AcQuisition Technology bv

Headquarters:
Raadhuislaan 27a
5341 GL Oss

Postal address:
P.O Box 627
5340 AP Oss
The Netherlands

Phone: +31-412-651055
Fax: +31-412-651050
E-mail: info@acq.nl
Web: http://www.acq.nl

M385

Universal Timer Counter M-module

User Manual

Version 2.4

CONTENTS

1. Introduction

1.1. Validity of the Manual

This document is of revision 2.4 and provides information on the use of the M385/R1.x.

The M-module revision number is composed as Rx.y where x is the revision of the PCB and y is the revision of the CPLD firmware.

1.2. Purpose

This manual serves as instruction for the operation of the M385 Universal Timer Counter M-module , the connection of measurement sources and the integration on an M-module carrier. Furthermore it gives the user additional information for special applications and configurations of the product.
Detailed information concerning the individual assemblies (data sheets etc.) are not part of this manual. This manual also contains a description of the provided example software.

1.3. Scope

The scope of this manual is the usage of the M385 Universal Timer Counter M-module and the APIS based example software written in ANSI-C.

1.4. Definitions, Acronyms and Abbreviations

AcQ	AcQuisition Technology bv
APIS	AcQ Platform Interface Software
CPLD	Complex Programmable Logic Device
ESD	Electronic Static Discharge
HLP	High-time. Low-time and period
SPI	Serial Peripheral Interface
UTC	Universal Timer Counter
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

1.5. Notes Concerning the Nomenclature

Hex numbers are marked with a leading “0x”-sign: for example: 0x20 or 0xff.

File names are represented in italic: *filename.txt*.

Code examples are printed in `courier`.

Active-low signals are represented by a trailing asterisks (i.e. IACK*).

1.6. Overview

In chapter two a short description of the M385 hardware can be found. The next chapter covers the installation and setup of the M-module as well as the connection of measurement sources. In chapter 4 the operation and usage of the M385 is described in detail. AcQ provides APIS based example software for the M385, the software is described in chapter 5. Finally this document contains an Annex containing the bibliography, component image, technical data and the document history.

2. Product Overview

2.1. Introduction

The M385 Universal Timer Counter M-module is used for automated test and measurement applications. The module has two physical input lines and several measurement functions. The analog front-end is software programmable with respect to trigger level, sensitivity and AC/DC coupling. The counter resolution is 5 nanoseconds.

2.2. Technical Overview

Below an overview of the functionality of the M385 is listed:

- The timer/counter has two signal inputs
- Programmable AC/DC coupling
- Input voltage range +/- 10 V
- Front-end galvanic isolated
- Programmable trigger level and sensitivity
- 32-bit counter
- 32-bit timer with 200 MHz time base

- Frequency measurement upto 250MHz
- Period measurement including average
- Time interval measurement
- Frequency ratio measurement
- Totalize gated by software measurement
- Totalize gated by hardware measurement
- Rise/Fall time measurement
- Positive, Negative Pulse Width measurement
- Minimum, Maximum AC/DC voltages

- A08D16 M-module interface
- INTA software-end-of interrupt
- Module identification EEPROM
- APIS based software support

3. INSTALLATION AND SETUP

3.1. UNPACKING THE HARDWARE

The hardware is shipped in an ESD protective container. Before unpacking the hardware, make sure that this takes place in an environment with controlled static electricity. The following recommendations should be followed:

- Make sure your body is discharged to the static voltage level on the floor, table and system chassis by wearing a conductive wrist-chain connected to a common reference point.
- If a conductive wrist-chain is not available, touch the surface where the board is to be put (like table, chassis etc.) before unpacking the board.
- Leave the board only on surfaces with controlled static characteristics, i.e. specially designed anti static table covers.
- If handling the board over to another person, touch this persons hand, wrist etc. to discharge any static potential.

IMPORTANT: Never put the hardware on top of the conductive plastic bag in which the hardware is shipped. The external surface of this bag is highly conductive and may cause rapid static discharge causing damage. (The internal surface of the bag is static dissipative.)

Inspect the hardware to verify that no mechanical damage appears to have occurred. Please report any discrepancies or damage to your distributor or to AcQuisition Technology immediately and do not install the hardware.

3.2. M-MODULE PERIPHERAL CONNECTIONS

This section describes the peripheral connections of the M385. Two possible options exist:
- Connection via the D-sub connector with coaxial contacts (recommended)
- Connection via the 24-pole female P2 header to the carrier board (not recommended)

The pin-assignments of the D-sub front connector with two coaxial contacts is shown in Figure 1.

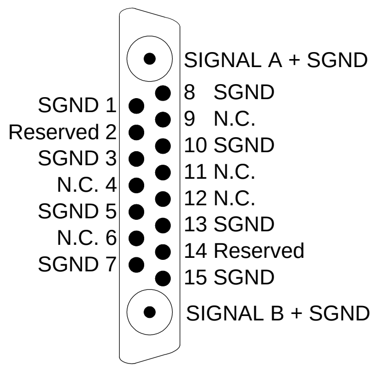

Figure 1 Front view of female D-sub connector with two coaxial contacts

To achieve the highest accuracy measurements, use the coaxial connections with appropriate cabling and shielding. A suitable mating connector from Deltron consist of the part numbers listed in table 1. Use an M-module carrier without 24-pin connectors.

Part number (Deltron)	Description
DTS17WPZ/2	Connector
DMS53740-1	Coaxial contacts
DMH25UN DE006486	Cover

Table 1 D-sub mating connector

The 24-pole female header on the M385 is meant to be used on M-module carriers with rear I/O. However because a lot of parasitic effects interfere with the input signals, the accuracy of the M385 decreases dramatically. The pin-assignments of the 24-pole female P2 header for connection to the carrier board is shown in Figure 2.

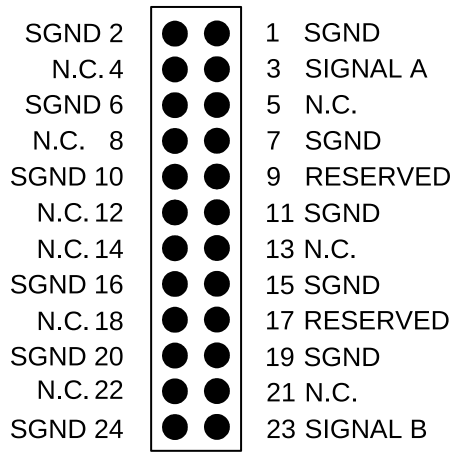

Figure 2 24-pole female P2 header

4. FUNCTIONAL DESCRIPTION

4.1. BLOCK DIAGRAM

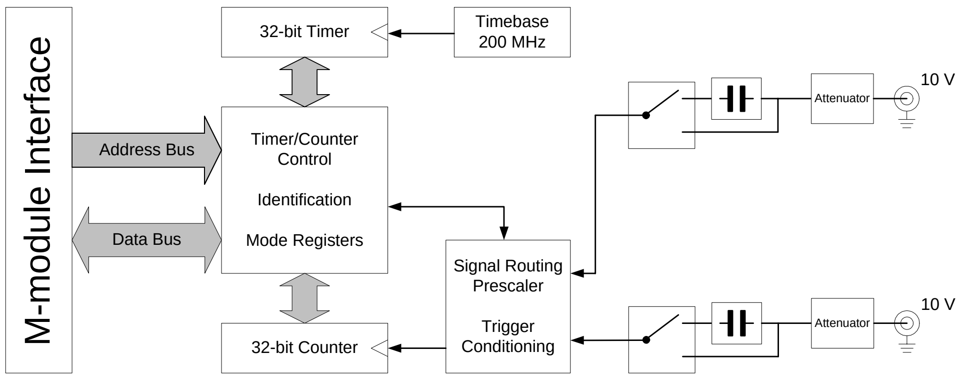

Figure 3 M385 Block Diagram

4.2. QUICK OVERVIEW

This section gives a quick overview of the address-map and on-card registers of the M385 Universal Timer Counter M-module

4.2.1. ADDRESS MAP OVERVIEW

The following table gives an overview of the address-map of the M385. The address-map as shown below gives an offset from the base address instead of an absolute address, since the absolute address is depending on the carrier on which the M385 is mounted
.

Offset	Width	Description
0x00	16-bit	Counter MSW
0x02	16-bit	Counter LSW
0x04	16-bit	Timer MSW
0x06	16-bit	Timer LSW
0x08	16-bit	Mode register
0x0a	16-bit	Control register
0x0c	16-bit	SPI register
0xfe	16-bit	EEPROM register

To avoid endianess problems it is best to access the on-card Control registers and the 32-bit timer/counter registers with 16-bit accesses at 16-bit address boundaries.

4.3. UNIVERSAL TIMER COUNTER

The Universal Timer Counter controller provides all the low-level hardware primitives needed to implement the timer counter functions. The controller is implemented in a CPLD using VHDL.

The Timer Counter controller has four registers to control the various timing and counting related functions. Each of these are described in the following paragraphs.

4.3.1. Counter Register

The Counter register is used to count input events. It can be used as an up or down counter depending on the measurement function selected. For example during frequency measurements the input transitions are counted during a specified time (count up), during period measurements the counter is used for averaging (count down)

This 32-bit read/write register is composed of two 16-bit registers.

COUNTERMSW Offset: 0x0
MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT31	CNT30	CNT29	CNT28	CNT27	CNT26	CNT25	CNT24	CNT23	CNT22	CNT21	CNT20	CNT19	CNT18	CNT17	CNT16
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Read / Write

CNT31..16 — Counter Bits
Most significant word (MSW) of up/down counter

COUNTERLSW Offset: 0x2
MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Read / Write

CNT15..0 — Counter Bits
Least significant word (LSW) of up/down counter

Note: Since the 32-bit counter register is implemented as two 16-bit registers using pipe-lining, the lowest number which can be used for averaging is 2 (write 0x00000001). To disable averaging use the single shot bit in the mode register

4.3.2. Timer Register

The Timer register is used to measure timing events. It can be used as an up or down counter depending on the measurement function selected. For example during frequency measurements the input transitions are counted during a specified time, the timer register is preloaded with the measurement time and the timer will be decremented until it reaches zero. During period measurements the timer will increment during one or more periods of the input signal.

This 32-bit read/write register is composed of two 16-bit registers.

TIMERMSW Offset: 0x4

MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR31	TMR30	TMR29	TMR28	TMR27	TMR26	TMR25	TMR24	TMR23	TMR22	TMR21	TMR20	TMR19	TMR18	TMR17	TMR16
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read / Write															

TMR31..16 — Timer Bits
Most significant word (MSW) of up/down timer

TIMERLSW Offset: 0x6

MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR15	TMR14	TMR13	TMR12	TMR11	TMR10	TMR9	TMR8	TMR7	TMR6	TMR5	TMR4	TMR3	TMR2	TMR1	TMR0
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read / Write															

TMR15..0 — Timer Bits
Least significant word (LSW) of up/down timer

4.3.3. MODE REGISTER

The mode register is used to select the measurement function and to configure the M385 for the desired input source and the polarity. The measurement function can be selected with bit 2-4. Bit 5 is only used in combination with the counter function gated by hardware, to select if the gate signal is active high or active low. The input source and polarity can be selected with bit 0-1.

