The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC851-SW-25

Integrity Device Driver

Multifunction I/O

16 bit ADC/DAC, TTL I/O, Counter

Version 1.0.x

User Manual

Issue 1.0.0

September 2014

TEWS TECHNOLOGIES GmbH

Am Bahnhof 7 25469 Halstenbek, Germany

Phone: +49 (0) 4101 4058 0 Fax: +49 (0) 4101 4058 19

e-mail: info@tews.com www.tews.com

TPMC851-SW-25

Integrity Device Driver

Multifunction I/O
16 bit ADC/DAC, TTL I/O, Counter

Supported Modules:
TPMC851-10

Issue	Description	Date
1.0.0	First Issue	September 15, 2014

Table of Contents

1 Introduction

The TPMC851-SW-25 Integrity device driver software allows the operation of the supported PMC conforming to the Integrity I/O system specification.

The driver software uses mutual exclusion to prevent simultaneous requests by multiple tasks from interfering with each other.

The TPMC851-SW-25 device driver supports the following features:

- Reading an ADC input value from a specified channel
- Configuring and using the ADC input sequencer
- Setting a DAC output value for a specified channel
- Configuring and using the DAC output sequencer
- Reading from digital I/O input register
- Writing to digital I/O output register
- Waiting for I/O input event (high, low or any transition on input line)
- Configuring I/O line direction
- Reading counter value
- Reset counter value
- Setting counter preload and match value
- Configuring counter mode
- Wait for counter match and control event

The TPMC851-SW-25 supports the modules listed below:

TPMC851-10	8x 16-bit DAC, 32x 16-bit ADC, 16x Digital Inputs/Outputs, 1x Counter/Timer	(PMC)

To get more information about the features and use of supported devices it is recommended to read the manuals of the supported modules listed below.

TPMC851 User Manual

2 Installation

The following files are located on the distribution media:

Directory path TPMC851-SW-25:

tpmc851.c	Device driver source
tpmc851def.h	Driver include file
tpmc851.h	Include file for driver and application
tpmc851api.c	Application interface, simplifies device access
tpmc851api.h	Include file for API and applications
example/*.c	Path with example applications
TPMC851-SW-25-1.0.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Driver Installation

Copy the TPMC851 driver files (tpmc851.c, tpmc851def.h, tpmc851api.h, and tpmc851.h) into a desired driver or project path. The driver source file tpmc851.c must be included into the kernel project and the BSP path must be added to the include search paths of the file. (Set Options… ➔ Project → Include Directories, than double click and add a new path and select …/int5011/bsp)

Afterwards the project must be rebuilt. The driver will be started automatically after booting the image and the driver will be requested if a matching device is detected in the system.

For further information about building a kernel, please refer to MULTI and INTEGRITY Documentation.

2.2 TPMC851 Applications

Copy the TPMC851 API files (tpmc851api.c, tpmc851api.h, and tpmc851.h) into a desired application path, and include tpmc851api.c into the application project.

The application source file must include tpmc851api.h. If these steps are done, the TPMC851 API can be used and the devices will be accessible.

3 API Documentation

3.1 General Functions

3.2 tpmc851Open

NAME

tpmc851Open() – open a device.

SYNOPSIS

```
TPMC851_HANDLE tpmc851Open
(
      char          *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a file descriptor must be opened by a call to this function.

PARAMETER

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first TPMC851 device is named “tpmc851_0”, the second device is named “tpmc851_1” and so on.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;

/*
** open file descriptor to device
*/
hdl = tpmc851Open(“tpmc851_0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURN VALUE

A device descriptor pointer or NULL if the function fails.

3.3 tpmc851Close

NAME

tpmc851Close() – close a device.

SYNOPSIS

```
TPMC851_STATUS tpmc851Close
(
     TPMC851_HANDLE       hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETER

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*
** close the device
*/
result = tpmc851Close(hdl);
if (result != TPMC851_OK)
{
     /* handle close error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified device handle is invalid

3.4 Device Access Functions

3.4.1 tpmc851AdcRead

NAME

tpmc851AdcRead – Read value from ADC channel

SYNOPSIS

```
TPMC851_STATUS tpmc851AdcRead
(
    TPMC851_HANDLE      hdl,
    int                 channel,
    int                 gain,
    unsigned int        flags,
    short               *pAdcValue
)
```

DESCRIPTION

This function starts an ADC conversion with specified parameters, waits for completion and returns the value.

