

## F2802x Burn In Unit

### Code Reference Manual

Wed Nov 6 2013



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	General Operation . . . . .	1
1.2	Digital Power Control . . . . .	2
<b>2</b>	<b>Unit Programming</b>	<b>7</b>
2.1	Language . . . . .	7
2.2	Instrument Selection . . . . .	8
2.3	The Command Tree . . . . .	9
<b>3</b>	<b>Command Reference</b>	<b>11</b>
3.1	CALibration Subsystem . . . . .	11
3.2	CONTRol Subsystem . . . . .	12
3.3	INSTrument Subsystem . . . . .	12
3.4	OUTPut Subsystem . . . . .	13
3.5	INPut Subsystem . . . . .	13
3.6	SOURce Subsystem . . . . .	13
3.7	MEASure Subsystem . . . . .	16
3.8	STATus Subsystem . . . . .	17
3.9	SYSTem . . . . .	17
<b>4</b>	<b>Data Structure Documentation</b>	<b>19</b>
4.1	acStageNets Struct Reference . . . . .	19
4.2	acStageSettings Struct Reference . . . . .	20
4.3	extDeviceSettings Struct Reference . . . . .	21
4.4	i2cMsg Struct Reference . . . . .	22
4.5	loadStageNets Struct Reference . . . . .	23
4.6	loadStageSettings Struct Reference . . . . .	24
4.7	xfrmStageNets Struct Reference . . . . .	25
4.8	xfrmStageSettings Struct Reference . . . . .	26
<b>5</b>	<b>File Documentation</b>	<b>29</b>
5.1	Adc.h File Reference . . . . .	29
5.2	Cntl.h File Reference . . . . .	32

5.3	Common.h File Reference . . . . .	38
5.4	Comparator.h File Reference . . . . .	39
5.5	EnableCtrl.h File Reference . . . . .	41
5.6	I2c.h File Reference . . . . .	46
5.7	MacroNets.h File Reference . . . . .	50
5.8	Ocp.h File Reference . . . . .	55
5.9	Opp.h File Reference . . . . .	60
5.10	Otp.h File Reference . . . . .	63
5.11	Ovp.h File Reference . . . . .	69
5.12	Pwm.h File Reference . . . . .	74
5.13	Sci.h File Reference . . . . .	75
5.14	SCPI_specificCmds.h File Reference . . . . .	77
5.15	Settings.h File Reference . . . . .	78
5.16	SineGen.h File Reference . . . . .	83
5.17	slaveMode.h File Reference . . . . .	87
5.18	SlewControl.h File Reference . . . . .	89
5.19	Spi.h File Reference . . . . .	93
5.20	StateMachine.h File Reference . . . . .	96
5.21	Timers.h File Reference . . . . .	97
5.22	Tmp.h File Reference . . . . .	98

# Chapter 1

## Introduction

This document forms the reference manual for the Burn In Unit based on the Texas Instruments F2802x.

This system has been implemented using several Texas Instruments libraries; digital power v3.1, SGen V1.01, IQ-Math v1.5c and SQMath. Please refer to the TI documentation pertaining to those libraries and to the C28x device to understand their operation.

This system also uses the System for Standard Commands for Programmable Instruments for C28x, again, refer to the documentation pertaining to that system to understand its use.

### 1.1 General Operation

This system operates using the concept of a state machine with multiplexed threading. This allows important tasks, such as signal generation and the monitoring of critical parameters to take precedence while still allowing other less critical tasks to run with acceptable frequency. This is allowed by creating several threads DPL, A, B and C and iterating them at different interval periods, such that every time the thread is iterated the next single task in that thread is run.

#### 1.1.1 Fast (DPL) Thread

The DPL thread is the fastest thread, is coded in assembly and consists of only two tasks. It is the only thread that uses an interrupt to iterate. Thus the tasks are formed by an interrupt service routine (ISR). This allows the DPL thread to interrupt the ongoing action of any of the other threads. This means that as long as the length of time each of the tasks take to execute is less than the interval time, then the threads operation will be deterministic. It is within this thread's two tasks that the critical and time-sensitive actions of the program are carried out. The thread ISR is triggered by a start of conversion (SOC) signal from the master enhanced pulse width modulation (ePWM) peripheral. This occurs at a frequency of 33 kHz. As there are two tasks on the thread, the frequency of each task is 16.5 kHz resulting in a period of 60us.

#### 1.1.2 Sequential Iteration Threads

The remaining threads (A, B, C) iterate by checking the rollover state of different CPU timers. Their operation and iteration is sequential and so less deterministic. The outline of the handling sequential iteration threads by the state machine is illustrated by the following flow diagram, where N and n rollover to A and 0, respectively whenever they reach their maximum value.

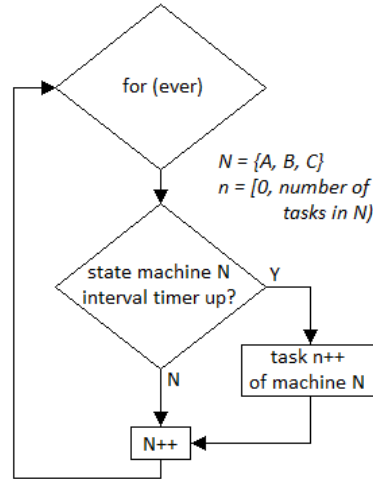


Figure 1.1: Sequential iteration thread state machine

The frequency of tasks can be determined from the period of the interval timers using the following equation, where  $n$  is the number of tasks in the thread and  $T_{in}$  is the interval timer period.

$$f_{ta} = \frac{1}{T_{in} \times n}$$

With interval periods such that  $T_A < T_B < T_C$  this results in the flow of execution illustrated by the following figure.

