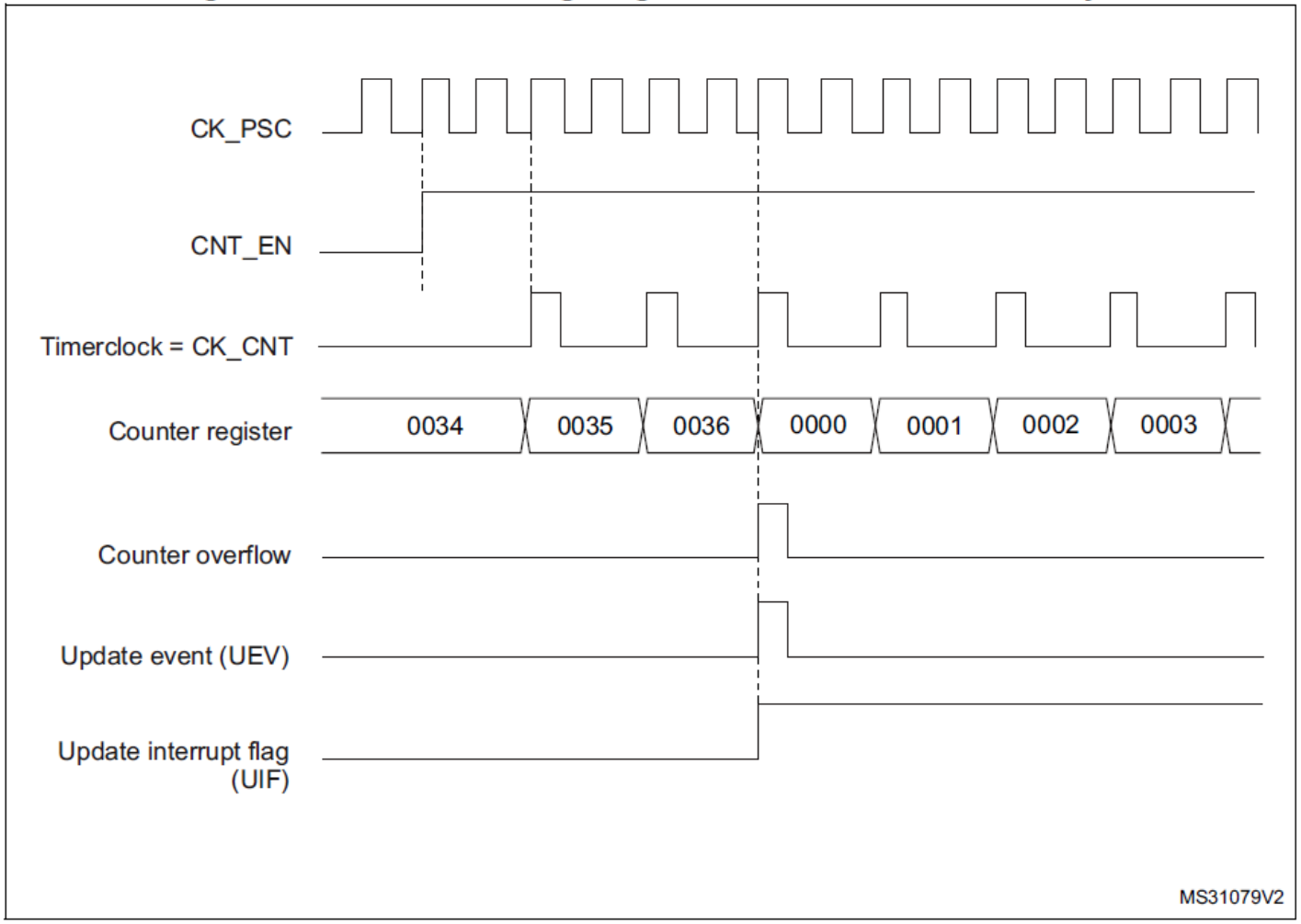
Counting Mode (Divider set to 1)