The ADC sequencer must be stopped for single ADC conversions.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

channel

Specifies the ADC channel number. Valid values are 1..16 for differential input and 1..32 for single-ended input.

gain

Specifies the input gain. Valid gain values are 1, 2, 4, and 8.

flags

This is an ORed value of the following flags:

Flag	Description
TPMC851_F_CORR	If set the function will return a corrected value of the input data in *adcValue*. Factory set and module dependent correction data is used for correction. If not set, the raw value read from the module will be returned in *adcValue*.
TPMC851_F_IMMREAD	If set the driver will start the conversion without waiting for settling time. This should only be used if the previous conversion has used the same interface parameters (channel, gain, differential/single-ended). If not set the driver will use the automatic mode, which sets interface configuration, waits settling time and then starts the conversion.
TPMC851_F_DIFF	If set the input channel will be a differential input. If not set the input channel will be a single-ended input.

pAdcValue

This parameter specifies a pointer to a *short* value which receives the current ADC value.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;
short              AdcValue;

/*---------------------------------------------------------------------
  Read a corrected value from differential channel 2 using gain of 4
  ---------------------------------------------------------------------*/
result = tpmc851AdcRead(
             hdl,
             2,                                 /* Channel    */
             4,                                 /* Gain       */
             TPMC851_F_CORR | TPMC851_F_DIFF,   /* Flags      */
             &AdcValue );                       /* ADC value  */
if (result == TPMC851_OK)
{
     /* function succeeded */
     printf("    ADC-value: %d", AdcValue);
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_BUSY	The ADC sequencer is currently running
TPMC851_ERR_INVAL	Invalid flags or gain value specified
TPMC851_ERR_ACCESS	Invalid ADC channel number specified
TPMC851_ERR_TIMEOUT	ADC conversion timed out

3.4.2 tpmc851AdcSeqConfig

NAME

tpmc851AdcSeqConfig – Configure ADC sequencer channel

SYNOPSIS

```
TPMC851_STATUS tpmc851AdcSeqConfig
(
    TPMC851_HANDLE      hdl,
    int                 channel,
    int                 enable,
    int                 gain,
    unsigned int        flags
)
```

DESCRIPTION

This function enables and configures, or disables an ADC channel for sequencer use.

The ADC sequencer must be stopped to execute this function.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

channel

Specifies the ADC channel number to configure. Valid values are 1..16 for differential input and 1..32 for single-ended input.

enable

Specifies if the channel shall be used in sequencer mode or not. (0 disables the channel, any other value will enable the channel)

gain

Specifies the input gain. Valid gain values are 1, 2, 4, and 8.

flags

Is an ORed value of the following flags:

Flag	Description
TPMC851_F_CORR	If set the sequencer will return a corrected value for the specified channel. Factory set and module dependent correction data is used for correction. If not set, the raw value read from the module will be returned.
TPMC851_F_DIFF	If set the input channel will be a differential input. If not set the input channel will be a single-ended input.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*-------------------------------------------------------
  Configure single-ended channel 3, using a gain of 4 and
  returning corrected data when the sequencer is running
  -------------------------------------------------------*/
result = tpmc851AdcSeqConfig(
              hdl,
              3,                    /* Channel    */
              1,                    /* Enable     */
              4,                    /* Gain       */
              TPMC851_F_CORR );     /* Flags      */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flags or gain value specified
TPMC851_ERR_ACCESS	Invalid ADC channel number specified
TPMC851_ERR_BUSY	The ADC sequencer is currently running

3.4.3 tpmc851AdcSeqStart

NAME

tpmc851AdcSeqStart – Start ADC Sequencer

SYNOPSIS

```
TPMC851_STATUS tpmc851AdcSeqStart
(
    TPMC851_HANDLE              hdl,
    unsigned short              cycTime,
    unsigned int                flags,
    TPMC851_ADC_SEQDATA_BUF     *pAdcSeqBuf
)
```

DESCRIPTION

This function configures the ADC sequencer time and starts the ADC sequencer.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

cycTime

Specifies the ADC sequencer cycle time. The sequencer time is specified in 100µs steps. With a value of 0, the “Sequencer Continuous Mode” is selected.

flags

One of the following optional flags is possible:

Flag	Description
TPMC851_F_EXTTRIGSRC	If set the ADC sequencer is triggered with digital I/O line 0. If not set, the ADC sequencer uses the ADC cycle counter.
TPMC851_F_EXTTRIGOUT	If set the ADC trigger is used as output on digital I/O line 0.

adcSeqBuf

Points to an ADC sequencer buffer. The buffer is used to store the ADC data. The ADC sequencer buffer is defined as a structure named *TPMC851_ADC_SEQDATA_BUF*. This structure is defined in *tpmc851api.h*. The size of the buffer is variable, therefore a macro *TPMC851_CALC_SIZE_ADC_SEQDATA_BUF(s)* is defined in *tpmc851api.h*, that calculates the size to allocate for the buffer. The macro-parameter *s* specifies the size of the *buffer* array of the structure.

```
typedef struct
{
    int             putIdx;
    int             getIdx;
    int             bufSize;
    unsigned int    seqState;
    short           buffer[1];
} TPMC851_ADC_SEQDATA_BUF;
```

putIdx

Specifies the index into *buffer* where the next data will be stored to. This index is handled by the driver and should only be read by the application to check if data is available. The driver initializes this index to 0 when sequencer starts.

getIdx

Specifies the index into *buffer* where the next input data can be read from. This index must be handled by the application and is only be read by the driver to check a FIFO overflow. The driver initializes this index to 0 when sequencer starts.

bufSize

Specifies the array size of *buffer*. This value must be the same as used for *s* in *TPMC851_CALC_SIZE_ADC_SEQDATA_BUF(s)* when calculating the allocation size for *adcSeqBuf*.