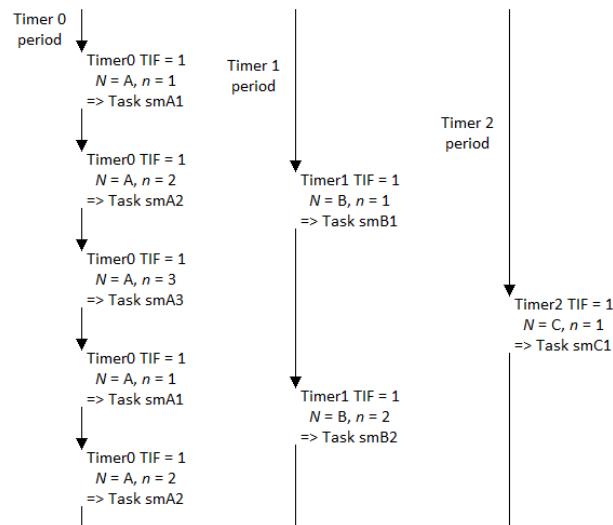


Figure 1.2: Sequence of thread tasks

## 1.2 Digital Power Control

The control of the power, voltage and current is managed using control loop logic, normally one loop for each stage (Transformer, AC and each of the four loads). These control loops are formed from a set of macros and helper functions.

Each of these macros acts as a block of functionality, much like a black-box electronic component, allowing their terminals to be wired together in the required order with nets. Once initialised and configured, or connected, correctly the macro blocks are executed, sequentially, by calling the appropriate code from an assembly ISR. In this

case, this is what forms the DPL ISR tasks. Each block performs a precisely defined computation and delivers the result to the appropriate net-list variables. Each control loop's net variables and operation parameter variables are grouped together and stored in a structure (channelParameters), with one such structure for each control loop.

The basic use of macros that are available within the program can be broken down into three parts:

1. Initialisation: the macro blocks are initialised from the program by using a C callable function, `DPL_Init()`, which is contained within the assembly file `DPL_ISR.asm`.
2. Configuration: C pointers of the macro block terminals are assigned to net nodes to form the desired control structure, i.e., the control loop.
3. Execution: Macro block code is executed in the assembly ISR, `DPL_ISR()`.

### 1.2.1 Boost Load Control

The following figure shows a diagram that illustrates the macro and net view of the control loop for one of the load boost converter stages. The loop depicted is current controlled. It uses a macro to retrieve a reading from an ADC peripheral to obtain a value that indicates the level of the current at the low side of the load input,  $CHANn\ I_{SNS}$ . This value is used as the feedback into a 2-pole 2-zero filter macro where it is compared to the user-set reference value and the output value is varied up or down, using the filter coefficients, as needed to move it towards matching the reference. This output value is then used by another macro to set the duty of the ePWM peripheral that controls the switching of the converter,  $CHANn\ PWM$ .

Another macro retrieves a value from a second ADC that represents the voltage level across the load input,  $CHANn\ V_{SNS}$ . A task in one of the sequential iteration threads compares this value to the user-set allowable voltage limit, the OVP limit. If the value is above this limit the control loop output is disabled and an over-voltage protection alert is raised.

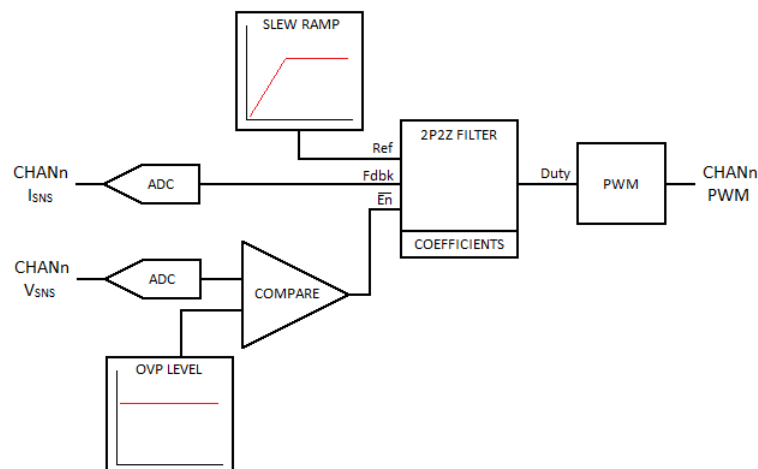


Figure 1.3: Macro view of control loop for single boost load

### 1.2.2 Transformer Control

The following figure shows a diagram that illustrates the macro and net view of the control for the transformer boost converter stage. The stage uses a fixed PWM frequency and duty cycle.

An on-board analogue comparator compares the value of DC MID  $V_{SNS}$  against the user-set allowable voltage limit, the OVP limit. If the value is above this limit the comparator output sets the related trip-zone, quickly disabling the output in software and hardware.

A macro is used to retrieve a reading from an ADC peripheral to obtain a value that indicates the level of the voltage across the stage output, DC HV  $V_{SNS}$ . This value is used by a task in one of the sequential iteration threads to

compare to the user-set allowable voltage limit, the OVP limit. If the value is above this limit the output is disabled and an over-voltage protection alert is raised.

Another macro retrieves a value from a second ADC that represents the current level at the low side of the input, DC MID  $I_{SNS}$ . This value is used by a task in one of the sequential iteration threads to compare to the user-set allowable voltage limit, the OCP limit. If the value is above this limit the output is disabled and an over-current protection alert is raised.