MODE Offset: 0x8

MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SS	SCL	MF2	MF1	MF0	ISS1	ISS0
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read / Write															

RES[15:7] — Reserved

SS — Single Shot
This bit selects single shot function for High-time, Low-time and Phase measurement. If this bit is set, the value of the counter register is ignored.

SCL — Secondary Input Level
This bit is used to configure the Level or edge of the Secondary Input which is needed for Ratio and Phase measurements. These measurements need a signal on both inputs. When SCL = ‘0’ an active low level or falling edge is selected, when SCL = ‘1’ an active high level or rising edge is selected.

MF[2:0] — Measurement Function

These bits are used to select the measurement function according to the following table:

MF[2:0]	Description
000	Frequency measurement
010	High-time, low-time measurement
011	Period measurement
100	Ratio measurement
101	Counter function gated by software
110	Phase measurement
111	Counter function gated by hardware

ISS[1:0] — Input Signal Selection

These bits are used to select the input signal and edge of choice, to be used for the configured measurement function. The possible values are listed in the following table:

ISS[1:0]	Description
00	Input signal A rising edge
01	Input signal B rising edge
10	Input signal A falling edge
11	Input signal B falling edge

4.3.4. Control Register

The Control register is used to start a measurement and to check the status of the measurement and the current level of the input signals. Furthermore the Control register controls the interrupts of the M385.

The following register describes the Control Register.

CTRL Offset: 0x8

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										SIGB	SIGA	IRQS	IRQEN	START	DONE
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read / Write															

RES[15:6] — Reserved

SIGB — Signal B

This bit reflects the current state of input signal B after the prescaler.

SIGA — Signal A

This bit reflects the current state of input signal A after the prescaler.

IRQS — Interrupt Request Status
This bit reflects the state of the interrupt line. If this bit is set, an interrupt is pending. To clear the pending interrupt, write a ‘0’ to the “START” bit. An interrupt will be generated upon completion of a measurement function.

IRQEN — Interrupt Request Enable
Set this bit to enable interrupts on the M385. Clear this bit to disable interrupts.

START — Start
This bit must be set to start a measurement. Clear this bit when a measurement is completed, the done bit is then automatically cleared too. When interrupts are enabled, clear this bit to clear a pending interrupt.

DONE — Done
This bit reflects the state of a measurement when set, the measurement is completed. This bit is automatically cleared when the start bit is cleared.

4.4. SPI INTERFACE

The input signal conditioning with respect to coupling, trigger level, hysteresis and prescaler can be controlled using several DACs and a CPLD which implements the prescaler and the coupling control. These devices are controlled using a 3-wire Serial Peripheral Interface (SPI™). The input configuration SPI chain is graphically depicted in Figure 4.

The M385 has a second SPI chain, which controls the DAC used to tune the time base. This chain is graphically depicted in Figure 5.

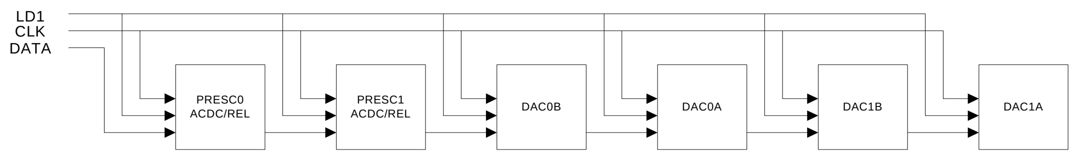

Figure 4 Input configuration SPI stream

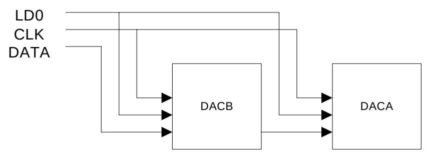

Figure 5 VCTXO oscillator configuration SPI stream

The register description of the prescaler stream and input configuration control for input signal A (PRESC0) and input signal B (PRESC1) is described as follows:

PRESC0/PRESC1

	LSB				MSB	
	0	1	2	3	4	
Din ->	PRES0	PRES1	PRES2	ACDC	REL	Dout ->
	Reset:					
	0	0	0	0	0	

Write Only

PRES[2:0] — input prescaler

These bits are used to configure the prescaler for each input signal. The maximum frequency output after the prescaler must not exceed 50MHz. The division factor is programmed according to the following table:

Prescaler Value	Division Factor
000	divide by 1
001	divide by 2
010	divide by 4
011	divide by 8
100	divide by 16

ACDC — input coupling

This bit is used to configure the input coupling either AC or DC. If this bit is set to ‘1’ DC is selected. If this bit is cleared to ‘0’ AC is selected.

REL — relay control

This bit is used to control the input relay. At this time it is not used and must written with ‘0’.

The hysteresis voltage and the trigger level can be configured individually for both input signal A and input signal B, using 12-bit D/A converters The function for each DAC is described in the following table:

DAC Channel	Function
DAC0B	Hysteresis input signal A
DAC0A	Trigger level input signal A
DAC1B	Hysteresis input signal B
DAC1A	Trigger level input signal B

The type DACs used are LTC1446, 12-Bit Rail-to-Rail Micropower DACs from Linear Technology Corporation. Detailed information can be found in the data sheet.

DAC
LSB MSB

	0	1	2	3	4	5	6	7	8	9	10	11	
Din ->	DAC0	DAC1	DAC2	DAC3	DAC4	DAC5	DAC6	DAC7	DAC8	DAC9	DAC10	DAC11	Dout ->
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	

Write Only

DAC[11:0] — Dac value
These 12 bits are used to set the dac output level.

4.4.1. SPI Control Register

On the M385, the SPI interface is controlled by the SPI Control register using bit-banging. Access to the SPI chains takes place through the following register:

SPI Offset: 0x0C
MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												LD0	LD1	CLK	DATA
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset: (values above)

Read / Write

RES[15:4] — Reserved

LD0 — Load 0
This bit corresponds to the Load or Chip select of SPI chain 0 which is used to tune the time base.

LD1 — Load 1
This bit corresponds to the Load or Chip select of SPI chain 1 which is used to configure the input stages.

CLK — Clock
This bit corresponds to the Clock input of both SPI chains.

DATA — Data
This bit corresponds to the Data input of both SPI chains.

4.5. EEPROM Register

The identification of an M-module is implemented using a serial EEPROM with a 64*16 word organization. The industry-standard component 93C46 is used in order to make the identification compatible throughout the complete range of available modules. Access to the identification EEPROM takes place through the following register:

EEPR Offset: 0xFE

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													EEPC	EEPCK	EEPDI/O
Reset:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Read / Write

RES[15:3] — Reserved

EEPCS — Chip-Select
: This bit corresponds to the chip select input of the EEPROM.

EEPCK — Clock
: This bit corresponds to the clock input of the EEPROM.

EEPDI/O — Data input/output
: This bit corresponds to the data input of the EEPROM when writing, and data output of the EEPROM when reading.

For information on controlling the EEPROM refer to the NM93C46 data sheets. A software example can be found in the file *eeplib.c* that is part of the software distribution. For information on the memory organization refer to the M-module Specification.

4.6. INPUT SIGNAL CONDITIONING

The two input circuits of the M385 include several programmable input signal conditioning controls.These controls are all programmable through the SPI interface.

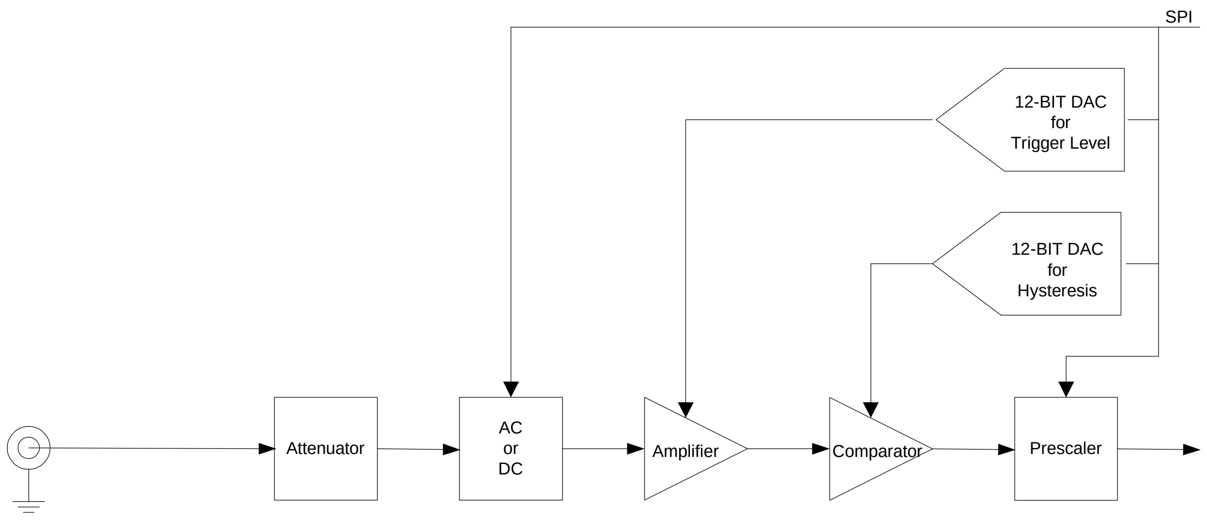

Figure 6 Input Signal Conditioning

The input characteristics described in the following paragraphs are:

- Range
- Sensitivity or Hysteresis
- Ac-Dc Coupling
- Trigger Level
- Prescaler

4.6.1. RANGE

The range defines the frequency range over which the input amplifier sensitivity is specified. The range varies with the selected measurement function. Consult the individual Signal Operating range and/or Dynamic range specifications in paragraph "Operating Mode Specifications" on page 53

4.6.2. SENSITIVITY OR HYSTERESIS

Sensitivity is the lowest amplitude signal at a particular frequency that the M385 can measure. The amplifier gain and the voltage difference between the input trigger levels set at a value equal to the midpoint of the input signal. The input waveform must cross both upper and lower hysteresis levels to generate a count, as shown in Figure 7.