seqState

Returns the sequencer state. This is an ORed value of the following status flags:

Flag	Description
TPMC851_SF_SEQACTIVE	If set the ADC sequencer is started. If not set, the ADC sequencer stopped.
TPMC851_SF_SEQOVERFLOWERR	If set the ADC sequencer has detected an overflow error. (Hardware detected)
TPMC851_SF_SEQTIMERERROR	If set the ADC sequencer has detected a timer error. (Hardware detected)
TPMC851_SF_SEQIRAMERROR	If set the ADC sequencer has detected an instruction RAM error. (Hardware detected)
TPMC851_SF_SEQFIFOOVERFLOW	If set the application supplied FIFO (*buffer*) has overrun. Data got lost.

buffer

Array used for ADC sequencer data FIFO.
The ADC data is stored by the sequencer into this FIFO. The assignment from data to channel is done as follows. The first data will be from the lowest enabled channel, the second from the next enabled channel and so on. There will be no data stored for disabled channels. If the end of *buffer* is reached the next data will be stored again at the beginning of the buffer.

Example:
Enabled channels: 1, 2, 5
Buffer Size: 10
The table shows the index the data is stored to for channel and cycle.

Sequencer Cycle	Channel 1	Channel 2	Channel 5
1st	0	1	2
2nd	3	4	5
3rd	6	7	8
4th	9	0	1
5th	2	3	4
…	…	…	…

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE              hdl;
TPMC851_STATUS              result;
TPMC851_ADC_SEQDATA_BUF     *seqBuf;
int                         realBufSize;

/*------------------------------
  allocate Buffer (100 word FIFO)
  ------------------------------*/
realBufSize = TPMC851_CALC_SIZE_ADC_SEQDATA_BUF(100);
seqBuf = (TPMC851_ADC_SEQDATA_BUF*)malloc(realBufSize);

seqBuf->bufSize = 100;

…
```

…

```
/*-----------------------------------------------
  Start sequencer with a buffer of 100 words and
  a cycle time of 100 ms, do not use external trigger
  -----------------------------------------------*/
result = tpmc851AdcSeqStart(
              hdl,
              1000,             /* Cycle Time (in 100us)    */
              0,                /* Flags                    */
              seqBuf );         /* Sequencer buffer         */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_BUSY	The ADC sequencer is currently running
TPMC851_ERR_INVAL	Invalid flag specified

3.4.4 tpmc851AdcSeqStop

NAME

tpmc851AdcSeqStop – Stop ADC Sequencer

SYNOPSIS

```
TPMC851_STATUS tpmc851AdcSeqStop
(
     TPMC851_HANDLE       hdl
)
```

DESCRIPTION

This function stops the ADC sequencer. All sequencer channel configurations are still valid after stopping.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*------------------
  Stop ADC sequencer
  ------------------*/
result = tpmc851AdcSeqStop( hdl );

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_ACCESS	The sequencer is not running

3.4.5 tpmc851DacWrite

NAME

tpmc851DacWrite – Write to DAC channel

SYNOPSIS

```
TPMC851_STATUS tpmc851DacWrite
(
    TPMC851_HANDLE      hdl,
    int                 channel,
    unsigned int        flags,
    short               DacValue
)
```

DESCRIPTION

This function writes a value to the DAC register and starts the conversion if specified.

The DAC sequencer must be stopped for single DAC writes.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

channel

Specifies the DAC channel number. Valid values are 1..8.

flags

Is an ORed value of the following flags:

Flag	Description
TPMC851_F_CORR	If set the function will correct the *dacValue* before writing to DAC channel. Factory set and module dependent correction data is used for correction. If not set, *dacValue* is written to the DAC channel.
TPMC851_F_NOUPDATE	If set the DACs will not update after changing the DAC value. The output voltage will change with the next write with unset *TPMC851_F_NOUPDATE* flag. If not set the DAC will immediately convert and output the new voltage.

DacValue

This value is written to the DAC channel.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*-------------------------------------------------------------
  Write uncorrected 0x4000 to DAC channel 5, immediate convert
  -------------------------------------------------------------
result = tpmc851DacWrite(
              hdl,
              5,                /* Channel    */
              0,                /* Flags      */
              0x4000 );         /* DAC value  */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flag specified
TPMC851_ERR_ACCESS	Invalid DAC channel number specified

3.4.6 tpmc851DacSeqConfig

NAME

tpmc851DacSeqConfig – Configure DAC sequencer channel

SYNOPSIS

```
TPMC851_STATUS tpmc851DacSeqConfig
(
    TPMC851_HANDLE      hdl,
    int                 channel,
    int                 enable,
    unsigned int        flags
)
```

DESCRIPTION

This function enables and configures, or disables a DAC channel for sequencer use.