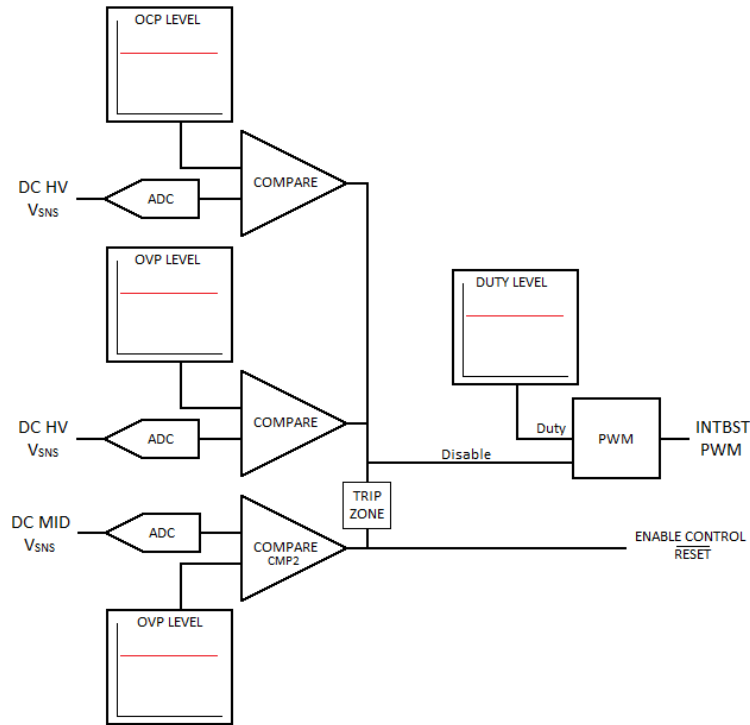


Figure 1.4: Macro view of control loop for inter-boost

### 1.2.3 AC Bridge Control

The following figure shows a diagram that illustrates the macro and net view of the control loop for the AC bridge converter stage. The loop depicted is both voltage *and* current controlled. It uses a macro to retrieve a reading from an ADC peripheral to obtain a value that indicates the voltage level across the stage output, AC  $V_{SNS}$ . This value is used as the feedback into a 2-pole 2-zero voltage filter macro where it is compared to the generated sine wave reference and the output value is varied up or down, using the filter coefficients, as needed to move towards matching the reference.

Another macro retrieves a value from a second ADC that represents the current level at the low side of the stage input, AC  $I_{SNS}$ . This value is used as the feedback into a 2-pole 2-zero current filter macro where it is compared to one of two reference signals, depending on the mode of the unit, and the output is varied up or down, using the filter coefficients, as needed to move towards matching the reference. If the unit is a stand-alone unit or a master unit the reference input is connected to the output of the voltage filter. If the unit is a slave the AC  $V_{SNS}$  signal is replaced by the external AC  $I$  signal from the master and is connected directly to the reference input of the 2-pole 2-zero current filter macro.

An on board analogue comparator also compares this AC  $I_{SNS}$  value against the user-set allowable current limit, the OCP limit. If the value is above the limit, the comparator output sets the related trip zone, quickly disabling the control loop output and all other stages output, STOP ALL.



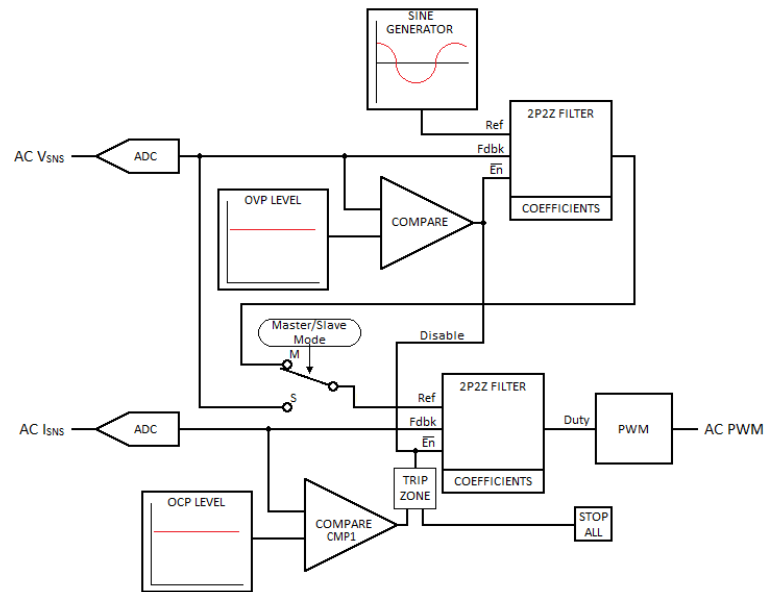


Figure 1.5: Macro view of control loop for AC bridge



## Chapter 2

# Unit Programming

### 2.1 Language

For remote programming operations the burn in unit uses commands that follow the SCPI syntax as outlined in the System for Standard Commands for Programmable Instruments for C28x Code Reference Manual, where in the standard programming commands that are included by this system are also outlined. Please refer to that document and, where necessary, any further referenced documents, for information on the syntax of the programming commands.

This unit programming chapter will focus on the function of the commands specific to the burn in unit

#### 2.1.1 Format

The burn in unit uses a standard raw TCP/IP Ethernet connection to communicate with a controlling system. Once the user's control system has established a valid raw TCP/IP connection with the burn in unit commands should be sent and received as ASCII character byte string data, with the most significant byte first.

Thus the data sent that would correspond to the <COMMON QUERY PROGRAM HEADER> \*IDN?; in transmission order would be (including the '^NL' <PROGRAM MESSAGE TERMINATOR>)

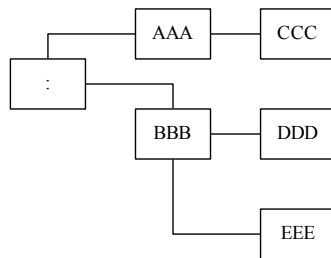
```
0x2A 0x49 0x44 0x4E 0x3F 0x3B 0x0A
```

#### 2.1.2 Program Message Unit Creation

The user may create a <PROGRAM MESSAGE UNIT> by traversing the command tree from the root towards the desired function and concatenating the <program mnemonic>s and <COMMAND PROGRAM HEADER> or <QUERY PROGRAM HEADER> stepped through to reach that destination.

