

PWMIN (Read duty of PWM signal)

This driver measures the duty cycle of an incoming signal. PWMIN can operate up to 8 different channels (Pins). This driver can only be included once, because it uses much CPU capacity. PWMIN can NOT be installed together with PLSIN or PLSOUT, because they use the same resources.

Installation of the driver

INSTALL DEVICE #D, "PWMIN_Px.TDD" [, P1, ..., P6]

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

x is a constant, variable or an expression of data type BYTE, WORD, LONG in range of 0...63 and is the port number used for PWM input lines.

P1...P6 are more parameters, which changes the settings of the sample process in this driver.

	Default	Description of parameters
P1	-	Bit mask pin(s)
P2	1	Input clock 1: 0,4 microseconds. (DEFAULT) 2: 1,6 microseconds. 3: 6,4 microseconds.
P3a	0	Maximum frequency variation (LOW BYTE) (f2)
P3b		Maximum frequency variation (HIGH BYTE) (f2)
P4a	0	Frequency variation step width (LOW BYTE) (s)
P4b		Frequency variation step width (HIGH BYTE) (s)
P5a	1	Time in ms until frequency changes (LOW BYTE) (t)
P5b		Time in ms until frequency changes (HIGH BYTE) (t)
P6	0	Method

PWMIN (Read duty of PWM signal)

Secondary addresses

Read out the result is possible from different secondary addresses:

Secondary address	Function	Instruction
0	Read out pin 0 (from Port)	GET
1	Read out pin 1 (from Port)	GET
2	Read out pin 2 (from Port)	GET
3	Read out pin 3 (from Port)	GET
4	Read out pin 4 (from Port)	GET
5	Read out pin 5 (from Port)	GET
6	Read out pin 6 (from Port)	GET
7	Read out pin 7 (from Port)	GET
8	Read out ALL pins (from port) at the same time	GET

Example to read out pin 0:

```
GET #PWMIN, #0, 0, result$
```

Example to read out ALL channels:

```
GET #PWMIN, #8, 0, result$
```

Data structure of GET

If you GET only one channel, the result is of 4 bytes. The first 2 bytes (the lower word) is the number of sampled highs, and the next 2 bytes (the higher word) is the number of sampled lows.

If you GET all channels at once, the result is of number of channels * 4 bytes length. The active channels will append together to one string. The lower channels will be the first in the string.

User Function Codes

User-Function-Codes of PWMIN_Px.TDD to read out parameters (Instruction GET, secondary address 0):

No.	Symbol Präfix UFCI_	Description
80H	UFCI_FREQ_INIT	Initial sample time (interval) in INPUT CLOCK cycles OR 0 to stop the driver ⇒ only method 0 & 1 (f1)
81H	UFCI_FREQ_VAR	Maximum frequency variation (0: constant frequency) ⇒ only method 0 & 1 (f2)
82H	UFCI_FREQ_STEP	Frequency variation step width ⇒ only method 0 & 1 (s)
83H	UFCI_CHG_TIME_MS	Time in ms until frequency changes ⇒ only method 0 (t)
84H	UFCI_INPUT_CLOCK	Input clock 1: 0,4 microseconds. (DEFAULT) 2: 1,6 microseconds. 3: 6,4 microseconds. ⇒ only method 0 & 1
86H	UFCI_METHODE2_BUF	Read out free space in buffer Wait until this value is 0, before GETting the result value ⇒ only Method 1
87H	UFCI_CUR_FREQ	Read out the current sampling frequency ⇒ only method 0 & 1

PWMIN (Read duty of PWM signal)

User-Function-Codes of PWMIN_Px.TDD to set parameters (Instruction PUT, secondary address 0):

No.	Symbol Präfix: UFCO_	Description
80H	UFCO_FREQ_INIT	Initial sample time (interval) in INPUT CLOCK cycles OR 0 to stop the driver => only method 0 & 1 (f1)
81H	UFCO_FREQ_VAR	Maximum frequency variation (0: constant frequency) => only method 0 & 1 (f2)
82H	UFCO_FREQ_STEP	Frequency variation step width => only method 0 & 1 (s)
83H	UFCO_CHG_TIME_MS	Time in ms until frequency changes => only method 0 (t)
84H	UFCO_INPUT_CLOCK	Input clock 1: 0,4 microseconds. (DEFAULT) 2: 1,6 microseconds. 3: 6,4 microseconds. => only method 0 & 1
85H	UFCO_MAX_INT_COUNT	Sets the maximum number of same periods in a measurement More same periods => Interference error