The DAC sequencer must be stopped to execute this function.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

channel

Specifies the DAC channel number to configure. Valid values are 1..8.

enable

Specifies if the channel shall be used in sequencer mode or not. (0 disables the channel, any other value will enable the channel)

flags

The following optional flag is possible:

Flag	Description
TPMC851_F_CORR	If set the function will correct the dacValue before writing to DAC channel. Factory set and module dependent correction data is used for correction. If not set, dacValue is written to the DAC channel.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*-----------------------------------------------------
  Configure DAC channel 1, using corrected data while
  the sequencer is running
  -----------------------------------------------------*/
result = tpmc851DacSeqConfig(
              hdl,
              1,                      /* Channel    */
              1,                      /* Enable     */
              TPMC851_F_CORR );       /* Flags      */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flag specified
TPMC851_ERR_ACCESS	Invalid DAC channel number specified

3.4.7 tpmc851DacSeqStart

NAME

tpmc851DacSeqStart – Start DAC Sequencer

SYNOPSIS

```
TPMC851_STATUS tpmc851DacSeqStart
(
    TPMC851_HANDLE              hdl,
    unsigned short              cycTime,
    unsigned int                flags,
    TPMC851_DAC_SEQDATA_BUF     *pDacSeqBuf
)
```

DESCRIPTION

This function configures the DAC sequencer time and starts the DAC sequencer.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

cycTime

Specifies the DAC sequencer cycle time. The sequencer time is specified in 100µs steps. With a value of 0, the “Sequencer Continuous Mode” is selected.

flags

Is an ORed value of the following flags:

Flag	Description
TPMC851_F_EXTTRIGSRC	If set the DAC sequencer is triggered with digital I/O line 1. If not set, the DAC sequencer uses the DAC cycle counter.
TPMC851_F_EXTTRIGOUT	If set the DAC trigger is used as output on digital I/O line 1.
TPMC851_F_DACSEQREPEAT	If set the DAC will repeat data when the end of the buffer is reached, the error *TPMC851_SF_SEQFIFOUNDERFLOW* is suppressed.

TPMC851_F_EXTTRIGSRC* and *TPMC851_F_EXTTRIGOUT cannot be used at the same time.

dacSeqBuf

Points to a DAC sequencer buffer. The buffer is used to supply the DAC data to the driver. The DAC sequencer buffer is defined as a structure named *TPMC851_DAC_SEQDATA_BUF*. This structure is defined in *tpmc851api.h*. The size of the buffer is variable, therefore a macro *TPMC851_CALC_SIZE_DAC_SEQDATA_BUF(s)* is defined in *tpmc851api.h*, that calculates the size to allocate for the buffer. The macro-parameter *s* specifies the size of the *buffer* array of the structure.

```
typedef struct
{
    int             putIdx;
    int             getIdx;
    int             bufSize;
    unsigned int    seqState;
    short           buffer[1];
} TPMC851_DAC_SEQDATA_BUF;
```

putIdx

Specifies the index into *buffer* where the next data shall be stored. This index must be handled by the application and is only be read by the driver to check a FIFO underrun.

getIdx

Specifies the index into *buffer* where the next data will be read from. This index is handled by the driver and should only be read by the application to check if there is space for new data.

bufSize

Specifies the array size of *buffer*. This value must be the same as used for *s* in *TPMC851_CALC_SIZE_ADC_SEQDATA_BUF(s)* when calculating the allocation size for *adcSeqBuf*.

seqState

Returns the sequencer state. This is an ORed value of the following status flags:

Flags	Description
TPMC851_SF_SEQACTIVE	If set the DAC sequencer is started. If not set, the DAC sequencer stopped.
TPMC851_SF_SEQUNDERFLOWERR	If set the DAC sequencer has detected an underrun error. (Hardware detected)
TPMC851_SF_SEQFIFOUNDERFLOW	If set the application supplied FIFO (*buffer*) is empty and the sequencer could not read new data from FIFO.

buffer

Array used for DAC sequencer data FIFO.
The DAC data is stored by the application into this FIFO. The assignment from data to channel is done as follows. The first data will be used for the lowest enabled channel, the second from the next enabled channel and so on. There will be no data used for disabled channels. If the end of *buffer* is reached the next data will be read again from the beginning of the buffer.

Example:
Enabled channels: 1, 2, 5
Buffer Size: 10
The table shows the index the data is used for channel and cycle.

Sequencer Cycle	Channel 1	Channel 2	Channel 5
1st	0	1	2
2nd	3	4	5
3rd	6	7	8
4th	9	0	1
5th	2	3	4
…	…	…	…

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE              hdl;
TPMC851_STATUS              result;
TPMC851_DAC_SEQDATA_BUF     *seqBuf;
int                         realBufSize;

/*------------------------------
  allocate Buffer (100 word FIFO)
  ------------------------------*/
realBufSize = TPMC851_CALC_SIZE_DAC_SEQDATA_BUF(100);
seqBuf = (TPMC851_DAC_SEQDATA_BUF*)malloc(realBufSize);

seqBuf->bufSize = 100;

/*-----------
  Fill buffer
  -----------*/
seqBuf->buffer[0] = …;
seqBuf->buffer[1] = …;
seqBuf->buffer[2] = …;