For example, using the hypothetical command tree shown in the following figure, the <PROGRAM MESSAGE UNIT> required to run the function 'EEE' would be:

```
BBB:EEE;
```



### 2.1.3 Queries and Responses

Each <PROGRAM MESSAGE> may contain only one <QUERY PROGRAM HEADER> which, if used, should be the last header in the message. If a <QUERY PROGRAM HEADER> is used the next operation should be to read the response generated by the query, before any further <PROGRAM MESSAGE> elements are sent to the unit.

## 2.2 Instrument Selection

Each burn in unit is split into several logical instruments that corresponds to the functional sections of the burn in unit. The instrument selection is controlled using the `INSTRUMENT` sub-system. These instruments are numbered according to the unit channel numbering shown in the following list. These numbers can be obtained by using the `CATalog?` query from the `INSTRUMENT` subsystem:

```
INSTRUMENT:CATalog?;
```

0. Load 0
1. Load 1
2. Load 2
3. Load 3
4. AC Current Control
5. DC Stage

## 6. AC Stage

This means that before setting any of the instruments' settings, the relevant instrument number must first be selected. This is done using the `NSElect` command from the `INSTRUMENT` subsystem. The default instrument selection is 0. Once selected the value is "sticky", and does not need to be set again until the user settings wishes to send commands for a different instrument. Thus, for example, the `<PROGRAM MESSAGE>` required to set the voltage level DC stage to 10 volts with a range set to 1V/V would be as follows:

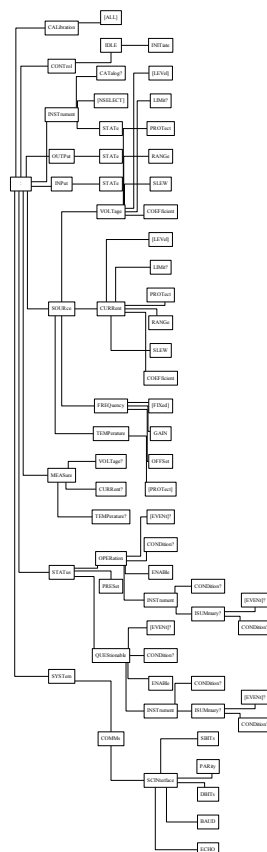
```
INSTRUMENT:NSElect 5;:SOURCE:VOLTage:LEVel 10;:SOURCE:VOLTage:RA-
NGe 1;
```

## 2.3 The Command Tree

The command tree provides the structure of the subsystems and their commands. As shown however the diagram does not illustrate that some of the commands have no meaning to some instruments. For example, attempting to set the voltage coefficients of a current controlled instrument will result in an error.

Again, the commands shown here are in addition to those that are implemented as standard by the System for Standard Commands for Programmable Instruments for C28x.

The diagram is followed by a listing of each command. The next section, [Command Reference](#) explains the operation and details of each command.



**CALibration**

ALL

**CONTROL**

IDLE

```
        INITiate
INSTRument
    CATalog?
    NSElect
    STATE
OUTPut
    STATE
INPut
    STATE
SOURce
    VOLTage
        LEVel
        LIMit?
        PROtect
        RANGE
        SLEW
        COEFFicient
    CURRent
        LEVel
        LIMit?
        PROtect
        RANGE
        SLEW
        COEFFicient
    FREQuency
        FIXed
        GAIN
        OFFSet
    TEMPerature
        PROtect
MEASure
    VOLTage?
    CURRent?
    TEMPerature?
STATus
    OPERation
    PRESet
    QUEStionable
SYSTem
    COMMs
        SCINterface
            SBITs
            PARity
            DBITs
            BAUD
            ECHO
```

## Chapter 3

# Command Reference

This section outlines the operation of each command and node on the command tree. All commands, unless otherwise specified, also have an equivalent query form that returns the value that the command is currently set to. For example, querying `SOURce:VOLTage:PROTect?` will return the current voltage protection level setting.

Some commands have a query form *only*, that is they have no none-query form. these commands are identified by the presence of the query terminator character, '?'.

Some of the commands are optional, or default. Such optionality is represented by the presence of square brackets enclosing the command, '[' and ']'. Where commands are optional, it is possible to use the parent node alone instead of needing to append the optional child command. For example, as the `:NSElect` command from the `INSTRument` subsystem is marked as optional, the command to select an instrument can be either:

```
INSTRument:NSElect <numeric_value>;
```

or

```
INSTRument <numeric_value>;
```

### 3.1 CALibration Subsystem

This subsystem has the function of performing system calibration. Currently the subsystem has only one function, the default command `ALL` and it's complementing query.

#### 3.1.1 [:ALL]

```
CALibration:ALL
```

The `CALibration:ALL` command performs the same function as the `CALibration:ALL?` command except there is no response. Calibration errors are reported through the status-reporting mechanism. While this command is executing the `CALibrating` bit of the `Operation Status Condition` register will be set.

#### 3.1.2 [:ALL]?

```
Calibration:ALL?
```

The `ALL?` query performs a full calibration of the instrument and responds with a `<numeric_value>` indicating the success of the calibration. A zero will be returned if calibration is completed successfully; otherwise a nonzero value which represents the appropriate error number shall be returned. An instrument will still report calibration errors through the status-reporting mechanism, even though an error is reported by the value of the query response.

## 3.2 CONTrol Subsystem

The CONTrol subsystem is used to turn on and off or control the state of the entire device.

### 3.2.1 :IDLE

CONTrol:IDLE

This node contains commands that control the IDLE state of the device.

#### 3.2.1.1 :INITiate

CONTrol:IDLE:INITiate

Returns the device to the idle state. This command is an event and has no associated \*RST condition or query command.