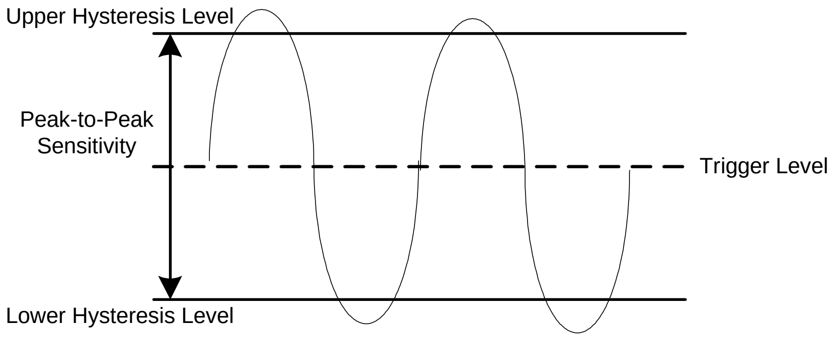

Figure 7 Acceptable Peak-to-Peak Amplitude

On the M385 the hysteresis (sensitivity) of the counter can be controlled by programming DAC0A for input signal A and DAC1A for input signal B (See “SPI Interface” on page 15.)

The relation between the DAC value and hysteresis window voltage is shown in the following table:

DAC value	Hysteresis
0x740	10 mV
0x7f0	15 mV
0x880	20 mV
0x980	25 mV
0xa00	30 mV
0xaf0	35 mV
0xb80	40 mV
0xbf0	45 mV
0xc80	50 mV
0xd80	55 mV
0xe00	60 mV
0xfff	> 60 mV

Note: At minimum sensitivity, the hysteresis window is increased requiring a larger peak-to-peak voltage to generate a count. Optimum sensitivity depends on measurement application and other factors such as noise or interfering signals.

4.6.3. AC-DC COUPLING

Selectable Ac or Dc coupling is provided for both input signals. Ac coupling must be used for signals with a dc offset. Figure 8 demonstrates the use of Ac coupling.

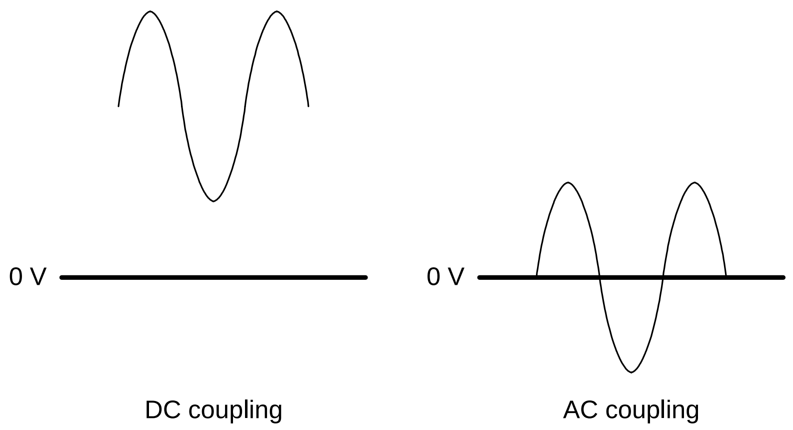

Figure 8 Ac-Dc Coupling

Note: An input signal with DC content would not be counted unless AC coupling, was used to remove the DC offset, or the appropriate trigger level was used.

Note: Switching from DC coupling to AC coupling takes about 300 milliseconds before the signal is stable enough for correct measurements.

4.6.4. Trigger Level

Trigger level is the voltage at the center of the hysteresis window. The actual trigger points are typically at the upper hysteresis level (positive slope) and the lower hysteresis level (negative slope), as shown in Figure 9.

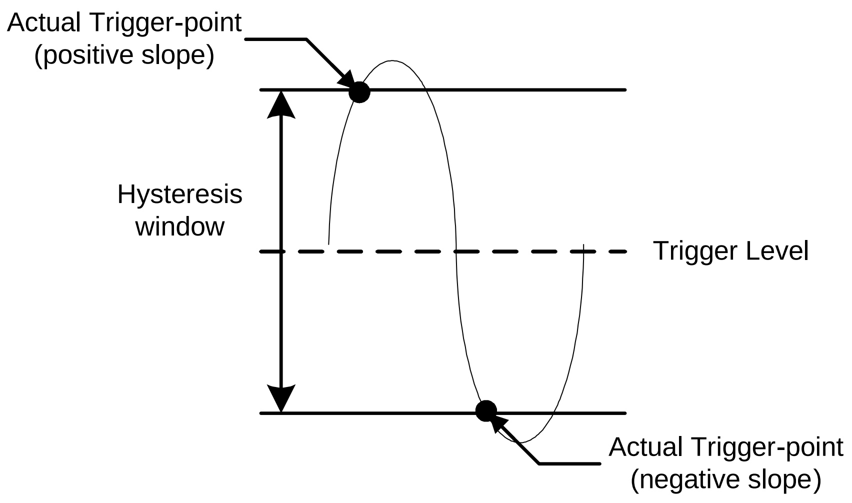
Actual Trigger-point (positive slope)
Hysteresis window
Trigger Level
Actual Trigger-point (negative slope)

Figure 9 Trigger Level and Actual Trigger Point

On the M385 the trigger level of the counter can be controlled by programming DAC0B for input signal A and DAC1B for input signal B (See “SPI Interface” on page 15.)

The trigger levels are adjustable over the dynamic range (-10V to +10V). The relation between the DAC value and trigger level voltage can be calculated using the following formula:

$$N_{DAC} = \left(\frac{V_{trigger}}{Attn} + 5\right)\left(\frac{4095}{10}\right)$$

Where $V_{trigger}$ is the desired trigger level, *Attn* is the input attenuation which is 2 and the N_{DAC} is the desired 12-bit DAC value.

4.6.5. PRESCALER

The analog front-end of the M385 is galvanically isolated from the digital part of the board, the galvanic isolation is able to pass digital counter signals with a frequency up to 50MHz. To be able to perform frequency and time related measurements on signals with a frequency higher than 50MHz, a programmable prescaler is implemented which acts as a frequency divider. The prescaler is located between the output of the comparator stages and the input of the isolation circuitry as shown in figure 6. Each input channel (A and B) has its own prescaler. The prescalers are individually software programmable to divide the signal frequency by 1, 2, 4, 8 or 16. For details on programming the prescaler please refer to section 4.4.
When a prescaler is programmed to a value other than 1, the duty cycle information of the scaled signal will be lost therefore scaling must only be used in situations where the duty cycle and phase of a signal is not important, for instance when measuring frequency, period, ratio, totalize etc. When the duty cycle or phase information does matter the prescaler must be set to 1 for instance when measuring pulse-width, rise- or falltime or time intervals.
When the prescaler is used the measurement result must be processed accordingly. The provided example software shows how the prescaler must be used.

4.7. Hardware Measurement Functions

The following sections give an overview of the basic (implemented in hardware) measurement functions provided by the M385.

4.7.1. Frequency measurement

Description Frequency measurements are made on repetitive input signals. For a specified period of time the input edges are counted through the counter register. The measurement time or gate time is determined through the timer register. The timer register must be preloaded with the required gate time and will count down to zero using the 200MHz time base.

At the start of the measurement the gate time will be synchronized with the first edge of the input signal and from there on start to count down. Once the timer register reaches zero, the counter register will stop counting the input signal edges. The timer register will however continue to count down (it will roll over) until the next edge of the input signal. This will signal the end of the measurement. The remainder captured in the timer register can be used to determine the input frequency with even more accuracy.

The frequency can be derived from:

$$f_n = \frac{N}{t_{gate} + t_{remainder}}$$

N is the value derived from the counter register. The gate time t_{gate} is the programmed value in the timer register x 5ns (200MHz time base) before the measurement starts and $t_{remainder}$ is the two's complement value from the timer register after the measurement ends x 5ns.

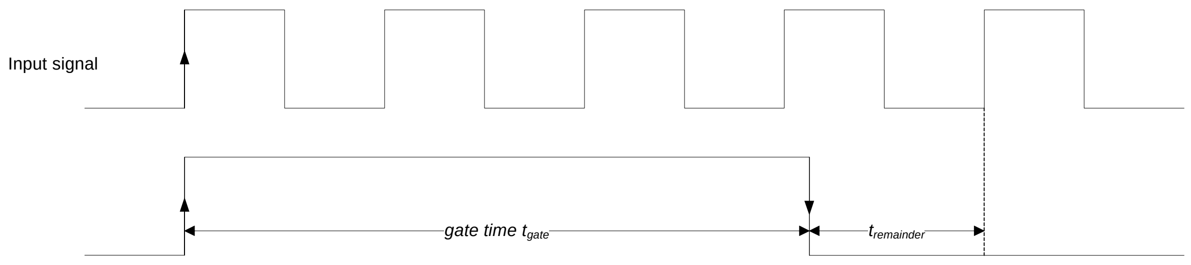

Figure 10 Frequency Measurement

Range The frequency range and resolution is:
Range: 0.05 Hz to 250 MHz
LSD: (5 ns / Gate Time) * FREQ

Signal Periodic signal 0.05Hz < frequency <= 250MHz (using prescaler).

Procedure Configure the input and prescaler (using SPI devices).

Clear the counter register

```
*(UINT32 *)(COUNTER) = 0x00000000;
```

Select the Frequency function for the mode register

```
*(UINT16 *)(MODE) = 0x0000;
```

Write the desired gate time to the timer register e.g. 10ms

```
t_gate = 10E-3 / 5E-9;
*(UINT32 *)(TIMER) = t_gate;
```

Start the measurement and wait for completion

```
*(UINT16 *)(CTRL) = 0x0002;
while !(*(UINT16 *)(CTRL) & 0x0001);
```

Read result and calculate the frequency.

```
N = *(UINT32 *)(COUNTER);
t_remainder = (0 - (INT32)(TIMER)) * 5E-9;
F = prescaler * N / (t_gate + t_remainder);
```

4.7.2. PERIOD MEASUREMENT

Description Period measurements are made on repetitive input signals. For a specified number of periods, the time is measured. The number of periods must be programmed in the counter register and the timer register must be cleared.

The first rising edge of the input signal will start the timer which will count up. The counter is decremented to zero upon each rising edge of the input signal. Once the counter reaches zero, the timer register will stop and the end of the measurement is signaled.

The period can be derived from:

$$t_{period} = \frac{t_{measurement}}{N_{average}}$$

$t_{measurement}$ is the value derived from the timer register x 5ns. $N_{average}$ is the number of periods programmed to the counter register before the measurement starts.