Figure 371. Counter timing diagram, internal clock divided by 1



Counting Mode (Divider set to 2)

Figure 372. Counter timing diagram, internal clock divided by 2



Basic Timer Registers

Offset	Register	Name	Access	Reset Value
0x00	Control register 1	TIMx_CR1	R/W	0x0000
0x04	Control register 2	TIMx_CR2	R/W	0x0000
0x0C	DMA/Interrupt enable register	TIMx_DIER	R/W	0x0000
0x10	Status register	TIMx_SR	R/W	0x0000
0x14	Event generation register	TIMx_EGR	W	0x0000
0x24	Counter	TIMx_CNT	RW	0x0000 0000
0x28	Prescaler	TIMx_PSC	RW	0x0000
0x2C	Auto-reload register	TIMx_ARR	RW	0xFFFF

Timer 6 – Base Address

APB1	0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG	*Section 37.4.4: WWDG register map*
	0x4000 2800 - 0x4000 2BFF	1 KB	RTC	*Section 38.6.21: RTC register map*
	0x4000 2400 - 0x4000 27FF	1 KB	LCD	*Section 25.6.6: LCD register map*
	0x4000 1800 - 0x4000 2400	3 KB	Reserved	-
	0x4000 1400 - 0x4000 17FF	1 KB	TIM7	*Section 33.4.9: TIM6/TIM7 register map*
	0x4000 1000 - 0x4000 13FF	1 KB	TIM6	*Section 33.4.9: TIM6/TIM7 register map*

Programming Timer 6 with HAL - Structure

```
typedef struct {
  TIM_TypeDef                 *Instance;        /* Reg. base address */
  TIM_Base_InitTypeDef        Init;            /* TIM Time Base par.*/
  HAL_TIM_ActiveChannel       Channel;         /* Active channel    */
  DMA_HandleTypeDef           *hdma[7];        /* DMA Handlers arr. */
  HAL_LockTypeDef             Lock;            /* Locking object    */
  __IO HAL_TIM_StateTypeDef   State;           /* TIM operat. state */
  __IO HAL_TIM_ChannelStateTypeDef  ChannelState[6];  /* TIM channel state */
  __IO HAL_TIM_ChannelStateTypeDef  ChannelNState[4]; /* TIM compl. ...    */
  __IO HAL_TIM_DMABurstStateTypeDef DMABurstState;    /* DMA burst state   */
  ...
  ...
} TIM_HandleTypeDef;
```