## 3.3 INSTRument Subsystem

This device uses multiple logical instruments (as described in [Instrument Selection](#)), this subsystem provides the mechanism to identify and select logical instruments by number.

### 3.3.1 :CATalog?

INSTRument:CATalog?

The CATalog query returns a comma-separated list of the numbers of all logical instruments.

### 3.3.2 :NSElect <numeric\_value>

INSTRument:NSElect

This command selects an instrument according to the numeric value parameter. When a logical instrument is selected, all other logical instruments or groups are unavailable for programming until selected. When queried it shall return the selected logical instrument number.

Note that the numbering used for logical instruments directly corresponds to the numbers used in status reporting for the multiple instruments; specifically the STATus:QUESTionable:INSTRument and STATus:OPERation:INSTRument commands.

At \*RST, instrument 0 is selected.

### 3.3.3 :STATe <Boolean>

INSTRument:STATe

Turns the selected logical instrument ON or OFF. A logical instrument does not have to be turned OFF before another logical instrument is selected. That is, several logical instruments may be active, while only one may be selected. When an instrument is active, measurements occur and signals are generated. When inactive, measurements do not occur and signals are not generated. When a logical instrument is selected, yet is inactive, all commands shall be processed so that the logical instrument shall reflect the state changes requested when the logical instrument is turned on.

At \*RST, this value is OFF.



## 3.4 OUTPut Subsystem

The OUTPut subsystem controls the characteristics of the selected instrument's output port.

### 3.4.1 [:STATe] <Boolean>

OUTPut:STATe

Selects the state of the output. When STATe is ON the signal generated by the source is emitted from the output port of the selected instrument.

## 3.5 INPut Subsystem

The INPut subsystem controls the characteristics of the selected instrument's input port.

### 3.5.1 :STATe <Boolean>

INPut:STATe

Selects the state of the input. When STATe is ON the signal generated by the source is emitted from the output port of the selected instrument.

## 3.6 SOURce Subsystem

The SOURce setup commands are divided into several sections. Each section or subsystem deals with controls that directly affect device-specific settings of the selected instrument.

### 3.6.1 :VOLTage

This subsection controls the signal voltage characteristics of the source.

#### 3.6.1.1 [:LEVel] <numeric\_value>

SOURce:VOLTage:LEVel

This command sets the target voltage level of a voltage controlled instrument. The numeric value parameter is the level to be set in volts, between 0 and the voltage limit returned in response the LIMit? query.

#### 3.6.1.2 :LIMit?

SOURce:VOLTage:LEVel

This command has a query form only. This is because the voltage limit value is fixed. The query response is a single numeric value in volts representing this fixed limit.

#### 3.6.1.3 :PROTect <numeric\_value>

SOURce:VOLTage:PROTect

This command sets the voltage protection level. This is the level at which the over-voltage protection is triggered if the voltage rises above it. The numeric value parameter is the level to be set in volts, between 0 and the voltage limit returned in response to the LIMit? query.

**3.6.1.4 :RANGe <numeric\_value>**

SOURce:VOLTage:RANGe

This command sets the voltage range. The numeric value parameter is the voltage scaling value in volts-per-volt from 0 to 1.

**3.6.1.5 :SLEW <numeric\_value>**

SOURce:VOLTage:SLEW

This command sets the voltage slew for a voltage controlled instrument. The numeric value parameter is the voltage slew rate value in volts-per-microseconds from 0 to 1.

**3.6.1.6 :COEFFicient <numeric\_value>, <numeric\_value>**

SOURce:VOLTage:COEFFicient

This command is used to specify the voltage control coefficients, of a voltage controlled instrument. The first numeric value parameter specifies the coefficient to be changed, according to the following list, while the second numeric value parameter specifies the level the selected coefficient is to be changed to.

- 0. Saturation Minimum
- 1. Saturation Maximum
- 2. B0
- 3. B1
- 4. A1
- 5. B2
- 6. A2
- 7. B3
- 8. A3

When queried, the response is a list of comma separated numeric values in the following order:

Saturation-Minimum-value, Saturation-Maximum-value, B0-value, B1-value, A1-value, B2-value, A2-value, B3-value, A3-value

**3.6.2 :CURRent**

SOURce:CURRent

This subsection controls the signal current characteristics of the source.

**3.6.2.1 [:LEVel] <numeric\_value>**

SOURce:CURRent:LEVel

This command sets the target current level of a current controlled instrument. The numeric value parameter is the level to be set in amps, between 0 and the current limit returned in response the LIMit? query.

## 3.6.2.2 :LIMit?

SOURce:CURRent:LEVel

This command has a query form only. This is because the current limit value is fixed. The query response is a single numeric value in amps representing this fixed limit.

## 3.6.2.3 :PROTect &lt;numeric\_value&gt;

SOURce:VOLTage:PROTect

This command sets the current protection level. This is the level at which the over-current protection is triggered if the current rises above it. The numeric value parameter is the level to be set in amps, between 0 and the current limit returned in response to the LIMit? query.

## 3.6.2.4 :RANGe &lt;numeric\_value&gt;

SOURce:CURRent:RANGe

This command sets the current range. The numeric value parameter is the current scaling value in amps-per-volt from 0 to 1. <---!!!---

## 3.6.2.5 :SLEW &lt;numeric\_value&gt;

SOURce:CURRent:SLEW

This command sets the current slew for a current controlled instrument. The numeric value parameter is the current slew rate value in amps-per-microseconds from 0 to 1.

## 3.6.2.6 :COEFFicient &lt;numeric\_value&gt;, &lt;numeric\_value&gt;

SOURce:CURRent:COEFFicient

This command is used to specify the current control coefficients, of a current controlled instrument. The first numeric value parameter specifies the coefficient to be changed, according to the following list, while the second numeric value parameter specifies the level the selected coefficient is to be changed to.