…
```

…

```
/*-------------------------------------------------------------
  Start sequencer with a buffer of 100 words and
  a cycle time of 100 ms, do not use external trigger, repeat data
  -------------------------------------------------------------*/
result = tpmc851DacSeqStart(
              hdl,
              1000,                       /* Cycle Time (in 100us)    */
              TPMC851_F_DACSEQREPEAT,     /* Flags                    */
              seqBuf );                   /* Sequencer buffer         */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_BUSY	The DAC sequencer is currently running
TPMC851_ERR_INVAL	Invalid flag specified

3.4.8 tpmc851DacSeqStop

NAME

tpmc851DacSeqStop – Stop DAC Sequencer

SYNOPSIS

```
TPMC851_STATUS tpmc851DacSeqStop
(
     TPMC851_HANDLE       hdl
)
```

DESCRIPTION

This function stops the DAC sequencer. All sequencer channel configurations are still valid after stopping.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*------------------
  Stop DAC sequencer
  ------------------*/
result = tpmc851DacSeqStop( hdl );

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_ACCESS	The sequencer is not running

3.4.9 tpmc851IoRead

NAME

tpmc851IoRead – Read from digital I/O

SYNOPSIS

```
TPMC851_STATUS tpmc851IoRead
(
    TPMC851_HANDLE      hdl,
    unsigned short      *pIoValue
)
```

DESCRIPTION

This function reads the current value of digital I/O input.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

pIoValue

This parameter specifies a pointer to an *unsigned short* value which receives the current I/O value. Bit 0 corresponds to the first I/O line, bit 1 corresponds to the second I/O line and so on.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;
unsigned short     IoValue;

…
```

…

```
/*--------------------
  Read I/O input value
  --------------------*/
result = tpmc851IoRead(
              hdl,
              &IoValue );                        /* I/O value   */

if (result == TPMC851_OK)
{
     /* function succeeded */
     printf("    I/O-value: 0x%04X", IoValue);
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.4.10 tpmc851IoWrite

NAME

tpmc851IoWrite – Write to digital I/O

SYNOPSIS

```
TPMC851_STATUS tpmc851IoWrite
(
      TPMC851_HANDLE      hdl,
      unsigned short      IoValue
)
```

DESCRIPTION

This function writes a value to digital I/O output.

Only I/O lines configured for output will be affected. Please refer to chapter 3.4.11 tpmc851IoConfig.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

IoValue

This value is written to the I/O output. Bit 0 corresponds to the first I/O line, bit 1 corresponds to the second I/O line and so on.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

```
…

/*------------------------------
  Write I/O output value 0x1234
  ------------------------------*/
result = tpmc851IoWrite(
              hdl,
              0x1234 );                          /* I/O value  */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.4.11 tpmc851IoConfig

NAME

tpmc851IoConfig – Configure direction of digital I/O

SYNOPSIS

```
TPMC851_STATUS tpmc851IoConfig
(
    TPMC851_HANDLE      hdl,
    unsigned short      Direction
)
```

DESCRIPTION

This function configures the direction (input/output) of the digital I/O lines.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

Direction

Specifies the new direction setting for digital I/O. A bit set to 1 enables output, a 0 means that the I/O line is input. Bit 0 corresponds to the first I/O line, bit 1 corresponds to the second I/O line and so on.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

```
…

/*-------------------------------------------------------
  Enable line 0,2,8 for output, all other lines are input
  -------------------------------------------------------*/
result = tpmc851IoConfig(
              hdl,
              (1 << 0) | (1 << 2) | (1 << 8) ); /* Direction */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.4.12 tpmc851IoDebConfig

NAME

tpmc851IoDebConfig – Configure digital I/O (input) debouncer

SYNOPSIS

```
TPMC851_STATUS tpmc851IoDebConfig
(
    TPMC851_HANDLE      hdl,
    unsigned short      EnableMask,
    unsigned short      DebounceTime
)
```

DESCRIPTION

This function configures the digital I/O input debouncing mechanism to avoid detection of invalid signal changes in noisy environments.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EnableMask

Specifies digital I/O lines to be filtered by the debouncing mechanism. A bit set to 1 enables the debouncer, and a 0 disables the debouncer for the corresponding I/O line. Bit 0 corresponds to the first I/O line, bit 1 corresponds to the second I/O line and so on.

DebounceTime

Specifies the debounce time. The time is specified in 100ns steps, using the following formula:
Debounce duration = (DebounceTimeValue * 100ns) + 100ns

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

```
…

/*-------------------------------------------------------
  Enable Debouncer for line 1 and 2 (debounce time 1ms)
  -------------------------------------------------------*/
result = tpmc851IoDebConfig(
              hdl,
              (1 << 1) | (1 << 2),    /* EnableMask                    */
              10000 );                /* DebounceTime (in 100ns steps) */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.4.13 tpmc851IoEventWait

NAME

tpmc851IoEventWait – Wait for I/O event

SYNOPSIS