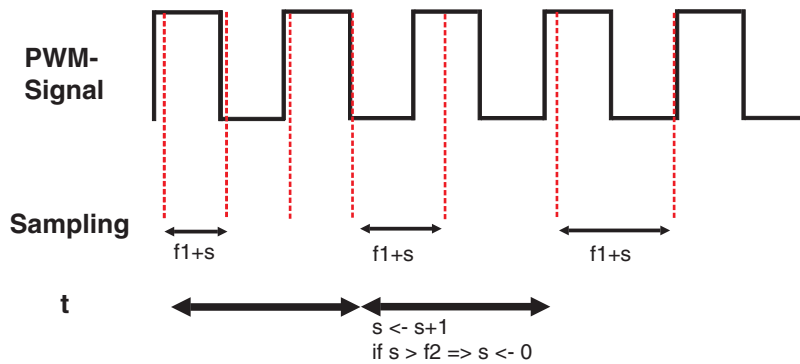
PWMIN (Read duty of PWM signal)

Method 0

The driver permanently samples the channels and writes the result into an internal buffer. The sample process does never end; oldest results are overwritten with the newest (ring buffer). The sample frequency changes after a fixed time interval or the frequency never changes.

To start the driver, send the initial frequency to sample with:

```
PUT #D, #0, #UFCO_FREQ_INIT, 250
```



To stop the driver (release CPU load):

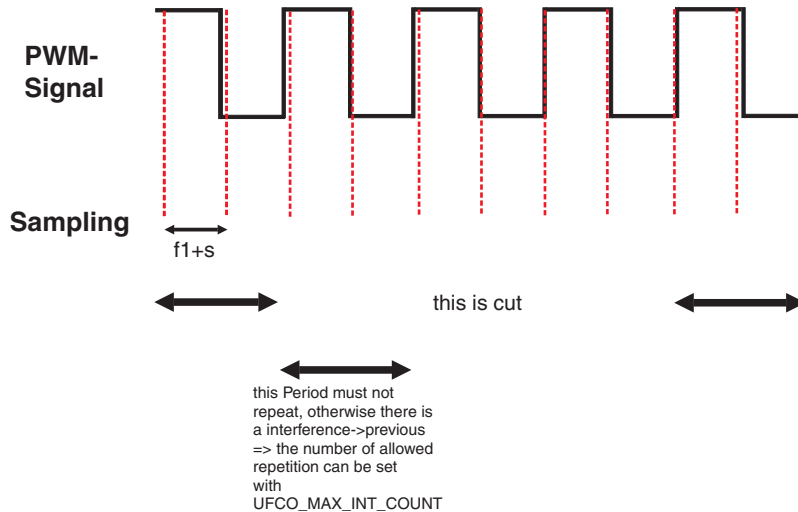
```
PUT #D, #0, # UFCO_FREQ_INIT, 0
```

Method 1

In method 2, the driver samples with a constant frequency and saves the results, until the internal buffer is full. You read out the result with GET, after that a new measurement with a probably new frequency starts. The frequency is changed depending on the parameters s & f_2 .

There is an interference check, before you get the result. If the sample frequency is the same as the PWM frequency, there might be an interference (error). This can be detected, if the first complete period is identically repeated several times. If this is the case, the measurement will be discarded. The number of allowed same periods at the beginning of the measurement is set with `UFCO_MAX_INT_COUNT`.

The result consists of only complete periods. The first and the last incomplete period will be cut from the result.



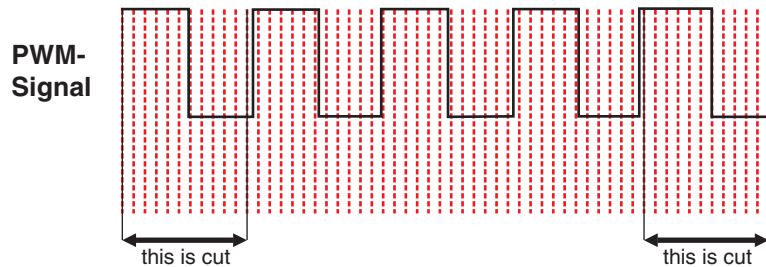
PWMIN (Read duty of PWM signal)

Method 2

This is a high speed sampling process. Parameters f1, f2, s, t are dummies here. Measurements are not started with a PUT instruction. After installing the driver, a result is sampled immediately when a GET instruction requires it.

The Tiger samples at FULL POWER for about 6 ms, generating 4096 values in high speed mode.

IMPORTANT: The rest of the Tiger system is busy at this time. No other device driver will work!!! The 1ms-ticks could have discrepancies!!!



PWMIN – XPORT (Read PWM signal duty at external pin)

This driver measures the duty cycle of an incoming signal at any XPORT Pin(s). PWMIN can operate many different channels (Pins). This driver can only be included once, because it uses much CPU capacity. PWMIN can NOT be installed together with PLSIN or PLSOUT, because they use the same resources.

PWMIN_XP samples with method 2 only!!!

Installation of the driver

INSTALL DEVICE #D, "PWMIN_XP.TDD" [, P1, ..., P7]

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

P1...P7 are more parameters, which changes the settings of the sample process in this driver.