- 0. Saturation Minimum
- 1. Saturation Maximum
- 2. B0
- 3. B1
- 4. A1
- 5. B2
- 6. A2

When queried, the response is a list of comma separated numeric values in the following order:

Saturation-Minimum-value, Saturation-Maximum-value, B0-value, B1-value, A1-value, B2-value, A2-value

### 3.6.3 :FREQuency

SOURce:FREQuency

The FREQuency subsystem controls the frequency characteristics of an instrument source that produces a periodic signal - such as the sine wave produced by the AC stage.

#### 3.6.3.1 [:FIXed] <numeric\_value>

SOURce:FREQuency:FIXed

This command is used to set the source frequency. The numeric value parameter should be the required frequency in hertz, from 0 to 1000.

#### 3.6.3.2 :GAIN <numeric\_value>

SOURce:FREQuency:GAIN

This command is used to set the frequency gain. The numeric value parameter should vary from 0 to 1

#### 3.6.3.3 :OFFSet <numeric\_value>

SOURce:FREQuency:OFFSet

This command sets the frequency offset. The numeric value parameter should vary between -0.5 and +0.5

### 3.6.4 :TEMPerature

SOURce:TEMPerature

The temperature subsystem controls the temperature of the selected logical instrument. Note that not all instruments have temperature control functionality.

#### 3.6.4.1 [:PROTect] <numeric\_value>

SOURce:TEMPerature:PROTect

This command sets the temperature protection level. This is the level at which the over-temperature protection is triggered if the temperature rises above it. The numeric value parameter is the level to be set in degrees Celsius, between 0 and 150 °C.

## 3.7 MEASure Subsystem

The MEASure subsystem allows the user to take measurements of certain aspects of the selected instruments operation.

### 3.7.1 :VOLTage?

MEASure:VOLTage? This query responds with a numeric value of the current voltage reading from the selected instrument in volts.

### 3.7.2 :CURRent?

MEASure:CURRent? This query responds with a numeric value of the current electrical current reading from the selected instrument in amps.

### 3.7.3 :TEMPerature?

MEASure:TEMPerature? This query responds with a numeric value of the current temperature reading from the selected instrument in degrees Celsius.

## 3.8 STATus Subsystem

### 3.8.1 :OPERation

### 3.8.2 :PRESet

### 3.8.3 :QUEStionable

## 3.9 SYSTem

The SYSTem subsystem collects the functions that are not related to instrument performance.

### 3.9.1 :COMMs

SYSTem:COMMs

This subsystem controls some aspects of the serial communications (SCI) interface.

#### 3.9.1.1 :SBITs <numeric\_value>

SYSTem:COMMs:SBITs

This command sets the number of stop bits used in the SCI frame. The numeric value parameter sets the number of stop bits and may be either 1 or 2. The default value is 1.

#### 3.9.1.2 :PARity <numeric\_value>

SYSTem:COMMs:PARity

This command sets what parity is used in the SCI frames. The default selection is for no parity. The numeric value parameter sets the parity in use and corresponds to parity settings as described by the following list:

- 0. No parity
- 1. Odd parity
- 2. Even parity

#### 3.9.1.3 :DBITs <numeric\_value>

SYSTem:COMMs:DBITs

This command sets how many data bits are in the SCI frame. The numeric value parameter sets the number of data bits and may vary from 1 to 8.

### 3.9.1.4 :BAUD <numeric\_value>

SYSTem:COMMs:BAUD

This command sets the baud rate used by the SCI interface. The numeric value parameter sets the baud rate. The allowed parameters are any of the standard baud rates which are listed in the table below. Acceptance of a baud rate does not indicate that the device is capable of use such a baud rate. The default baud rate is 9600.

Accepted Baud Rates	Accepted Baud Rates	Accepted Baud Rates
110	9600	57600
300	14400	115200
600	19200	230400
1200	28800	460800
2400	38400	921600
4800	56000	

### 3.9.1.5 :ECHO <Boolean>

SYSTem:COMMs:ECHO

This command enables or disables the echo functionality of the SCI interface. If the boolean parameter is ON the interface echos any data it receives without passing that data on to any other part of the device. The default setting is OFF.

## Chapter 4

# Data Structure Documentation

### 4.1 acStageNets Struct Reference

```
#include <MacroNets.h>
```

#### Data Fields

- volatile int32 [vRefNet](#)
- volatile int32 [vFdbkNet](#)
- volatile int32 [iFdbkNet](#)
- volatile int32 [iRefNet](#)
- volatile int32 [iFiltOutNet](#)

#### 4.1.1 Detailed Description

A structure that contains the nets used in the AC stage.

#### 4.1.2 Field Documentation

##### 4.1.2.1 volatile int32 acStageNets::iFdbkNet

Current feedback net (IQ24).

##### 4.1.2.2 volatile int32 acStageNets::iFiltOutNet

Current IIR filter control law output net (IQ24).

##### 4.1.2.3 volatile int32 acStageNets::iRefNet

IIR filter control law current reference net (IQ24).

##### 4.1.2.4 volatile int32 acStageNets::vFdbkNet

Voltage feedback net (IQ24).

#### 4.1.2.5 volatile int32 acStageNets::vRefNet

IIR filter control law voltage reference net (IQ24).

The documentation for this struct was generated from the following file:

- [MacroNets.h](#)

## 4.2 acStageSettings Struct Reference

```
#include <MacroNets.h>
```

### Data Fields

- int32 [gainRate](#)
- int32 [gainTarget](#)
- int32 [ocpLevel](#)
- int32 [ovpLevel](#)
- int32 [otpLevel](#)
- int16 [iMaxRms](#)
- int16 [vMaxRms](#)
- int16 [iScale](#)
- int16 [vScale](#)
- int16 [vGainLmt](#)
- [slaveMode](#) mode
- Uint16 [acFrequency](#)
- Uint16 [enable](#)

### 4.2.1 Detailed Description

A structure that contains the settings that pertain to the AC stage.