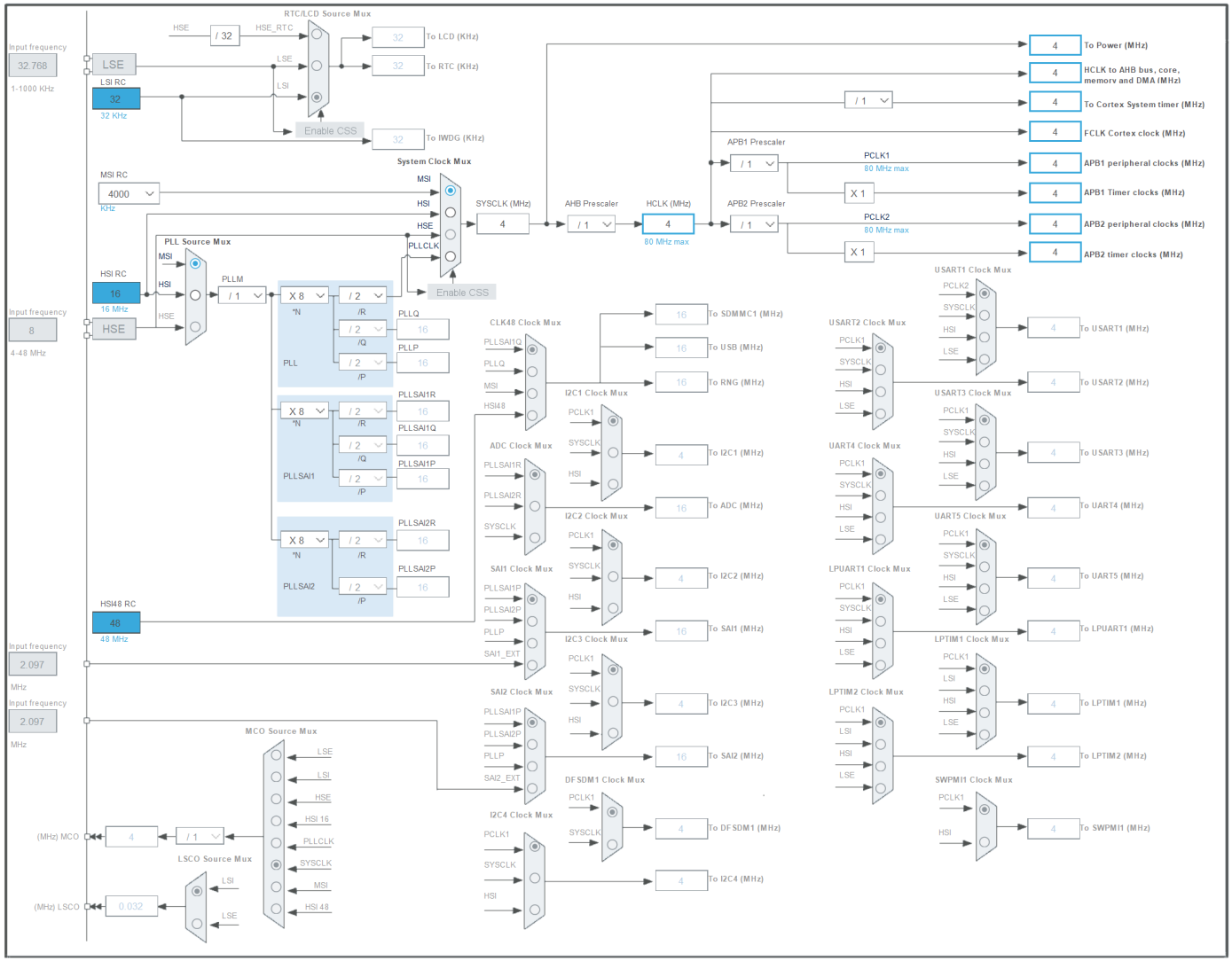
Programming Timer 6 with HAL - Initialization

```
static TIM_HandleTypeDef s_TimerInstance = {
    .Instance = TIM6
  };
  __HAL_RCC_TIM6_CLK_ENABLE();

s_TimerInstance.Init.AutoReloadPreload=TIM_AUTORELOAD_PRELOAD_ENABLE;
s_TimerInstance.Init.Period = 999;              // 2000 counts = 2000 ms
s_TimerInstance.Init.Prescaler = 3999;          // 4 MHz / (4000)  = 1 kHz

HAL_TIM_Base_Init(&s_TimerInstance);
HAL_TIM_Base_Start(&s_TimerInstance);
```