	Default	Description of parameters
P1	-	Bit mask pin(s)
P2	1	Input clock 1: 0,4 microseconds. (DEFAULT) 2: 1,6 microseconds. 3: 6,4 microseconds.
P3a	0	Maximum frequency variation (LOW BYTE) (f2)
P3b		Maximum frequency variation (LOW BYTE) (f2)
P4a	0	Frequency variation step width (LOW BYTE) (s)
P4b		Frequency variation step width (HIGH BYTE) (s)
P5a	1	Time in ms until frequency changes (LOW BYTE) (t)
P5b		Time in ms until frequency changes (HIGH BYTE) (t)
P6	0	Method (ONLY method 2)
P7	OFFH	XPORT Address

PWMIN (Read PWM signal duty at external pin)

Secondary addresses

Read out the result is possible from different secondary addresses:

Secondary address	Function	Instruction
0	Read out pin 0 (from Port)	GET
1	Read out pin 1 (from Port)	GET
2	Read out pin 2 (from Port)	GET
3	Read out pin 3 (from Port)	GET
4	Read out pin 4 (from Port)	GET
5	Read out pin 5 (from Port)	GET
6	Read out pin 6 (from Port)	GET
7	Read out pin 7 (from Port)	GET
8	Read out ALL pins (from port) at the same time	GET

Example to read out pin 0:

```
GET #PWMIN, #0, 0, result$
```

Example to read out ALL channels:

```
GET #PWMIN, #8, 0, result$
```

Data structure of GET

If you GET only one channel, the result is of 4 bytes. The first 2 bytes (the lower word) is the number of sampled highs, and the next 2 bytes (the higher word) is the number of sampled lows.

If you GET all channels at once, the result is of number of channels * 4 bytes length. The active channels will append together to one string. The lower channels will be the first in the string.

User Function Codes

User-Function-Codes of PWMIN_XP.TDD to read out parameters (Instruction GET, secondary address 0):

No.	Symbol Präfix UFCI_	Description
80H	UFCI_FREQ_INIT	Initial sample time (interval) in INPUT CLOCK cycles OR 0 to stop the driver ⇒ only method 0 & 1 (f1)
81H	UFCI_FREQ_VAR	Maximum frequency variation (0: constant frequency) ⇒ only method 1 & 1 (f2)
82H	UFCI_FREQ_STEP	Frequency variation step width ⇒ only method 0 & 1 (s)
83H	UFCI_CHG_TIME_MS	Time in ms until frequency changes ⇒ only method 0 (t)
84H	UFCI_INPUT_CLOCK	Input clock 1: 0,4 microseconds. (DEFAULT) 2: 1,6 microseconds. 3: 6,4 microseconds. ⇒ only method 0 & 1
86H	UFCI_METHODE2_BUF	Read out free space in buffer Wait until this value is 0, before GETting the result value ⇒ only Method 1
87H	UFCI_CUR_FREQ	Read out the current sampling frequency ⇒ only method 0 & 1
88H	UFCI_XPORT_ADR	Reads out current XPORT address ⇒ only method 2

PWMIN (Read PWM signal duty at external pin)

User-Function-Codes of PWMIN_XP.TDD to set parameters (Instruction PUT, secondary address 0):

No.	Symbol Präfix: UFCO_	Description
80H	UFCO_FREQ_INIT	Initial sample time (interval) in INPUT CLOCK cycles OR 0 to stop the driver => only method 0 & 1 (f1)
81H	UFCO_FREQ_VAR	Maximum frequency variation (0: constant frequency) => only method 0 & 1 (f2)
82H	UFCO_FREQ_STEP	Frequency variation step width => only method 0 & 1 (s)
83H	UFCO_CHG_TIME_MS	Time in ms until frequency changes => only method 0 (t)
84H	UFCO_INPUT_CLOCK	Input clock 1: 0,4 microseconds. (DEFAULT) 2: 1,6 microseconds. 3: 6,4 microseconds. => only method 0 & 1
85H	UFCO_MAX_INT_COUNT	Sets the maximum number of same periods in a measurement More same periods => Interference error
88H	UFCO_XPORT_ADR	Sets the XPORT address to read from => only method 2

Data & control pins used for XBus (default):

Pins	Description
Port 6	Data bus
L33	ACLK (Address Clock)
L34	DCLK (Data Clock)
L35	INE (Input Enable)

PWMIN (Read PWM signal duty at external pin)

Method 2

This is a high speed sampling process. Parameters f1, f2, s, t are dummies here. Measurements are not started with a PUT instruction. After installing the driver, a result is sampled immediately when a GET instruction requires it.

The Tiger samples at FULL POWER for about 6 ms, generating 4096 values in high speed mode.

IMPORTANT: The rest of the Tiger system is busy at this time. No other device driver will work!!! The 1ms-ticks could have discrepancies!!!