### 4.2.2 Field Documentation

#### 4.2.2.1 Uint16 acStageSettings::acFrequency

Sine signal generator frequency setting (Hz).

#### 4.2.2.2 Uint16 acStageSettings::enable

Channel enable status {FALSE, TRUE}.

#### 4.2.2.3 int32 acStageSettings::gainRate

Reference sine gain rate (IQ24).

#### 4.2.2.4 int32 acStageSettings::gainTarget

Reference sine gain target (IQ24).



**4.2.2.5 int16 acStageSettings::iMaxRms**

Maximum RMS current setting limit (SQ10).

**4.2.2.6 int16 acStageSettings::iScale**

Current scaling setting in volts-per-amp for scaling between a voltage level measured by an ADC to a real current value (SQ14).

**4.2.2.7 slaveMode acStageSettings::mode**

Master or slave mode select.

**4.2.2.8 int32 acStageSettings::ocpLevel**

Normalised OCP limit (IQ24).

**4.2.2.9 int32 acStageSettings::otpLevel**

OTP limit in ° C (SQ7).

**4.2.2.10 int32 acStageSettings::ovpLevel**

Normalised OVP limit (IQ24).

**4.2.2.11 int16 acStageSettings::vGainLmt**

Sine signal generator voltage gain limit (SQ14).

**4.2.2.12 int16 acStageSettings::vMaxRms**

Maximum RMS voltage setting limit (SQ10).

**4.2.2.13 int16 acStageSettings::vScale**

Voltage scaling setting in volts-per-volts for scaling between a voltage level measured by an ADC to a real voltage value (SQ14).

The documentation for this struct was generated from the following file:

- [MacroNets.h](#)

## 4.3 extDeviceSettings Struct Reference

```
#include <MacroNets.h>
```

## Data Fields

- int32 [ext1OtpLevel](#)
- int32 [ext2OtpLevel](#)
- Uint16 [extFanEnable](#)
- Uint16 [extPsuEnable](#)

### 4.3.1 Detailed Description

A structure that contains the settings that pertain to external devices.

### 4.3.2 Field Documentation

#### 4.3.2.1 int32 extDeviceSettings::ext1OtpLevel

External 1 OTP limit in ° C (SQ7).

#### 4.3.2.2 int32 extDeviceSettings::ext2OtpLevel

External 2 OTP limit in ° C (SQ7).

#### 4.3.2.3 Uint16 extDeviceSettings::extFanEnable

External fan enable status {TRUE | FALSE}.

#### 4.3.2.4 Uint16 extDeviceSettings::extPsuEnable

External PSU enable status {TRUE | FALSE}.

The documentation for this struct was generated from the following file:

- [MacroNets.h](#)

## 4.4 i2cMsg Struct Reference

```
#include <I2c.h>
```

## Data Fields

- volatile Uint16 [msgStatus](#)
- Uint16 [slaveAddress](#)
- Uint16 [numOfBytes](#)
- Uint16 [numSlavePtrBytes](#)
- Uint16 [slavePtrAddrHigh](#)
- Uint16 [slavePtrAddrLow](#)
- Uint16 [msgBuffer](#) [I2C\_MAX\_BUFFER\_SIZE]

### 4.4.1 Detailed Description

The structure used to contain all settings and values relevant to a particular I2C message.

### 4.4.2 Field Documentation

#### 4.4.2.1 `Uint16 i2cMsg::msgBuffer[I2C_MAX_BUFFER_SIZE]`

A buffer array for message data. The maximum buffer size, `MAX_BUFFER_SIZE`, is 4 due to the FIFO's size.

#### 4.4.2.2 `volatile Uint16 i2cMsg::msgStatus`

Indicates which state the message is in.

#### 4.4.2.3 `Uint16 i2cMsg::numOfBytes`

The number of valid bytes in (or to be put in `msgBuffer`).

#### 4.4.2.4 `Uint16 i2cMsg::numSlavePtrBytes`

The number of slave register pointer address bytes.

#### 4.4.2.5 `Uint16 i2cMsg::slaveAddress`

The slave device I2C address this message is intended for.

#### 4.4.2.6 `Uint16 i2cMsg::slavePtrAddrHigh`

The slave register pointer high byte.

#### 4.4.2.7 `Uint16 i2cMsg::slavePtrAddrLow`

The slave register pointer low byte.

The documentation for this struct was generated from the following file:

- [I2c.h](#)

## 4.5 loadStageNets Struct Reference

```
#include <MacroNets.h>
```

### Data Fields

- `volatile int32 iFdbkNet`
- `volatile int32 vFdbkNet`
- `volatile int32 iRefNet`
- `volatile int32 iFiltOutNet`

### 4.5.1 Detailed Description

A structure that contains the nets used in a single load stage.

## 4.5.2 Field Documentation

### 4.5.2.1 volatile int32 loadStageNets::iFdbkNet

Current feedback net (IQ24).

### 4.5.2.2 volatile int32 loadStageNets::iFiltOutNet

IIR filter control law output net (IQ24).

### 4.5.2.3 volatile int32 loadStageNets::iRefNet

IIR filter control law current reference net (IQ24).

### 4.5.2.4 volatile int32 loadStageNets::vFdbkNet

Voltage feedback net (IQ24).

The documentation for this struct was generated from the following file:

- [MacroNets.h](#)

## 4.6 loadStageSettings Struct Reference

```
#include <MacroNets.h>
```

### Data Fields

- int32 [slewRate](#)
- int32 [slewTarget](#)
- int32 [ocpLevel](#)
- int32 [ovpLevel](#)
- int32 [oppLevel](#)
- int32 [otpLevel](#)
- int16 [iMax](#)
- int16 [vMax](#)
- int16 [iScale](#)
- int16 [vScale](#)
- Uint16 [enable](#)

### 4.6.1 Detailed Description

A structure that contains the settings that pertain to a single load stage.