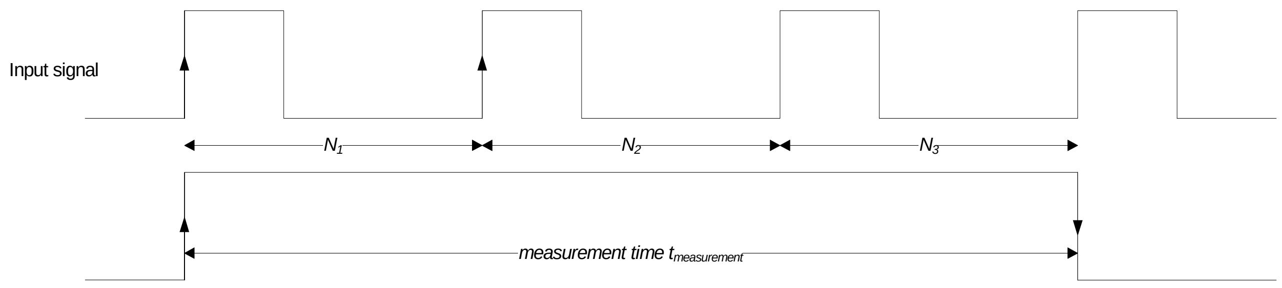

Figure 11 Period Measurement

Range The period range and resolution is:
Range: 5 ns to 20 s
LSD: (5 ns / Gate Time) * PER

Signal Periodic signal 0.05Hz < frequency <= 250MHz (using prescaler).

Procedure Configure the input and prescaler (using SPI devices).

Clear the timer register

```
*(UINT32 *)(TIMER) = 0x00000000;
```

Select the Period function for the mode register

```
*(UINT16 *)(MODE) = 0x000c;
```

Write the desired number of periods to the counter register e.g. 10000

```
N = 10000;
*(UINT32 *)(COUNTER) = N;
```

Start the measurement and wait for completion

```
*(UINT16 *)(CTRL) = 0x0002;
while !(*(UINT16 *)(CTRL) & 0x0001);
```

Read result and calculate the period.

```
t_measurement = *(UINT32 *)(TIMER) * 5E-9;
t_period = t_measurement / (prescaler * N);
```

4.7.3. Positive and Negative Pulse Width

Description Pulse width measurements are made on repetitive input signals. For a positive pulse width measurement, the time is measured between a rising edge and a falling edge of the input signal (either A or B). For a negative pulse width measurement, the time is measured between a falling edge and a rising edge of the input signal.

Pulse width measurements are not dependent on gate time. Greater resolution can be obtained by increasing the average count on which the measurement takes place.

At the start of the measurement, the counter register must be preloaded with the average count and the timer register must be cleared. At every rising edge of the input signal (for a positive pulse width measurement) the timer is enable and starts counting up using the 200MHz time base. The counter is decremented by one. At every falling edge the timer is disabled and halts counting up.

The pulse width can be derived from:

$$t_{pulse} = \frac{t_{gate}}{N_{average}}$$

t_{gate} is the total measurement time value derived from the timer register x 5ns (200MHz time base). $N_{average}$ is average count which has been programmed to the counter register before the measurement starts.

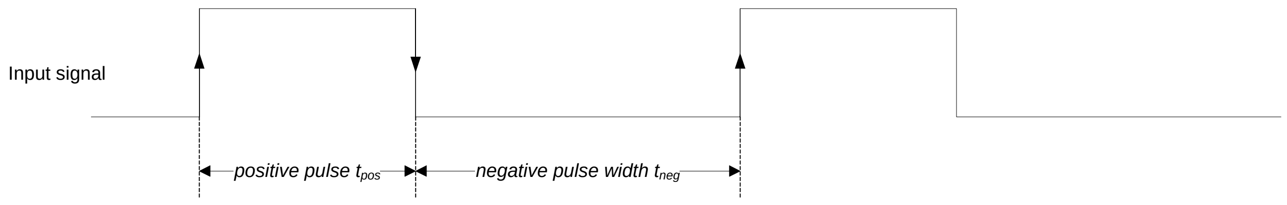

Figure 12 Positive and Negative Pulse Width measurements

Range The positive and negative pulse width range and resolution is:
Range: 20 ns to 20 s
LSD: (5 ns / Gate Time) * PULSE WIDTH

Signal Periodic signal 0.05Hz < frequency < 50MHz.

Procedure Configure the input and set the prescaler to 1x (using SPI devices).

Clear the timer register

```
*(UINT32 *)(TIMER) = 0x00000000;
```

Select the pulse width (high time, low time) function for the mode register

```
*(UINT16 *)(MODE) = 0x0008;
```

Write the desired number of periods to the counter register e.g. 10000

```
N = 10000;
*(UINT32 *)(COUNTER) = N;
```

Start the measurement and wait for completion

```
*(UINT16 *)(CTRL) = 0x0002;
while !(*(UINT16 *)(CTRL) & 0x0001);
```

Read result and calculate the pulse width.

```
t_gate = *(UINT32 *)(TIMER) * 5E-9;
t_pulse = N / t_gate;
```

4.7.4. TIME INTERVAL

Description The time Interval measurement measures the length of time between programmable edges of input signal A and input signal B. The type of edge (rising or falling) of both signals can be configured individually. The time interval measurements can be made with single-shot or averaging. The number of periods must be programmed in the counter register (or the single shot bit in the mode register must be used) and the timer register must be cleared

The first programmed edge of input signal A will start the timer which will count up and decrement the counter. The timer will be stopped at the first programmed edge of input signal B. If averaging is to be used, the next programmed edge of input signal A will continue the timer, and so on, until the counter reaches zero which will signal the end of the measurement.

The time interval can be derived from:

$$t_{interval} = \frac{t_{measurement}}{N_{average}}$$

$t_{measurement}$ is the value derived from the timer register x 5ns. $N_{average}$ is the number of periods programmed to the counter register before the measurement starts.

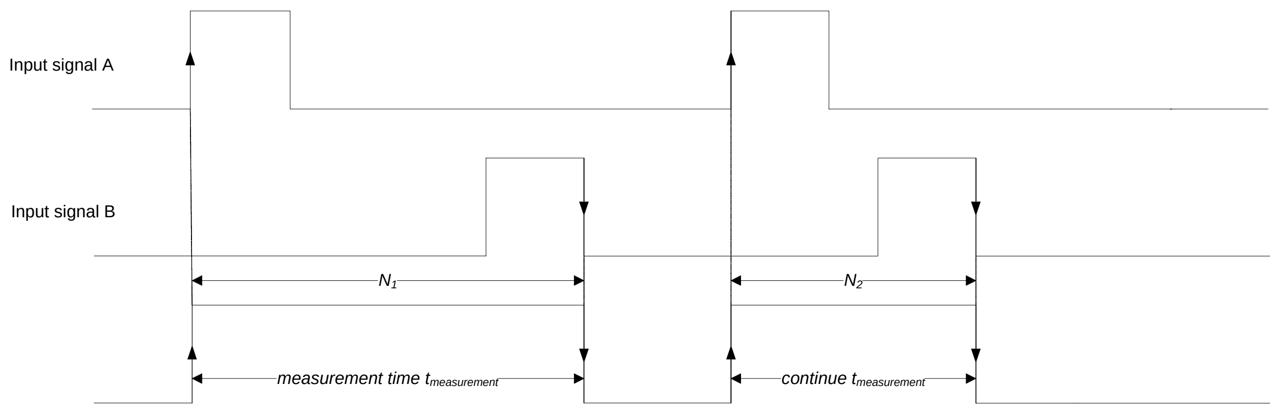
Input signal A
Input signal B
N_1
N_2
measurement time $t_{measurement}$
continue $t_{measurement}$

Figure 13 Time Interval Measurement

Range The time interval range and resolution is
Range: 20 ns to 20 s (single shot)
LSD: 5 ns

Signal Periodic signal 0.05Hz < frequency < 50MHz.

Procedure Configure the inputs and set the prescalers to 1x (using SPI devices).

Clear the timer register

```
*(UINT32 *)(TIMER) = 0x00000000;
```

Select the Phase function for the mode register

```
*(UINT16 *)(MODE) = 0x0018;
```

Write the single shot bit to measure one interval

```
*(UINT16 *)(MODE) |= 0x0040;
```

Start the measurement and wait for completion

```
*(UINT16 *)(CTRL) = 0x0002;
while !(*(UINT16 *)(CTRL) & 0x0001);
```

Read result and calculate the interval time.

```
t_measurement = *(UINT32 *)(TIMER) * 5E-9;
t_interval = t_measurement / 1;
```

4.7.5. Frequency Ratio Measurement

Description The ratio measurement function provides measurement of the ratio between two frequencies. The frequency ratio is measured on input signal A in relation to input signal B or input signal B in relation to input signal A.

The frequency ratio is measured for a specified number of gate periods using the input signal B as a gate signal while counting input edges of input signal A. The counter register must be preloaded with the required number of gate periods and will count down to zero using the input signal B. The timer register must be cleared at start and is used to count edges of input signal A during the time that the gate signal is valid.

The frequency ratio can be derived from:

$$f_{ratio} = \frac{N}{G}$$

N is the value derived from the timer register, G is the value which has been programmed to the counter register before the measurement has started.

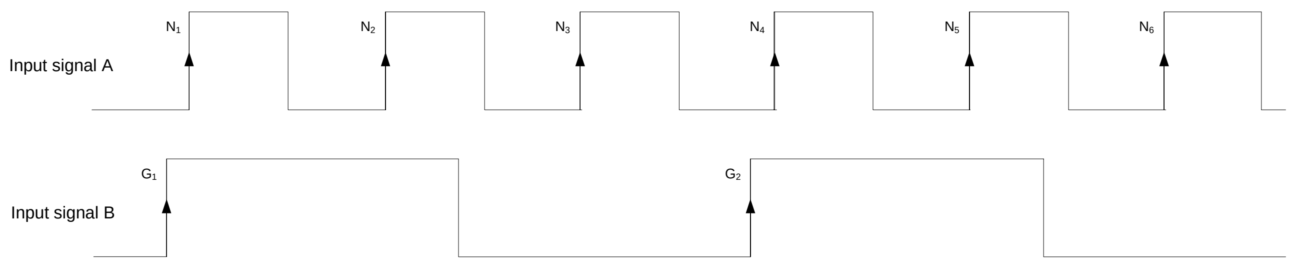

Figure 14 Frequency Ratio Measurement

Range The frequency ratio range is
Range: 0.4 E-9 to 2.5 E+9

Signal Periodic signal 0.05Hz < frequency <= 250MHz (using prescaler).

Procedure Configure the inputs and prescalers (using SPI devices), both prescalers must be set to the same division factor.

Clear the timer register

```
*(UINT32 *)(TIMER) = 0x00000000;
```

Select the Ratio function for the mode register

```
*(UINT16 *)(MODE) = 0x0010;
```

Write the desired number of gate periods to the counter register e.g. 10000

```
N = 10000;
*(UINT32 *)(COUNTER) = N;
```

Start the measurement and wait for completion

```
*(UINT16 *)(CTRL) = 0x0002;
while !(*(UINT16 *)(CTRL) & 0x0001);
```

Read result and calculate the frequency ratio.

```
t_measurement = *(UINT32 *)(TIMER);
t_ratio = t_measurement / N;
```

Note: The information above assumes that the higher frequency is connected to input A and the lower frequency is connected to input B, vice versa is also allow but keep in mind that the number of gate periods must be enough to capture the low frequency signal on input A. In general it is recommended to connect the high frequency to input A and the low frequency to input B, in this case the number of gate periods is less critical, although a high number of gate periods in relation to the frequency ratio might result in a long measurement delay.

