

Index

Index	1
Output pulse with high resolution	2
User-Function-Codes of PLSOUT1.TDD	4
Documentation History	8

Output pulse with high resolution

The device driver 'PLSOUT1' is able to generate pulses with a resolution of up to 0.4 µsec. The area is determined during installation of the driver. However, the area can also be altered at a later time through commands to the driver.

File name: PLSOUT1.TDD

INSTALL DEVICE #D, "PLSOUT1.TDD", Area

D is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0...63 and stands for the device number of the driver.

Area is a parameter to determine the area.

Area	Timers	Resolution	Time area
1	2.500.000 kHz	0.400 µsec	0.0004...26.214 sec
2	625.000 kHz	1.600 µsec	0.0016...104.856 sec
3	156.250 kHz	6.400 µsec	0.0064...419.424 sec

Secondary address 0 selects the channel 0 of the pulse-out-device driver. The possible number of channels depends on the module In BASIC-Tiger® or Tiny-Tiger® modules version 1.0xx only this channel is available. The input pin is always **Pin L86**.

The device driver PLSOUT1 enables a very fast pulse output up to 1.25 MHz hereby uses hardware resources of the BASIC-Tiger® or Tiny-Tiger® module. Since other fast drivers may also need these hardware resources, the simultaneous use of a number of drivers is excluded.

Possible uses of the driver PLSOUT1.TDD together with PLSIN1.TDD in BASIC-Tiger® and Tiny-Tiger® modules

PLSOUT1	PLSIN1
Module Version 1.0xx	
1 channel	—
—	1 channel

Pulse output:

PUT #D, #0, *cnt*, *duty*, *cycle*

D	is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0...63 and stands for the device number of the driver.
cnt	is a constant, variable or expression of the data type LONG and specifies the number of pulses to be output.
duty	is a constant, variable or expression of the data type WORD in the range from 0...65535 and specifies the time in units of the set area for which the pulse should be 'low'.
cycle	is a constant, variable or expression of the data type WORD in the range from 0...65535 and specifies the total time of a pulse in units of the set area.

Attention: this device driver needs variables of the above given type: LONG for cnt, and WORD for cycle and duty.

When counting pulses then the current Tiger modules are restricted to the following values of CYCLE and DUTY:

CYCLE, range-1 min. 32, range-2 min. 8, range-3 min. 3

DUTY, range-1 min. 31, range-2 min. 7, range-3 min. 2

Smaller values with COUNT \neq 0 will cause a runtime error.

User-Function-Codes of PLSOUT1.TDD

User-Function-Codes for input (instruction GET):

No	Symbol Prefix: UFCI_	Description
176	UFCI_OPL_STAT	read current Count-Down value 0: last pulse just finishing n: n complete pulses follow -1: already at a standstill 7FFFFFFF: endless output

User-Function-Codes for the device drivers PLSOUT1 for output are defined in the Include-File 'UFUNCn.INC' (instruction PUT):

No	Symbol	Description
144	UFCO_OPL_RNG	set area
145	UFCO_OPL_CNT_ADD	add new number of pulses
146	UFCO_OPL_CNT_SET	set new number of pulses n: generates n pulses 0: soft stop -1: hard stop
147	UFCO_OPL_SET_LEV	Set level of L86, only if pulses are stopped 0: L86 Low ↔0: L86 High

Example: (area: 3) output 10 pulses with the cycle time 32µsec (5*6.4µsec) and the Low-Time of 12.8µsec (2* 6.4µsec):

```
PUT #10, #0, 10, 2, 5
```

Example: Set area 2 during the runtime:

```
PUT #10,#0, #UFCO_OPL_RNG, 2
```

PLSOUT1.TDD

Example: read the number of pulses following the pulse which is currently running:

```
GET #10,#0, #UFCI_OPL_STAT, 4, CNT
```

Example: with running, possibly infinite pulse output, set the new number of pulses to 1. The output can thus be aborted, whereby the scanning rate and the frequency are retained, the last pulse is still completely output:

```
PUT #10,#0, #UFCO_OPL_CNT_SET, 1
```

Example: with running, possibly infinite pulse output, set the new number of pulses to 0. This is a soft stop. The current pulse will be finished and then the output is stopped:

```
PUT #10,#0, #UFCO_OPL_CNT_SET, 0
```

Example: with running, possibly infinite pulse output, set the new number of pulses to -1. This is a hard stop. The pulse output is directly stopped and the current pulse will not be finished:

```
PUT #10,#0, #UFCO_OPL_CNT_SET, -1
```

Example: set L86 to high. Ensure that all pulses are stopped.

```
PUT #10, #0, #UFCO_OPL_SET_LEV, 1      ' L86 <- High
```

Program example endless:

```
#INCLUDE UFUNC3.INC
#INCLUDE DEFINE_A.INC
LONG no_of_pulses
WORD cycle, duty

TASK MAIN
    install_device #OPL1, "PLSOUT1.TDD", 1      ' Range-1

    no_of_pulses = 0                            ' endless
    cycle = 3                                   ' 1,2 ys
    duty = 1                                    ' 400 ns
    put #OPL1, no_of_pulses, duty, cycle        ' start

END
```

Program example soft stop:

```

user_var_strict
#include UFUNC3.INC ' User Function Codes
#include DEFINE_A.INC ' Symbol-Definitions

#define UFCO_OPL_CNT_SET 146
#define UFCO_OPL_CNT_ADD UFCO_OPL_CNT

LONG no_of_pulses ' number of pulses
WORD cycle, duty ' PLS01-Parameter

TASK MAIN
    BYTE ever ' endlessloop

    install_device #LCD, "LCD1.TD2" ' LCD driver
    install_device #OPL1, "PLSOUT1.TD2", 3 ' Range 3

    no_of_pulses = 0 ' endless pulses
    cycle = 200

    ' start, min, max, step, ID
    duty = modulo_updo ( 10, 10, 199, 1, 0 )

    run_task do_pol '

    for ever = 0 to 0 step 0 ' endless loop
        duty = modulo_updo ( duty, 0 ) ' change duty
        print #1, "<1BH>A<0><0><F0H>duty:";duty;" ' show
        wait_duration 10 ' wait a bit
    next

END

TASK do_pol
    BYTE ever ' endlessloop
    WORD old_duty, old_cycle ' PLS01-Parameter
    LONG rest

    old_duty = duty
    old_cycle = cycle
    put #OPL1, no_of_pulses, duty, cycle ' start puls output

    for ever = 0 to 0 step 0 ' endless loop
        if (duty <> old_duty) or (cycle <> old_cycle) then '
            old_duty = duty ' if changed
            old_cycle = cycle ' store current values
            rest = 1 ' init
            wait_duration 100 ' pulse 100ms
            put #OPL1, #0, #UFCO_OPL_CNT_SET, 0 ' soft stop
            while rest > 0 ' wait for end of pules
                get #OPL1, #0, #UFCI_OPL_STAT, 4, rest ' rest pulses
            endwhile
            wait_duration 100 ' 100ms pause for scope
            put #OPL1, no_of_pulses, duty, cycle ' set new values
        endif
    next

END

```

Documentation History

Version of Documentation	Version of driver	Description / Changes
001	1.0m	-