```
TPMC851_STATUS tpmc851IoEventWait
(
    TPMC851_HANDLE      hdl,
    int                 IoLine,
    unsigned int        flags,
    int                 timeout
)
```

DESCRIPTION

This function waits for an I/O input event.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

IoLine

Specifies the digital I/O line where the event shall occur. Valid values are 0..15.

flags

Specifies the event that shall occur. This is an ORed value of the following flags:

Flag	Description
TPMC851_F_HI2LOTRANS	If set, the function will return after a high to low transition occurs.
TPMC851_F_LO2HITRANS	If set, the function will return after a low to high transition occurs.

At least one flag must be specified.

timeout

Specifies the maximum time the function will wait for the specified event. The time is specified in milliseconds. Specify -1 to wait indefinitely.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*-------------------------------------------------------------
  Wait for any transition on I/O line 12 (max wait 10 sec)
  -------------------------------------------------------------*/
result = tpmc851IoEventWait(
              hdl,
              12,                                          /* IoLine     */
              TPMC851_F_HI2LOTRANS | TPMC851_F_LO2HITRANS,/* Flags     */
              10000 );                                     /* Timeout    */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flag specified
TPMC851_ERR_ACCESS	Invalid I/O line specified
TPMC851_ERR_TIMEOUT	Timeout has occurred
TPMC851_ERR_BUSY	A task is already waiting for an event on the specified I/O line

3.4.14 tpmc851CntRead

NAME

tpmc851CntRead – Read counter/timer value

SYNOPSIS

```
TPMC851_STATUS tpmc851CntRead
(
    TPMC851_HANDLE      hdl,
    unsigned int        *pCounterValue,
    unsigned int        *pCounterStatus
)
```

DESCRIPTION

This function reads the value of the counter.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

pCounterValue

This parameter is a pointer to an *unsigned int* data buffer where the current counter value is stored.

pCounterStatus

This parameter is a pointer to an *unsigned int* data buffer where the counter status is returned. If possible the flags are cleared after read. This is an ORed value of the following flags.

Flag	Description
TPMC851_SF_CNTBORROW	Counter borrow bit set (current state)
TPMC851_SF_CNTCARRY	Counter carry bit set (current state)
TPMC851_SF_CNTMATCH	Counter match event has occurred since last read.
TPMC851_SF_CNTSIGN	Counter sign bit (current state)
TPMC851_SF_CNTDIRECTION	If set, counter direction is upward. If not set, counter direction is downward.
TPMC851_SF_CNTLATCH	Counter value has been latched.
TPMC851_SF_CNTLATCHOVERFLOW	Counter latch overflow has occurred.
TPMC851_SF_CNTSNGLCYC	Counter Single Cycle is active

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;
unsigned int       CounterValue;
unsigned int       CounterStatus;


/*------------------
  Read counter value
  ------------------*/
result = tpmc851CntRead(
              hdl,
              &CounterValue,                  /* Counter Value   */
              &CounterStatus );               /* Counter Status  */

if (result == TPMC851_OK)
{
     /* function succeeded */
     printf("    Counter: %d", CounterValue);
     printf("    State:   %Xh", CounterStatus);
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.4.15 tpmc851CntConfig

NAME

tpmc851CntConfig – Configure counter

SYNOPSIS

```
TPMC851_STATUS tpmc851CntConfig
(
    TPMC851_HANDLE      hdl,
    unsigned int        inputMode,
    int                 clockDivider,
    unsigned int        countMode,
    unsigned int        controlMode,
    unsigned int        invFlags
)
```

DESCRIPTION

This function configures the counter.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

inputMode

Specifies the counter input mode. The following modes are defined and valid:

Flag	Description
TPMC851_M_CNTIN_DISABLE	Counter disabled
TPMC851_M_CNTIN_TIMERUP	Timer Mode Up
TPMC851_M_CNTIN_TIMERDOWN	Timer Mode Down
TPMC851_M_CNTIN_DIRCOUNT	Direction Count
TPMC851_M_CNTIN_UPDOWNCOUNT	Up/Down Count
TPMC851_M_CNTIN_QUAD1X	Quadrature Count 1x
TPMC851_M_CNTIN_QUAD2X	Quadrature Count 2x
TPMC851_M_CNTIN_QUAD4X	Quadrature Count 4x

clockDivider

Specifies clock divider for Timer Mode. Allowed clock divider values are:

Clock Divider Value	Clock Input Frequency
1	40 MHz
2	20 MHz
4	10 MHz
8	5 MHz

countMode

Specifies the count mode. The following modes are defined and valid:

Flag	Description
TPMC851_M_CNT_CYCLE	Cycling Counter
TPMC851_M_CNT_DIVN	Divide-by-N
TPMC851_M_CNT_SINGLE	Single Cycle

controlMode

Specifies the counter control mode. These events can generate counter control events. The following modes are defined and valid:

Flag	Description
TPMC851_M_CNTCTRL_NONE	No Control Mode
TPMC851_M_CNTCTRL_LOAD	Load Mode
TPMC851_M_CNTCTRL_LATCH	Latch Mode
TPMC851_M_CNTCTRL_GATE	Gate Mode
TPMC851_M_CNTCTRL_RESET	Reset Mode

invFlags

Specifies if counter input lines shall be inverted or not. This is an ORed value of the following flags:

Flag	Description
TPMC851_F_CNTINVINP2	If set, input line 2 is low active If not set, input line 2 is high active
TPMC851_F_CNTINVINP3	If set, input line 3 is low active If not set, input line 3 is high active
TPMC851_F_CNTINVINP4	If set, input line 4 is low active If not set, input line 4 is high active

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*-----------------------------------------------------
  Setup counter for direction count, clock divider 1,
  cycling count, no control mode and all lines high active
  -----------------------------------------------------*/
result = tpmc851CntConfig(
              hdl,
              TPMC851_M_CNTIN_DIRCOUNT,    /* inputMode       */
              1,                           /* clockDivider    */
              TPMC851_M_CNT_CYCLE,         /* countMode       */
              TPMC851_M_CNTCTRL_NONE,      /* controlMode     */
              0 );                         /* invFlags        */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid mode or flag specified

3.4.16 tpmc851CntReset

NAME

tpmc851CntReset – Reset counter

SYNOPSIS