4.7.6. Totalize Measurements Gated by Software

Description The totalize measurement gated by software measures the number of counts (events) received through signal input A or input B. The count is accumulated from input cycle to input cycle. The counter will start and stop counting input events using software control.

Totalize measurements are independent of the gate time. However the timer will run for as long as the totalize function is counting.

The totalize count can be derived from the counter register.

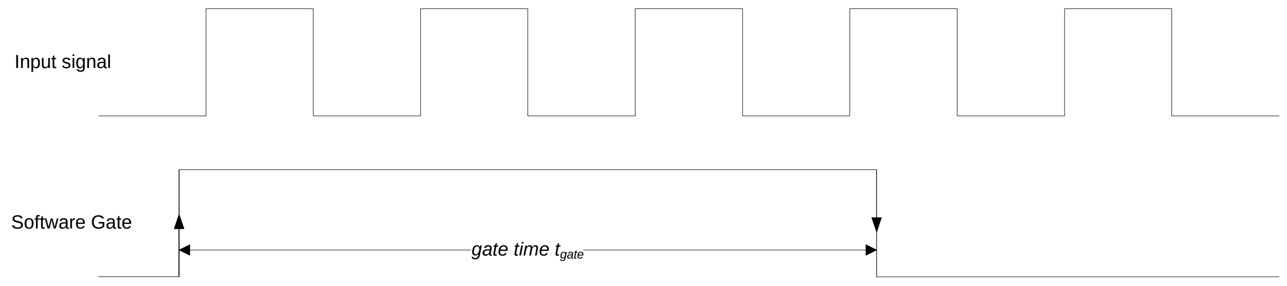

Figure 15 Totalize Measurement Gated by Software

Range The totalize range and resolution is
Range: 0 to 1 x 10^{32} -1
LSD: 1 count of input signal

Signal Periodic signal 0.05Hz < frequency <= 250MHz (using prescaler).

Procedure Configure the inputs and prescalers (using SPI devices).

Clear the timer and the counter register

```
*(UINT32 *)(TIMER) = 0x00000000;
*(UINT32 *)(COUNTER) = 0x00000000;
```

Select the Totalize function (counter gated by software) for the mode register

```
*(UINT16 *)(MODE) = 0x0014;
```

Start the measurement

```
*(UINT16 *)(CTRL) = 0x0002;
```

After a while stop the measurement

```
*(UINT16 *)(CTRL) = 0x0000;
```

Read result.

```
c_totalize = *(UINT32 *)(COUNTER) * prescaler;
```

4.7.7. Totalize Measurements Gated by Hardware

Description The totalize measurement gated by hardware measures the number of counts (events) received through input signal A while the gate (input signal B) is open. The count is accumulated from input cycle to input cycle. The timer will start and stop counting input events using the gate signal on input signal B control.

The totalize measurements can be made with single-shot or averaging. The number of periods must be programmed in the counter register (or the single shot bit in the mode register must be used) and the timer register must be cleared

The totalize count can be derived from the timer register.

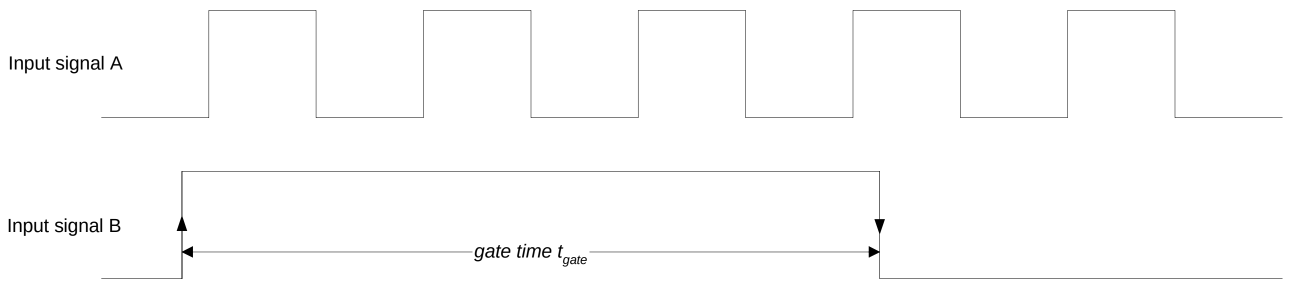

Figure 16 Totalize Measurement Gated by Hardware

Range The totalize range and resolution is
Range: 0 to 1 x 10^{32} -1
LSD: 1 count of input signal

Signal Periodic signal 0.05Hz < frequency <= 250MHz (using prescaler).

Procedure Configure the inputs and prescalers (using SPI devices).

Clear the timer and the counter register

```
*(UINT32 *)(TIMER) = 0x00000000;
```

Select the Totalize function (counter gated by hardware) for the mode register

```
*(UINT16 *)(MODE) = 0x001c;
```

Write the desired number of gate periods to the counter register e.g. 10000

```
N = 10000;
*(UINT32 *)(COUNTER) = N;
```

Start the measurement and wait for completion

```
*(UINT16 *)(CTRL) = 0x0002;
while !(*(UINT16 *)(CTRL) & 0x0001);
```

Read result.

```
c_totalize = *(UINT32 *)(TIMER) * prescaler;
```

5. M385 LIBRARY SOFTWARE

This chapter describes the example software which is available for the M385 Universal Timer Counter M-module . The example software consists of a measurement function library and an example application. LabView and LabWindows/CVI software support is available as a separate package.
The example software is distributed as ANSI-C source code and consists mainly of an M385 function library which provides functions for easy access to the M385 and a demo program which illustrates the usage of the software library.

The M385 library functions are APIS based, physical accesses and interrupt support are handled by APIS, AcQ Platform Interface Software. The next section contains general information on APIS, for detailed information please refer to the APIS Programmer's Manual.

5.1. APIS SUPPORT

AcQ produces and supports a large number of standard M-modules varying from networking and process I/O to motion control applications. Physically, the M-modules are supported by a large number of hardware platforms: VMEbus, PCI, CompactPCI as well as a wide variety of operating systems: OS-9, Windows NT, Linux etc.

APIS offers a way to program platform independent applications, example- and test software for controlling hardware. Application software written for APIS only needs re-compiling for a particular platform and is operational with little effort (provided that the application is operating system independent).

5.1.1. CONCEPT

Hardware accesses to registers and memory are handled by APIS. Some minor operating system dependent functions frequently used in hardware related software, such as interrupt handling and a delay function are also provided by APIS.

APIS platform support consists of an Application Programming Interface in the form of definition files coded in ANSI-C and platform dependent modules e.g. source files, libraries and/or drivers. In the most simple outline, a platform dependent APIS module consist of nothing more then macro definitions in which APIS calls are substituted by direct hardware accesses. But in most cases an APIS module will consist of a library with interface routines and in some implementations a device driver is needed for interaction with the operating system.

5.1.2. API

The Application Programming Interface for APIS is implemented in two ANSI-C coded definition files: *apis.h* which contains general definitions and *platform_apis.h* which contains platform specific definitions and references to the APIS function calls.

The application source file must include the APIS header file *apis.h*. Porting of the application to a platform, consists of re-compiling the source code with a defined pre-processor macro for selection of the used platform. The APIS header file contains generic APIS definitions and includes a platform specific header file according to the platform selection macro.

APIS calls are translated to the platform specific calls in the APIS header file and the platform specific definition file *platform_apis.h* (*platform* is a name that identifies a hardware and operating system combination, e.g i4000os9).

The macro PLATFORM must be defined, either via a pre-processor definition provided at compile time or via a macro-definition in the application source.

5.1.3. Code Generation

APIS based example software is available in ANSI-C source code. Source code files must be compiled with pre-processor definition PLATFORM set to a valid value, conforming the target platform. Building of the example software for the M385 is platform dependent, for details refer to release notes of the APIS support package of the target platform and the APIS Programmer's Manual.

Example of APIS supported platforms are i4000os9, i2000dos, i3000win etc.

5.2. External Interface Description

This section describes the external interface of the M385 Software Library.

5.2.1. Constant Definitions

This paragraph contains an overview of macro definitions which are relevant for the external interface.

Name	Value	Description
TRUE	1	Boolean true
FALSE	0	Boolean false
DEF_IVECTOR	0	Default interrupt vector
DEF_ILEVEL	0	Default interrupt level
ATTS	10	Timespan [ms] used for Auto trigger software function
AUTO_LEVEL	888.0	Use this level for automatic triggering

Error definitions in addition to standard APIS error codes:

Name	Value	Description
NOERR	0x0000	No error
ERR_TIMEOUT	0x8001	Time out
ERR_UNKFUNC	0x8002	Unknown function
ERR_INVINP	0x8003	Invalid input
ERR_DNRDY	0x8004	Data not ready
ERR_MODID	0x8005	Module not a M385
ERR_IDEEP	0x8006	Error reading ID eeprom
ERR_CALIB	0x8007	M385 not calibrated
ERR_MEM	0x8008	System memory error
ERR_SPI_DATA	0x8009	SPI data error
ERR_PRANGE	0x800a	Parameter out of range

Name	Value	Description
ERR_NOTSUP	0x800b	Not supported
ERR_OUTOFRNG	0x800c	Out of range
ERR_TRGLEVEL	0x800d	Trigger level out of range
ERR_NOTCAL	0x800e	Module not calibrated
ERR_IVIDEEP	0x800f	Contents of ID eeprom invalid
ERR_GAIN	0x8010	Gain error
ERR_OFFSET	0x8011	Offset error
ERR_NOTRG	0x8012	No trigger point found
ERR_ILLPRESC	0x8013	Prescaler value illegal for this measurement
ERR_VCTCXO	0x8014	VCTCXO not tuned
ERR_NOFUNC	0x8015	No function specified
ERR_ABORT	0x8016	Measurement aborted by the user

5.2.2. Type Definitions and Structures

The table below contains a list of all types and structures used in the M385 example software which are relevant to the external interface (standard ANSI-C types and types used exclusively by the library are not listed).

Name	Type	Description
UINT8	unsigned char	8-bit unsigned data
UINT16	unsigned short	16-bit unsigned signed data
UINT32	unsigned long	32-bit unsigned signed data
INT16	signed short	16-bit signed data
M385_INPUT	enum INP_A INP_B	Signal input (A and B)
INP_COUPLING	enum COUPLING_AC COUPLING_DC	Input coupling DC, pass input signal AC, subtract DC component
INP_PRESCALER	enum PRESCALER_1X PRESCALER_2X PRESCALER_4X PRESCALER_8X PRESCALER_16X PRESCALER_AUTO	Signal prescaler for digitized signal, 1x, 2x, 4x, 8x, 16x or AUTO.