Programming Timer 6 with HAL - Usage

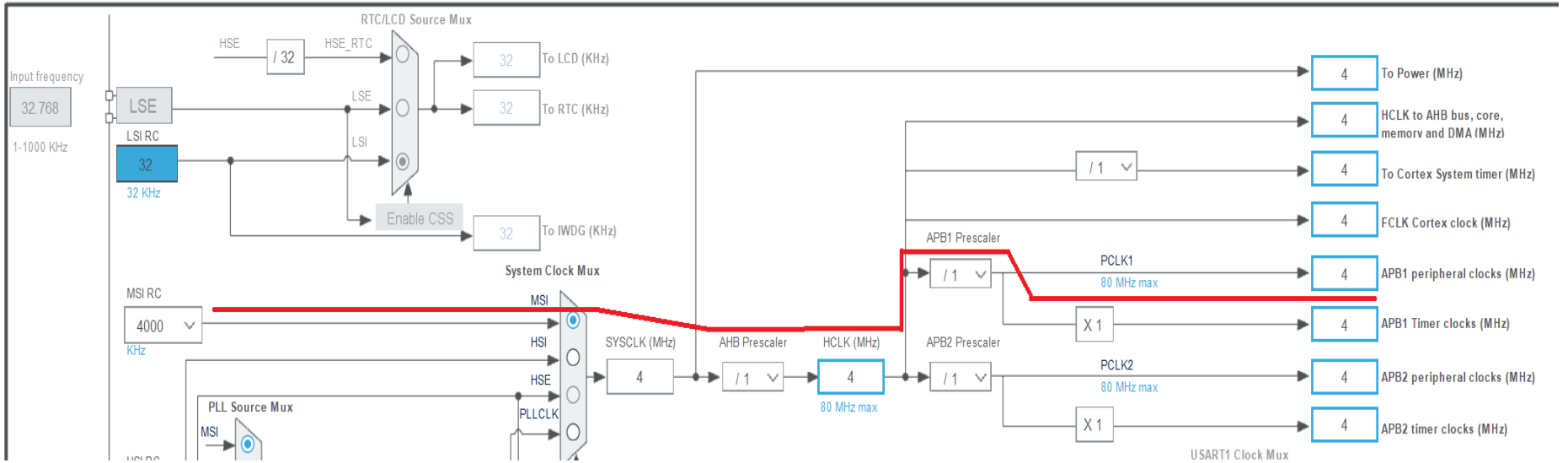
```
while (1)
 {
 uint8_t flag = __HAL_TIM_GET_FLAG(&s_TimerInstance, TIM_FLAG_UPDATE);
       if (flag)
       {
            /* USER CODE END WHILE */
            __HAL_TIM_CLEAR_FLAG(&s_TimerInstance, TIM_FLAG_UPDATE);
       }
  /* USER CODE BEGIN 3 */
 }
```

Configuration of Reference Clock



Configuration of Reference Clock (2)

- Timer 6 uses the APB1 clock
- Generated by internal MSI RC oscillator, f = 4 MHz
- APB1 clock derived directly from MSI clock (also 4 MHz)
- We do not recommend changing these settings



Timer Usage...

- Enable timer clock: RCC_APB1ENR1, bit TIM6EN
- Configure timer (2000 ms, periodic)

  1. Configure timer to auto-reload, periodic mode: TIM6_CR1, bits: ARPE, OPM, URS
  2. Enable update event: TIM6_CR1, bit: UDIS
  3. Configure prescaler: TIM6_PSC
  4. Configure period: TIM6_ARR
  5. Configure reload timer after event: TIM6_EGR, bit UG
- Enable timer: TIM6_CR1, bit: CEN
- Use polling for checking Status Flag: TIM6_SR, bit UIF
- Clear UIF (if required) after counted period → Count next period, etc.

Optional:

- Check TIM6_CNT to see if timer is counting

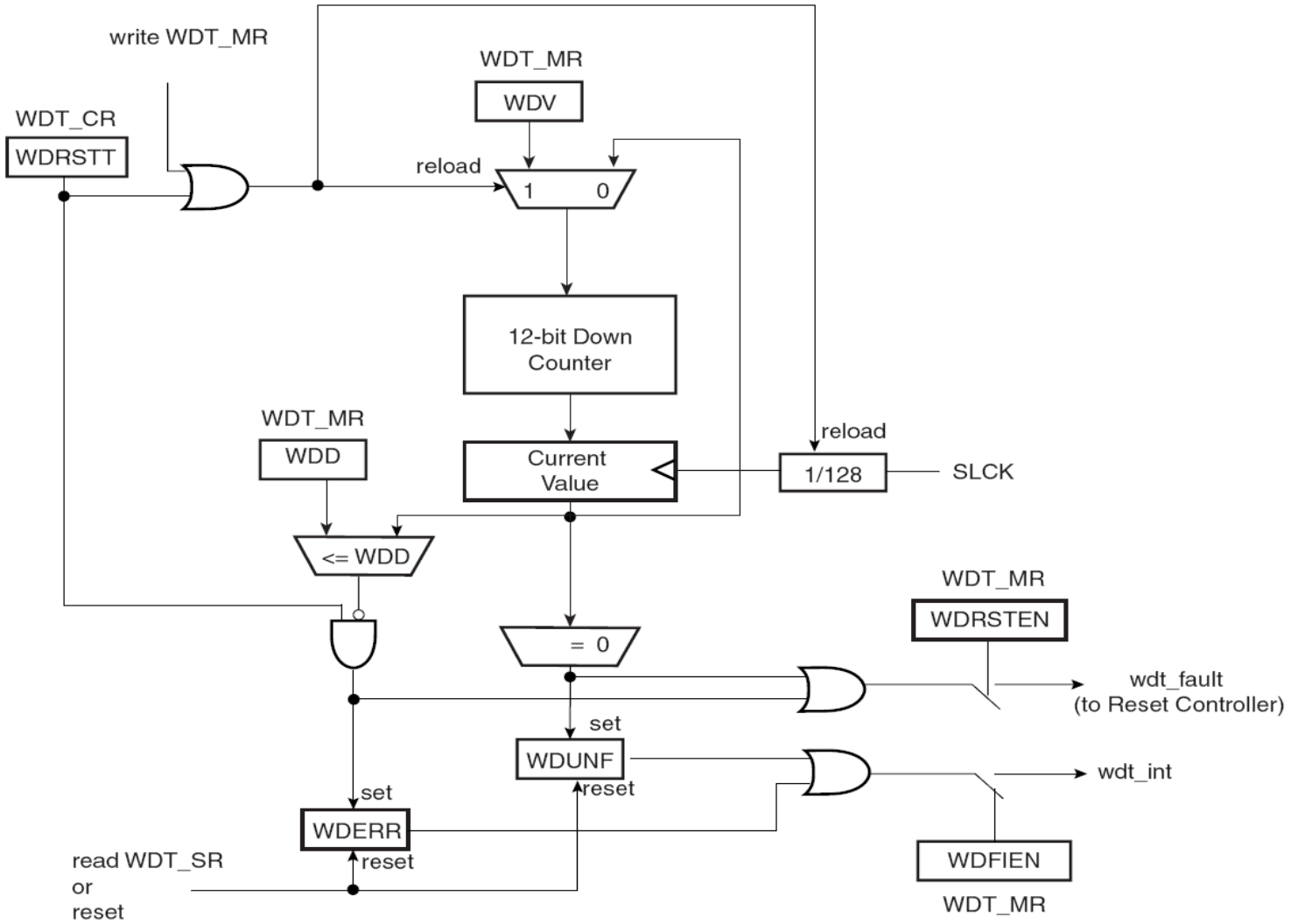
Watchdog Timer

Watchdog Timer (WDT) is used to prevent microprocessor system lock-up if the software becomes trapped in a deadlock.