```
TPMC851_STATUS tpmc851CntReset
(
     TPMC851_HANDLE       hdl
)
```

DESCRIPTION

This function resets the counter value to 0x00000000.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*-------------------
  Reset counter value
  -------------------*/
result = tpmc851CntReset( hdl );

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.4.17 tpmc851CntSetPreload

NAME

tpmc851CntSetPreload – Set counter preload value

SYNOPSIS

```
TPMC851_STATUS tpmc851CntSetPreload
(
    TPMC851_HANDLE      hdl,
    unsigned int        PreloadValue,
    unsigned int        PreloadFlags
)
```

DESCRIPTION

This function sets the counter preload value, either immediately or on the next preload condition.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

PreloadValue

Specifies the new counter preload value.

PreloadFlags

The following flag is optional:

Flag	Description
TPMC851_F_IMMPRELOAD	If set, the function will immediately load the preload value into the counter If not set, preload value will be used for the next preload condition.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;


/*----------------------------------------
  Immediately load 0x11223344 into the counter
  and preload register
  ----------------------------------------*/
result = tpmc851CntSetPreload(
              hdl,
              0x11223344,                    /* Preload Value   */
              TPMC851_F_IMMPRELOAD );        /* Flags           */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flag specified

3.4.18 tpmc851CntSetMatch

NAME

tpmc851CntSetMatch – Set counter match value

SYNOPSIS

```
TPMC851_STATUS tpmc851CntSetMatch
(
    TPMC851_HANDLE      hdl,
    unsigned int        MatchValue
)
```

DESCRIPTION

This function sets the counter match value. If counter and match value are the same, a match event occurs. The driver can wait for this event.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

MatchValue

Specifies the new counter match value.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

```
…

/*--------------------------
  Set match value to 0x10000
  --------------------------*/
result = tpmc851CntSetMatch(
              hdl,
              0x10000 );              /* MatchValue */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.4.19 tpmc851CntMatchWait

NAME

tpmc851CntMatchWait – Wait for counter match event

SYNOPSIS

```
TPMC851_STATUS tpmc851CntMatchWait
(
      TPMC851_HANDLE      hdl,
      int                 timeout
)
```

DESCRIPTION

This function waits for a counter match event.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

timeout

Specifies the maximum time the function will wait for the counter match event. The time is specified in milliseconds. Specify -1 to wait indefinitely.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

…

```
/*-----------------------------------------------------
  Wait for counter match event (max wait 10000 milliseconds)
  -----------------------------------------------------*/
result = tpmc851CntMatchWait(
              hdl,
              10000 );                                  /* Timeout    */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_BUSY	A task is already waiting for a counter match event
TPMC851_ERR_TIMEOUT	Timeout has occurred

3.4.20 tpmc851CntCtrlWait

NAME

tpmc851CntCtrlWait – Wait for counter control event

SYNOPSIS

```
TPMC851_STATUS tpmc851CntCtrlWait
(
      TPMC851_HANDLE      hdl,
      int                 timeout
)
```

DESCRIPTION

This function waits for counter control event. The event to wait for is chosen with API function *tpmc851CntConfig(),* specifying the parameter *controlMode*.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

timeout

Specifies the maximum time the function will wait for the counter control event. The time is specified in milliseconds. Specify -1 to wait indefinitely.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

…

```
/*-----------------------------------------------------
  Wait for counter control event (max wait 10000 milliseconds)
  -----------------------------------------------------*/
result = tpmc851CntCtrlWait(
              hdl,
              10000 );                                  /* Timeout    */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_BUSY	A task is already waiting for a counter control event
TPMC851_ERR_TIMEOUT	Timeout has occurred

4 Appendix

4.1 Example Applications

The example application shall give an overview about the use of the TPMC851 devices and how to use the TPMC851 API.

All delivered examples are written to work with the same external wiring. This wiring is described in 4.2 Example Wiring.

4.1.1 tpmc851io.c

This example configures, writes output value and reads the state of the I/O lines and it waits for events on I/O lines.