Name	Type	Description
M385_INP_COND	struct float level HYS_VALS hysteresis INP_COUPLING coupling int reserved INP_PRESCALER prescaler	Input conditioner setting Trigger level Hysteresis [mV](10,15,20..65) AC/DC coupling reserved Frequency divider setting
MEASFUNC	enum NOMEAS MEASURE_FREQUENCY MEASURE_PERIOD MEASURE_TIMEINTERVAL MEASURE_RATIO MEASURE_TOTALIZE_SW MEASURE_TOTALIZE_HW MEASURE_RISE_FALL_TIME MEASURE_PULSE_WIDTH MEASURE_VOLTAGE	 Perform no measurement Frequency measurement Period measurement Time interval measurement Ratio measurement Totalize measurement sw gated Totalize measurement hw gated Rise/Fall time measurement Pulse width measurement Voltage measurement
PWTYPE	enum POSITIVE_PW NEGATIVE_PW	Type of pulse width measurement, either Positive or Negative pulse width
VTYPE	enum VMIN VMAX VDC VAC	Type of voltage measurement, Minimum value, Maximum value, DC level of a sine (offset), AC value of a sine (peak)
ETYPE	enum RISING FALLING	Type of edge for event counting, either Rising edge or Falling edge
RFTYPE	enum RTIM FTIM	Type of rise/fall time measurement, either rise time or fall time
RATIOTYPE	enum ATOB BTOA	Type of ratio measurement A over B or B over A
GTYPE	enum AHIGH ALOW	Gate signal type, active high or active low
RAW_FORMAT	struct UINT32 cnt_val UINT32 tmr_val	Raw data format consisting of the counter value and the timer value
LTYPE	enum AUTO_SIG_LEV TTL CMOS ECL	Signal levels for Rise/Fall time measurement
FREQ_FORMAT	double	Frequency data format [Hz]
PRD_FORMAT	double	Period data format [s]

Name	Type	Description
PW_FORMAT	double	Pulse width data format [s]
VOLT_FORMAT	float	Voltage data format [V]
TOTALIZE_FORMAT	UINT32	Counter value data format
RATIO_FORMAT	double	Frequency ratio format
RISE_FALL_TIME	double	Rise/fall time data format [s]
INTERVAL_FORMAT	double	Interval data format [s]
M385_READ_ FORMAT	enum RAW_FORMAT raw FREQ_FORMAT frequency PRD_FORMAT period PW_FORMAT pulsewidth VOLT_FORMAT voltage TOTALIZE_FORMAT counter RISE_FALL_TIME risetime RISE_FALL_TIME falltime INTERVAL_FORMAT interval RATIO_FORMAT ratio	Data format for m385_read() function.

Name	Type	Description
M385_FUNC_T	struct	M385 Function Structure
	MEASFUNC function	Measurement function
	M385_INP_COND input_a	Input conditioning ch A
	M385_INP_COND input_b	Input conditioning ch B
	UINT8 read_blk	Read blocked ?
	UINT8 irq_enable	Interrupt enable ?
	UINT32 timeout	Timeout
	UINT8 raw_data	Raw data results ?
	struct func_par	Measurement parameters
	struct freq	MEASURE_FREQUENCY
	M385_INPUT input	Input (A or B)
	long gate_tm	Gate time (msecs)
	struct period	MEASURE_PERIOD
	M385_INPUT input	Input (A or B)
	UINT32 nr_msr	Number of measurements
	struct pw	MEASURE_PULSE_WIDTH
	M385_INPUT input	Input (A or B)
	PWTYPE msr_type	Pulse width measurement type
	UINT32 nr_msr	Number of measurements
	struct voltage	MEASURE_VOLTAGE
	M385_INPUT input	Input (A or B)
	VTYPE msr_type	Measurement type
	UINT32 timespan	Scan during this timespan
	struct totalize_sw	MEASURE_TOTALIZE_SW
	M385_INPUT input	Input (A or B)
	ETYPE edge	Edge to count on
	struct totalize_hw	MEASURE_TOTALIZE_HW
	ETYPE edge	Input A: edge to count on
	GTYPE gate	Input B: type of gate signal
	UINT32 nr_msr	Number of measurements
	struct interval	MEASURE_TIMEINTERVAL
	ETYPE edge_inpa	Edge for input A
	ETYPE edge_inpb	Edge for input B
	UINT32 nr_msr	Number of measurements
	struct rftime	MEASURE_RISE_FALL_TIME
	RFTTYPE msr_type	Measurement type RTIM, FTIM
	UINT32 nr_msr	Number of measurements
	struct ratio	MEASURE_RATIO
	UINT32 nr_msr	Number of measurements

5.2.3. Function Reference

m385_open () — Open M385 device

Syntax:	`int m385_open(APIS_PATH path_id, M385_DEV *device)`
Description:	Open and initialize a M385 timer counter device indicated by it's APIS path ID. The caller must provide an initialized pointer to a allocated device structure with the size of M385_DEV.
Arguments:	APIS_PATH path APIS path of M385 module. M385_DEV device Pointer to M385 device structure.
Returns:	NOERR or an appropriate error code.

m385_close () — Close M385 device

Syntax:	`int m385_close(M385_DEV *device)`
Description:	Clear control register. When interrupts are installed remove interrupt handler. Close hardware path to M385.
Arguments:	M385_DEV *dev Pointer to M385 device structure.
Returns:	NOERR or an appropriate error code.

m385_ioctl () — Configure and start a measurement

Syntax:	`int m385_ioctl(M385_DEV *dev, M385_FUNC_T *pFunc)`
Description:	Configure and start a measurement function on the M385. The result of the measurement can be read with the m385_read() function. Supported Measurement functions: MEASURE_FREQUENCY: Frequency measurement MEASURE_PERIOD: Period measurement MEASURE_TIMEINTERVAL: Time interval measurement MEASURE_RATIO: Ratio measurement MEASURE_TOTALIZE_SW: Totalize measurement, software gated MEASURE_TOTALIZE_HW: Totalize measurement, hardware gated MEASURE_RISE_FALL_TIME: Rise/Fall time measurement MEASURE_PULSE_WIDTH: Pulse width measurement MEASURE_VOLTAGE: Voltage measurement
Arguments:	M385_DEV *device pointer to M385 device M385_FUNC_T *pFunc pointer to M385 function struct. This struct contains all information that is necessary to configure the M385 for a measurement and must be filled by the user.
Returns:	NOERR or an appropriate error code.

m385_read () — Read measurement result

Syntax:	`int m385_read(M385_DEV *device, void *data)`
Description:	This function reads the result of a measurement started with the m385_ioctl() function.
Arguments:	M385_DEV *device pointer to M385 device. void *data pointer to measurement result. The data type is dependent of the measurement.
Returns:	NOERR or an appropriate error code.

5.3. Using the M385 Library

This section contains information on the usage of the M385 function library, the configuration of a measurement and retrieving measurement results are explained as well as performing specific measurement tasks.

5.3.1. General Information

The M385 can be programmed for specific measurements through a basic function library containing the following functions:

- m385_open() - opens a M385 device.
- m385_ioctl() - configure and start a measurement.
- m385_read() - get the result of a measurement.
- m385_close() - closes a M385 device.

First the M385 device must be opened using m385_open() for a specific device ID, the caller must provide an initialized pointer which refers to a storage location for the internal data structures with the type M385_DEV. Next a measurement must be configured and started using m385_ioctl(). Measurements can be configured for using interrupts or polling based and for using blocking or non blocking data reads. If the measurement has been configured for blocking reads the function m385_read() must be used to wait for the measurement to finish and get the measurement data. If the measurement has been configured for non-blocking reads the function m385_read() must be used to check wether a measurement has finished, if a measurement is still busy m385_read() returns ERR_DNRDY, if the measurement is done the measurement result is returned, if the result is NOERR the measurement data is passed in the format related to the measurement function. The measurement can be configured for raw data format, if so m385_read returns the contents of the timer and control registers regardless of the measurement result.

5.3.2. Performing a Measurement

This section describes the basic steps for performing any type of measurement.
Each measurement must be configured and started using the control function m385_ioctl() which takes two parameters, the device pointer and a pointer to a measurement configuration block with the format M385_FUNC_T. The measurement configuration block consists of a part which is the same for all types of measurement functions and a part which is measurement specific. Next the common part of the measurement configuration block is described:

function
: This field must be used to select the type of measurement, e.g. MEASURE_FREQUENCY or MEASURE_PERIOD etc.

input_a
: This sub-block must be used to configure the input conditioner for signal A.

input_b
: This sub-block must be used to configure the input conditioner for signal B.

read_blk
: This switch must be either set to TRUE for using blocking reads to retrieve data or must be set to false for using non-blocking reads.

irq_enable

This switch must be set to TRUE for interrupt based measurements or set to FALSE for polling based measurements. Wether to use interrupts or a polling based mechanism is up to the programmer and depends on the system requirements.

timeout

This field must be configured with the requested timeout value for any measurement. The timeout is defined as milliseconds. The M385 is able to measure signals with a period time from 4 nanoseconds up to 20 seconds, the measurement timeout must be set according to the frequency range of signals to be measured, to avoid unjustified timeouts or unnecessary long waiting time when the module does not trigger. For instance when the frequency of the input signals is always > 1 kHz, a good timeout value is 2 milliseconds (twice the longest period).

raw_data

For normal operation this switch is set to FALSE so data is represented according to the type of measurement, for instance frequency measurement delivers the frequency in [Hz]. If the switch is set to TRUE the unprocessed contents of the timer and counter are passed.

Both input signal conditioners must be configured at all times regardless of the input on which the measurement is performed. The signal conditioners must be configured using the M385_INP_COND blocks in the measurement configuration structure: input_a and input_b. The M385_INP_COND consists of the following fields:

level

This is the trigger level which must be set to any value between -5.0V and +5.0V. A good choice for the trigger level is 50% of the input signal span. Additionally the trigger level can be set to AUTO_LEVEL to request automatic selection of the trigger level. When auto-triggering is selected the library software will perform a voltage measurement to determine the maximum voltage and a voltage measurement to determine the minimum voltage of the input signal, consequently the trigger level will be set to 50% of the input signal.

hysteresis

The hysteresis of the trigger circuitry is programmable between 10 mV and 60 mV in steps of 5 mV. The macro HYST_MAX is used to select the maximum possible hysteresis (>60 mV), thus the minimum possible sensitivity and the macro HYST_MIN is provided to select the minimum hysteresis (5 mV) and therefore maximum sensitivity. The amount of required hysteresis depends on the input signal, with high hysteresis noise has less influence but low amplitude signals can not be detected, with low hysteresis low amplitude signal can be detected but noise could affect the measurement.

coupling

This field must be set to either COUPLING_DC or COUPLING_AC. The input circuitry can be configured to pass the signal: DC coupled or to subtract the DC component: AC coupled.

prescaler

The M385 analog front-end and trigger circuitry is galvanic isolated from the actual timer counter logic and M-module bus. The bandwidth of the isolation circuitry is limited to 50MHz. The M385 features a signal prescaler for measurements above 50MHz, the prescaler can be set to divide the clock signal derived from the input signal, by 2, 4, 8 or 16. Additionally the prescaler can be set to PRESCALER_AUTO to request the auto prescaler set function. When the auto prescaler function is requested, the M385 library will perform a frequency measurement with the prescaler set to 16x, prior to the actual measurement and will set the prescaler to the ideal setting.