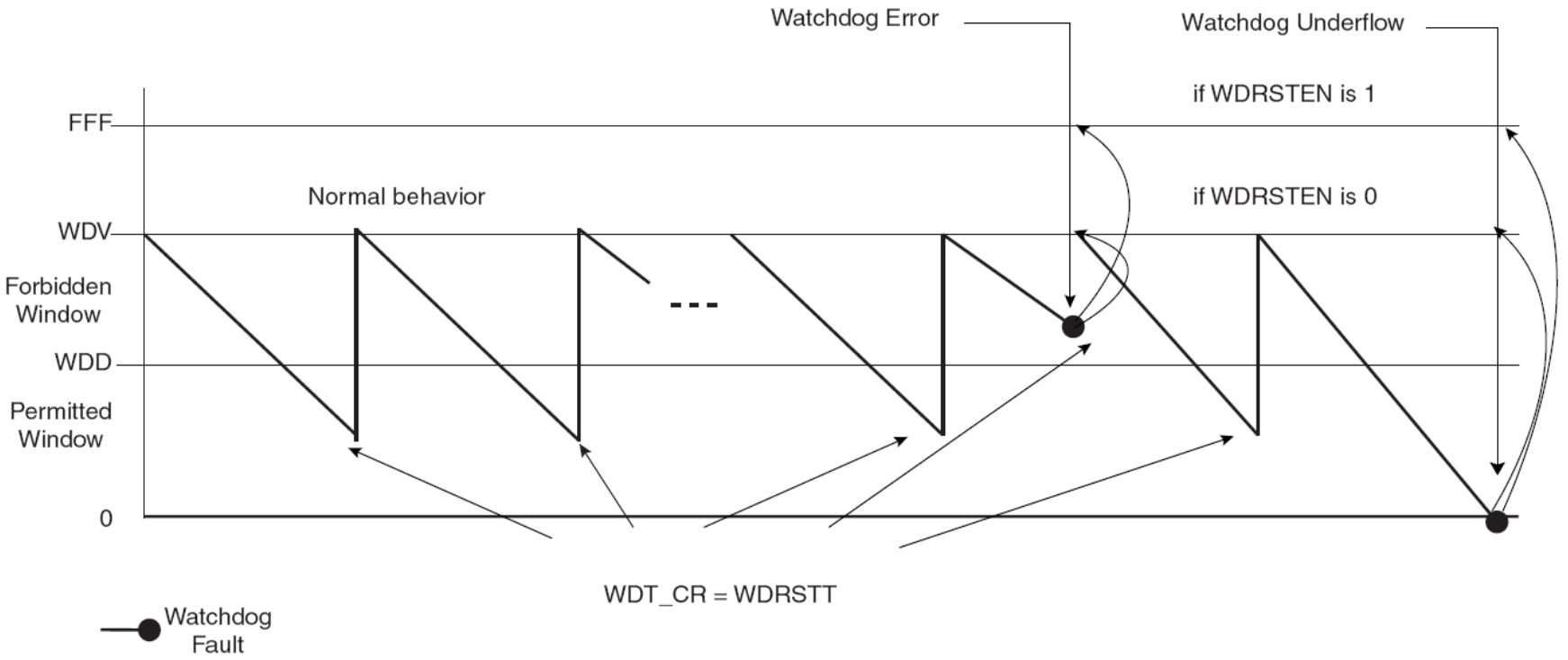
Features of WDT:

- 12-bit down counter,
- Triggered with slow clock (32.768 kHz),
- Maximum watchdog period of up to 16 seconds,
- Can generate a general reset or a processor reset only,
- WDT can be stopped while the processor is in debug mode or idle mode,
- Write protected WDT_CR (control register).

Watchdog Timer – block diagram



WDT – timing charts



WDT – registers (1)

17.4.1 Watchdog Timer Control Register

Register Name: WDT_CR

Access Type: Write-only

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

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the Watchdog.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

WDT – registers (2)

17.4.2 Watchdog Timer Mode Register

Register Name: WDT_MR

Access Type: Read/Write Once

31	30	29	28	27	26	25	24
		WDIDLEHLT	WDDBGHLT	WDD			
23	22	21	20	19	18	17	16
WDD							
15	14	13	12	11	10	9	8
WDDIS	WDRPROC	WDRSTEN	WDFIEN	WDV			
7	6	5	4	3	2	1	0
WDV							