- ➔ open devices
- ➔ configure I/O line direction (0..7 output, 8..15 input)
- ➔ loop writing output data and afterwards read and print out the I/O state
- ➔ set outputs to 0
- ➔ start tasks waiting for different events (see below)
- ➔ generate events the tasks are waiting for
- ➔ close devices

Event waiting tasks:

- ➔ open devices
- ➔ wait for an event (different transitions)
- ➔ print out a message that event has occurred
- ➔ close devices

4.1.2 tpmc851dac_on.c

This simple example writes to the analog output register and starts the DA conversion. This example sets the analog output lines to a voltage.

- ➔ open devices
- ➔ set a voltage different to 0V on each of the DAC outputs
- ➔ close devices

4.1.3 tpmc851dac_off.c

This simple example writes to the analog output register and start the DA conversion. This example sets the analog output lines to 0.0V.

- ➔ open devices
- ➔ set a voltage of 0V on each of the DAC outputs
- ➔ close devices

4.1.4 tpmc851adc.c

This simple example executes AD conversion for the analog input lines and prints the read values and voltages.

Program flow:

- ➔ open devices
- ➔ loop reading ADC input voltages (differential, correction and gain = 1)
- ➔ loop reading ADC input voltages (single-ended and gain = 2)
- ➔ loop reading ADC input voltages (differential, correction and gain = 4)
- ➔ loop reading ADC input voltages (single-ended and gain = 8)
- ➔ close devices

4.1.5 tpmc851seq.c

This example configures and starts both sequencers (DAC and ADC). It prints out the read data provided by the ADC sequencer.

- ➔ open devices
- ➔ setup channels for DAC sequencer (enable channel 1,2,3,4 and 8)
- ➔ start DAC sequencer (cycle time: 2s, repeat output)
- ➔ create and start ADC sequencer task (see below)
- ➔ wait and let DAC sequencer work
- ➔ signalize ADC task to stop
- ➔ stop DAC sequencer
- ➔ close devices

ADC sequencer task:

- ➔ open devices
- ➔ setup channels for ADC sequencer (enable channel 1,2,16, 18 and 32)
- ➔ start ADC sequencer (cycle time: 0,3s)
- ➔ loop and printout every second the available data
- ➔ break loop when signalized by main()
- ➔ stop ADC sequencer
- ➔ close devices

4.1.6 tpmc851cnt.c

This example configures the counter to count up/down mode with gate. The I/O lines are generating counting events while the counter gate is enabled or disabled.

- ➔ open devices
- ➔ disable counter
- ➔ configure I/O lines (for counter signal generation)
- ➔ read and print counter value and state (disabled counter)
- ➔ reset counter
- ➔ read and print counter value and state (reset and disabled counter)
- ➔ configure counter (up/down, gate controlled)
- ➔ read and print counter value and state (configure and enabled counter)
- ➔ generate counter signal on I/O lines (count up, gate enabled)
- ➔ read and print counter value and state
- ➔ generate counter signal on I/O lines (count down, gate enabled)
- ➔ read and print counter value and state
- ➔ generate counter signal on I/O lines (count down, gate disabled)
- ➔ read and print counter value and state
- ➔ generate counter signal on I/O lines (count up, gate enabled)
- ➔ read and print counter value and state
- ➔ disable counter
- ➔ close devices

4.1.7 tpmc851timer.c

This example configures the counter to work as a timer running with 5MHz. The counter value will be reset, preloaded and the counter value will be read. Counter match events will be generated while waiting for the event or no event occurs and the wait times out.

- ➔ open devices
- ➔ disable counter
- ➔ read and print counter value and state (disabled counter)
- ➔ reset counter
- ➔ read and print counter value and state (reset and disabled counter)
- ➔ preload counter (immediately)
- ➔ set counter match value
- ➔ read and print counter value and state (preloaded and disabled counter)
- ➔ configure counter (timer counting up, 5MHz, cycling)
- ➔ loop for a while and read set counter value and state (counting up)
- ➔ change counter match value
- ➔ wait for counter match event (should occur)
- ➔ change counter match value
- ➔ wait for counter match event (should not occur in specified timeout) →Timeout error
- ➔ disable counter
- ➔ read and print counter value and state (preloaded and disabled counter)
- ➔ close devices

4.2 Example Wiring

The tables below show a wiring which will work for the example applications.

4.2.1 DAC and ADC

DAC			ADC		
Channel	**Pin**		**Pin**	**SE-Channel**	**Diff-Channel**
#1	19	→	1	#1	#1+
#4	23	→	35	#17	#1-
#2	20	→	2	#2	#2+
#8	57	→	36	#18	#2-
Gnd	9	→	3	#3	#3+
Gnd	43	→	37	#19	#3-
#3	22	→	17	#16	#16+
Gnd	52	→	51	#32	#16-

4.2.2 Digital I/O and Counter

IO-Lines				
IO Line	**Pin**		**Pin**	**IO Line**
#0	25	↔	64	#12
#1	26	↔	65	#13
#2	27	↔	66	#14
#3	28	↔	67	#15
#4	30	↔	59	#8
#5	31	↔	60	#9
#6	32	↔	61	#10
#7	33	↔	62	#11