Next the measurement specific parameter block must be configured. The measurement specific blocks are described in the remaining sections of this chapter. Once the measurement configuration block is fully configured the measurement must be started by calling m385_ioctl().
As soon as the measurement has finished the result can be read and processed by the application. Depending on the read_blk switch of the measurement configuration structure one of the following two schemes must be applied:

Blocking read

The function m385_read() can be called at any time and will wait until the measurement has finished and will finally return the measurement result and provided that the measurement was successful, pass the measurement data.

Non-blocking read

In this case the function m385_read() must be used to check wether or not the measurement is ready. If the measurement is busy, m385_read() will return ERR_DNRDY. If he measurement is ready the function returns the measurement result. If the result is NOERR, the measurement data is passed.

The function m385_read() must be called with two parameters, a device pointer and a pointer to the measurement data storage. The programmer must make sure enough space for the measurement data is available.
Next the measurement data can be processed, the format of the data depends on the raw_data switch of the measurement configuration structure and the performed measurement function. Refer to the M385_READ_FORMAT type definitions for details.

The following sections contain measurement function specific information.

5.3.3. Frequency Measurement

The frequency measurement uses the frequency function implemented in the hardware of the M385. The frequency measurement function will be configured for detecting rising edges on one of the two inputs.To initiate a frequency measurement the common parameter block of the measurement function must be configured in addition to the following specific fields:

input

The input INP_A or INP_B, on which the frequency measurement must be performed.

gate_tm

Gate time in milliseconds which is the time during which the rising edges of a repetitive input signal are counted.

A good setting of the timeout field in the common configuration block is twice the gate time. A timeout will occur when no edge is detected during the time programmed in the timeout field. Frequency measurement supports all prescaler settings, including PRESCALER_AUTO. The measurement result will be stored as the frequency in [Hz] in 64-bit double format.

5.3.4. Period Measurement

The period measurement uses the period function implemented in the hardware of the M385. The period measurement function will be configured for detecting rising edges on one of the two inputs.
To initiate a period measurement the common parameter block of the measurement function must be configured in addition to the following specific fields:

input
  The input INP_A or INP_B, on which the period measurement must be performed.

nr_msr
  Number of periods of the input signal of which the duration must be measured.

Period measurement is performed over a programmable number of cycles of the input signal, the result is the average period. The number of cycles over which is averaged is programmable between 1 and 0xFFFFFFFF and must be stored in the nr_msr field.

Period measurement supports all prescaler settings, including PRESCALER_AUTO. The measurement result will be stored as the period time in [seconds] in 64-bit double format and shows the average period over nr_msr cycles.

5.3.5. Pulse Width Measurement

The pulse width measurement uses the positive or negative pulse width function implemented in the hardware of the M385. To initiate a pulse width measurement the common parameter block of the measurement function must be configured in addition to the following specific fields:

input
  The input INP_A or INP_B, on which the pulse width measurement must be performed.

msr_type
  Measurement type, either POSITIVE_PW for positive pulse width measurement, or NEGATIVE_PW for negative measurement.

nr_msr
  Number of pulses of the input signal of which the duration must be measured.

Pulse width measurement is performed over a programmable number of cycles of the input signal, the result is the average period. The number of cycles over which is averaged is programmable between 1 and 0xfffffff and must be stored in the nr_msr field.

For pulse width measurement the prescaler must be set to PRESCALER_1X or PRESCALER_AUTO, if auto prescaler is requested the prescaler will be set to divide by one. The measurement result will be stored as the high or low time in [seconds] in 64-bit double format and shows the average pulse duration over nr_msr cycles.

For achieving duty cycle measurement, pulse width measurement must be combined with period measurement.

5.3.6. Voltage Measurement

Voltage measurement is a special measurement implemented in software, using the hardware period measurement function. Voltage measurements can be performed on sine-shaped input signals and consists of multiple period measurements with various trigger level settings using a successive approximation scheme, a voltage extreme is found as soon as the period measurement with a specific trigger level does not expire, the trigger level corresponds to the voltage value. The successive approximation mechanism requires any number of period measurements between 4 and 40.

The following types of voltage measurements are possible:

Maximum value:

The maximum voltage of an input signal is determined by varying the trigger level from the positive extreme to the negative extreme and performing a period measurement for each level setting, as soon as the period measurement does not expire (measurement ready, no timeout) the current trigger level is the maximum value.

Minimum value:

The minimum voltage of an input signal is determined by varying the trigger level from the negative extreme to the positive extreme and performing a period measurement for each level setting, as soon as the period measurement does not expire (measurement ready, no timeout) the current trigger level is the minimum value.

AC value:

The AC voltage is defined as the peak to peak value of a sine signal. The peak to peak voltage of an input signal is determined by measuring both the maximum and the minimum voltage, the peak to peak voltage is calculated with the following formula:

$$V_{ac} = (V_{max} - V_{min})$$

DC value:

The DC voltage is defined as the offset value of a sine signal. The offset voltage of an input signal is determined by measuring both the maximum and the minimum voltage, the offset voltage is calculated with the following formula:

$$V_{dc} = \frac{(V_{max} + V_{min})}{2}$$

To initiate a voltage measurement the common parameter block of the measurement function must be configured as well as following specific fields:

input

The input INP_A or INP_B, on which the voltage measurement must be performed.

msr_type

Measurement type, VMAX for measuring the maximum voltage, VMIN for measuring the minimum voltage, VAC measuring the peak voltage of a sine or VDC for measuring the DC component, or offset of a sine shaped signal.

timespan

Voltage measurement is achieved by performing repeatedly period measurements with a varying trigger level until no timeout occurs. The timeout for voltage measurement must be programmed via the timespan field.
The timespan must be long enough to pass at least 2 periods of the input signal. A long timespan results in a long measurement time since the number of required period measurements is at least 4 and at the most 40. The best performance is achieved with the timespan set to twice the period time of the lowest frequency of the input signal.

The result of a voltage measurement is represented as a 4-byte floating point value and the unit is Volts.

5.3.7. Software Gated Totalize Measurement

The software gated totalize measurement uses the totalize gated by software measurement function of the M385. To initiate the software gated totalize function the following specific fields are required in addition to the common part of the measurement configuration structure:

input
: The input INP_A or INP_B, on which the events must be counted.

edge
: Edge type, either RISING for counting on rising edges of the input signal or FALLING for counting falling edges.

Unlike other measurements the totalize measurement does not finish. The totalize function must be initialized once and the function starts counting events. The current counter can be read at any time using the m385_read() function. The totalize gated by software function will remain active until a new measurement is initiated. The data format is the raw 32-bit counter value.

5.3.8. Hardware Gated Totalize Measurement

The hardware gated totalize measurement uses the totalize gated by hardware measurement function of the M385. This function will start counting edges on input A as long as the gate signal on input B is active. Both the edge to count on as the polarity of the gate signal are programmable. Additionally the measurement can be repeated automatically for a programmable number of times providing an average count. To initiate the totalize function the following specific fields are required in addition to the common part of the measurement configuration structure:

edge
: This field must be programmed to either RISING or FALLING and specifies the edge of the signal connected to input A to count on.

gate
: Input B is used as a gate signal, the polarity of the gate signal can be programmed with this field to active high or active low.

nr_msr
: This field is provided to program the number of measurements that have to be performed.

The totalize gated by hardware function finishes as soon as the gate signal becomes inactive, the data format of the measurement result is a raw 32-bit counter value. If the nr_msr field is more than 1, then the measurement is repeated for the requested amount of times and the result is the average counter value.

5.3.9. Time Interval Measurement.

The time interval measurement is a hardware measurement function provided by the M385 timer counter logic. This function measures the phase between a signal applied to input A and a signal with the same frequency applied to input B. In addition to the command parameter block the following fields of the measurement configuration structure must be configured:

edge_inpa
: The event edge for input A, RISING or FALLING

edge_inpb
: The event edge for input B, RISING or FALLING

nr_msr
: Number of measurements that have to be performed. If greater than 1 the measurement result is the average over nr_msr measurements.

For time interval measurement the prescaler must be set to PRESCALER_1X or PRESCALER_AUTO, if auto prescaler is requested the prescaler will be set to divide by one. The measurement result will be stored as the phase difference in [seconds] in 64-bit double format and shows the average phase over nr_msr cycles.

5.3.10. Rise and Fall Time Measurement.

For the rise and fall time measurement both inputs are connected together by an onboard relay without affecting the input impedance at the connector. Input A must be used to connect the measurement source. The rise and fall time measurement is based on the a hardware time interval measurement, with the trigger level of input A set to 10% and the level of input B set to 90%. The rise and fall time measurement is partly implemented in software, when initiated first the minimum and maximum voltages of the input signal is measured as described in section 5.3.6. and the voltage span is calculated. Next the trigger level of input A will be set to 10% of the span and the trigger level of input stage B will be set to 90% of the span. Finally a time interval measurement is performed using the rising edges, the measurement result is the rise time of the signal. For measuring the fall time the same procedure is followed, except that the trigger level of input A is set to 90%, the trigger level for input B is set to 10% and the time interval measurement is performed using falling edges.
In addition to the command parameter block the following fields of the measurement configuration structure must be configured:

msr_type
: The event edge, RTIM for measuring the rise time or FTIM for measuring the fall time.

nr_msr
: Number of measurements that have to be performed. If greater than 1 the measurement result is the average over nr_msr measurements.

level
: Signal level of the input signal. A predefined level can be set (TTL, CMOS and ECL) or can be determined automatically when level is set to AUTO_SIG_LEV. When level is set to determine automatically, the endurance of the measurement will increase.

For rise and fall time measurement the prescaler must be set to PRESCALER_1X or PRESCALER_AUTO, if auto prescaler is requested the prescaler will be set to divide by one. The measurement result will be stored as the a time value in [seconds] in 64-bit double format and shows the average time over nr_msr cycles.

5.3.11. Ratio Measurement

The ratio measurement uses the ratio hardware measurement function of the M385 to measure the proportion of the frequency of the signal applied to input A in relation to the signal applied to input B. To initiate the ratio function the following specific field is required in addition to the common part of the measurement configuration structure:

msr_type

With this field the nature of the frequency ratio measurement must be configured to either signal A over signal B or signal B over signal A.

nr_msr

Number of measurements that have to be performed. If greater than 1 the measurement result is the average over nr_msr measurements.

The frequency ratio will be represented as a 64-bit double number.

5.4. AUTO TRIGGER LEVEL FUNCTION

The M385 library software is able to perform an auto set function for configuration of the trigger level. To request auto triggering, the value AUTO_LEVEL must be programmed in the required level field of the measurement configuration structure.
The auto trigger level request causes the library software to perform a voltage measurement for determining the maximum voltage and a voltage measurement for determining the minimum voltage, prior to the actual measurement. The found maximum and minimum value are used for configuration of the trigger level according to the following equation:

$$V_{trigger} = V_{min} + 50 \cdot \frac{(V_{max} - V_{min})}{100}$$

The trigger level is set to 50% of the input signal. The maximum and minimum voltage of a signal are determined using period measurement. The trigger level is set 0.0 V and is increased/decreased with a value defined with the preprocessor macro AT_DELTA, till the period measurement expires. Now the voltage window is determined, the trigger level is set back to the under level of the voltage window and is increased/ decreased with small steps defined with the preprocessor macro AT_RES. The maximum or minimum voltage is found when the period measurement expires. The voltage measurement of the auto trigger function uses fixed time span defined with the preprocessor macro ATTS. The minimum frequency of a signal on which the auto trigger function can be used depends on the value of ATTS, for details refer to section 5.3.6.

5.5. AUTO PRESCALER FUNCTION

The M385 library software is able to perform an auto set function for configuration of the prescaler. To request auto prescaler, the value PRESCALER_AUTO must be programmed in the required prescaler field of the measurement configuration structure. If the prescaler is set to PRESCALER_AUTO a frequency measurement is performed by the library prior to performing the requested measurement. The additional frequency measurement is performed with the prescaler set to PRESCALER_16X. The result is the frequency of the input signal and the prescaler will be set accordingly.
The auto prescaler function can be used in combination with the following functions:

- MEASURE_FREQUENCY
- MEASURE_PERIOD
- MEASURE_VOLTAGE
- MEASURE_TOTALIZE_SW
- MEASURE_TOTALIZE_HW
- MEASURE_RATIO

5.6. SOFTWARE DISTRIBUTION

This section gives an overview of the software distribution.

File	Description
M385\SOFTWARE\m385rel.txt	Release notes
M385\SOFTWARE\LIB\m385defs.h	Module definitions
M385\SOFTWARE\LIB\m385lib.c	APIS based ANSI-C library
M385\SOFTWARE\LIB\m385lib.h	Library definitions
M385\SOFTWARE\LIB\misclib.c	Miscellaneous function library
M385\SOFTWARE\LIB\misclib.h	Definitions for miscLib
M385\SOFTWARE\LIB\eeplib.c	EEPROM access library
M385\SOFTWARE\LIB\eeplib.h	Definitions for eepLib
M385\SOFTWARE\LIB\spilib.c	SPI function library
M385\SOFTWARE\LIB\spilib.h	Definitions for spiLib
M385\SOFTWARE\LIB\callib.h	Calibration data definitions
M385\SOFTWARE\LIB\tunelib.h	Tuning data definitions.
M385\SOFTWARE\EXAMPLE\m385demo.c	Demo program
M385\SOFTWARE\EXAMPLE\makefile.bor	Borland example make file
M385\SOFTWARE\EXAMPLE\makefile.os9	OS-9 example make file
M385\SOFTWARE\EXAMPLE\makefile.lin	Linux example make file
M385\SOFTWARE\EXAMPLE\makefile.usb	Borland makefile for i5000win platform

M385 example software is APIS based, therefore APIS support for the target platform is required for code generation.

The following figure is an example of the M385 software integrated in the APIS environment with as target platform the i4000/OS-9.

```
+---PROJECT                             AcQ’s distribution
     +---APIS                           APIS basis distribution
     |    +---DOC                       APIS documentation
     |    +---SOFTWARE
     |         | readme.txt             Distribution overview
     |         +---COMMON
     |         |    +---DEFS
     |         |    |    apis.h         General definitions
     |         |    +---OS9TRAP
     |         |    |    +---CMDS
     |         |    |         apistrap  OS-9 trap handler
     |         +---I4000OS9             i4000/OS-9 support
     |         |    relnotes.txt        Release notes / version info
     |         |    apis_i4000os9.h     Platform specific definitions
     |         |    apis_i4000os9.c     Platform support library
     +---M385                           M385 distribution
          +---SOFTWARE
                    |   m385rel.txt     release notes
                    +---LIB             M385 function library
                    |       m385lib.c
                    |       m385lib.h
                    |       m385defs.h
                    |       misclib.c
                    |       misclib.h
                    |       eeplib.c
                    |       eeplib.h
                    |       spilib.c
                    |       spilib.h
                    |       callib.h
                    |       tunelib.h
                    +---EXAMPLE         M385 example application
                            m385demo.c
                            makefile.*  Makefiles
```

Code generation is platform dependent, for information on building the software please refer to the release notes of the target platform and the APIS Programmer’s Manual.

6. ANNEX

6.1. BIBLIOGRAPHY

Specification for M-module interface and physical dimensions
M-module specification manual, April 1996, MUMM.
Simon-Schöffel-Strasse 21, D-90427 Nürnberg, Germany.

APIS Programmer’s Manual
AcQuisition Technology
P.O. Box 627, 5340 AP Oss, The Netherlands.
Downloadable from www.acq.nl

LTC1446 Data Sheet
LTC1446/LTC1446L Dual 12-Bit Rail-to-Rail Micropower DACs in SO-8
Linear Technology Corporation
1630 McCarthy Blvd., Milpitas, CA 95035-7417, USA

6.2. COMPONENT IMAGE

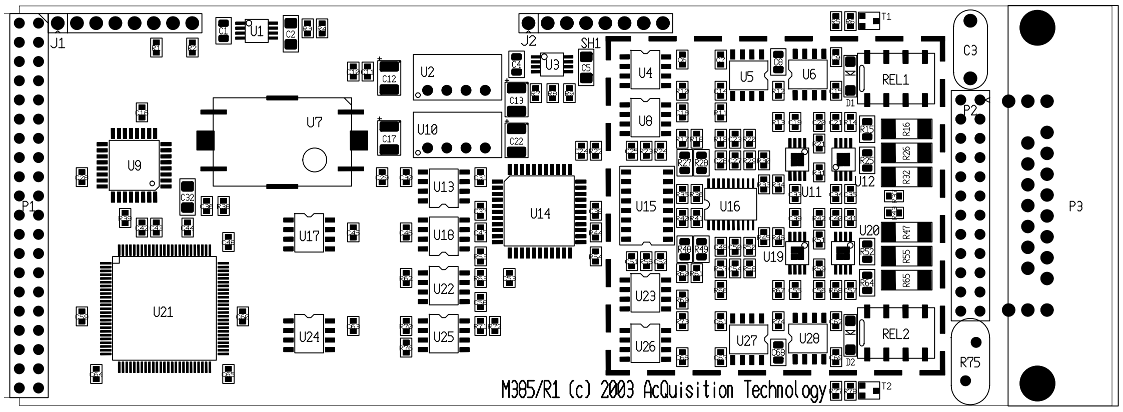

Figure 17 Component Image

6.3. OPERATING MODE SPECIFICATIONS

Operating Mode		Specifications
Frequency	Range	0.05 Hz to 250 MHz
	LSD	(5 ns / Gate Time) * FREQ
Period	Range	5 ns to 20 s
	LSD	(5 ns / Gate Time) * PER
Time Interval 1-2	Range	20 ns to 20 s (single shot)
	LSD	5 ns
Frequency Ratio	Range	0.4E-9 to 2.5E+9

Operating Mode		Specifications
Totalize	Range	0 to 1 x 10^{32} -1
	LSD	1 count of input signal
Rise/Fall Time	Range	20 ns to 20 s (single shot)
	LSD	5 ns
Positive, Negative Pulse Width	Range	20 ns to 20 s
	LSD	(5 ns / Gate Time) * PULSE WIDTH
Voltage		± 5 V

6.4. INPUT SPECIFICATIONS

The two inputs on the M385 have the following specifications:

Input Characteristics		Specifications
Input 1 Range	dc coupled	0 to 250 MHz
	ac coupled	10 Hz to 250 MHz
Input 2 Range	dc coupled	0 to 250 MHz
	ac coupled	10 Hz to 250 MHz
Sensitivity 1,2 (MAX)		35 mV rms sine wave, 100 mV pk-pk
50 Ohm	Dynamic Range (ac)	10 V pk-pk
	Signal Operating Range (dc)	± 5 V
Trigger Level Range 1,2		± 5 V with stepsize of 2.5 mV

6.5. TCXO TIME BASE

The TCXO Time Base on the M385 has the following specifications:

TCXO Time Base		Specifications
Frequency		50 MHz
Stability	Aging Rate	< 1 x 10^{-6}/year
	Temperature	< 1 x 10^{-6},0 to 60 °C
	Line Voltage	< 3 x 10-7 for ±5% change

6.6. TECHNICAL DATA

Slots on the base-board:
Requires one 16-bit M-module slot.

Connection:
To base-board via 40 pole M-module interface.
To peripheral via 15 pole D-sub connector with two coaxial contacts.
To peripheral via 24 pole female header connector.

Power supply:
+5VDC +/-10%, typical 650 mA.

Temperature range:

Operating:	0..+60°C.
Storage:-	20..+70°C.

Humidity:
Class F, non-condensing.

6.7. Document History

- Version 1.0
  First release

- Version 1.1
  Explanation of SPI register changed.

- Version 2.0
  Manual revised to reflect M385 hardware revision 1.x

- Version 2.1
  Ratio measurement function: removed signal A frequency must be higher restriction and software measurement function can be configured to determine the ratio of input A over B or input B over A.
  Totalize measurement renamed to totalize gated by software measurement.
  Totalize gated by hardware measurement function description added.
  Time interval function modified: edges of both signals are now programmable.
  Voltage measurement modified: VAC measurement delivers the peak-peak value.
  Rise and fall time measurement modified: no external connection of input A and B required, the required connection is made by an on-board relay controlled by the library software.

- Version 2.2
  Description of prescalers added.
  Measurement ranges modified and some details regarding measurement functions added.

- Version 2.3
  Contents of table in section 4.6.2. modified

- Version 2.4
  Extra parameter added for rise/fall time measurement.
  Added note about changing from DC coupling to AC coupling.
  Auto trigger level function changed.