## 4.6.2 Field Documentation

### 4.6.2.1 Uint16 loadStageSettings::enable

Channel enable status {FALSE, TRUE}.

#### 4.6.2.2 int16 loadStageSettings::iMax

Maximum DC current setting limit (SQ10).

#### 4.6.2.3 int16 loadStageSettings::iScale

Current scaling setting in volts-per-amp for scaling between a voltage level measured by an ADC to a real current value (SQ14).

#### 4.6.2.4 int32 loadStageSettings::ocpLevel

Normalised OCP limit (IQ24).

#### 4.6.2.5 int32 loadStageSettings::oppLevel

Normalised OPP limit (IQ24).

#### 4.6.2.6 int32 loadStageSettings::otpLevel

OTP limit in ° C (SQ7).

#### 4.6.2.7 int32 loadStageSettings::ovpLevel

Normalised OVP limit (IQ24).

#### 4.6.2.8 int32 loadStageSettings::slewRate

IIR filter control law reference slew rate (IQ24).

#### 4.6.2.9 int32 loadStageSettings::slewTarget

IIR filter control law reference slew target (IQ24).

#### 4.6.2.10 int16 loadStageSettings::vMax

Maximum DC voltage setting limit (SQ10).

#### 4.6.2.11 int16 loadStageSettings::vScale

Voltage scaling setting in volts-per-volts for scaling between a voltage level measured by an ADC to a real voltage value (SQ14).

The documentation for this struct was generated from the following file:

- [MacroNets.h](#)

## 4.7 xfmrStageNets Struct Reference

```
#include <MacroNets.h>
```

## Data Fields

- volatile int32 [iSnsNet](#)
- volatile int32 [midVSnsNet](#)
- volatile int32 [hvVSnsNet](#)
- volatile int32 [pwmDutyNet](#)

### 4.7.1 Detailed Description

A structure that contains the nets used in the transformer stage.

### 4.7.2 Field Documentation

#### 4.7.2.1 volatile int32 xfmrStageNets::hvVSnsNet

DC HV voltage sense net (IQ24).

#### 4.7.2.2 volatile int32 xfmrStageNets::iSnsNet

Current sense net (IQ24).

#### 4.7.2.3 volatile int32 xfmrStageNets::midVSnsNet

DC mid voltage sense net (IQ24).

#### 4.7.2.4 volatile int32 xfmrStageNets::pwmDutyNet

Transformer PWM duty net (IQ24).

The documentation for this struct was generated from the following file:

- [MacroNets.h](#)

## 4.8 xfmrStageSettings Struct Reference

```
#include <MacroNets.h>
```

## Data Fields

- int32 [ocpLevel](#)
- int32 [hvOvpLevel](#)
- int32 [otpLevel](#)
- int16 [iMax](#)
- int16 [midVMax](#)
- int16 [hvVMax](#)
- int16 [iScale](#)
- int16 [midVScale](#)
- int16 [hvVScale](#)
- Uint16 [enable](#)

### 4.8.1 Detailed Description

A structure that contains the settings that pertain to the transformer stage.

### 4.8.2 Field Documentation

#### 4.8.2.1 `uint16 xfmrStageSettings::enable`

Channel enable status {FALSE, TRUE}.

#### 4.8.2.2 `int32 xfmrStageSettings::hvOvpLevel`

Normalised DC HV OVP limit (IQ24).

#### 4.8.2.3 `int16 xfmrStageSettings::hvVMax`

Maximum DC HV voltage setting limit (SQ10).

#### 4.8.2.4 `int16 xfmrStageSettings::hvVScale`

DC HV voltage scaling setting in volts-per-volts for scaling between a voltage level measured by an ADC to a real voltage value (SQ14).

#### 4.8.2.5 `int16 xfmrStageSettings::iMax`

Maximum DC current setting limit (SQ10).

#### 4.8.2.6 `int16 xfmrStageSettings::iScale`

Current scaling setting in volts-per-amp for scaling between a voltage level measured by an ADC to a real current value (SQ14).

#### 4.8.2.7 `int16 xfmrStageSettings::midVMax`

Maximum DC mid voltage setting limit (SQ10).

#### 4.8.2.8 `int16 xfmrStageSettings::midVScale`

DC mid voltage scaling setting in volts-per-volts for scaling between a voltage level measured by an ADC to a real voltage value (SQ14).

#### 4.8.2.9 `int32 xfmrStageSettings::ocpLevel`

Normalised OCP limit (IQ24).

#### 4.8.2.10 `int32 xfmrStageSettings::otpLevel`

OTP limit in ° C (SQ7).

The documentation for this struct was generated from the following file:

- [MacroNets.h](#)



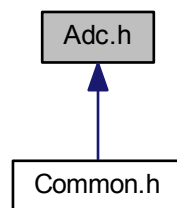
## Chapter 5

# File Documentation

### 5.1 Adc.h File Reference

ADC functions.

This graph shows which files directly or indirectly include this file:



#### Functions

- void [initAdc](#) (void)
- Uint16 [getLoadCurrent](#) ([loadStage](#) load, float32 \*dcCurrent)
- Uint16 [getLoadVoltage](#) ([loadStage](#) load, float32 \*dcVoltage)
- Uint16 [getDcMidCurrent](#) (float32 \*dcCurrent)
- Uint16 [getDcMidVoltage](#) (float32 \*dcVoltage)
- Uint16 [getDcHvVoltage](#) (float32 \*dcVoltage)

#### Variables

- volatile int32 \* [ADCDRV\\_1ch\\_Rlt1](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt2](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt3](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt4](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt5](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt6](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt7](#)

- volatile int32 \* [ADCDRV\\_1ch\\_Rlt8](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt9](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt10](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt11](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt12](#)
- volatile int32 \* [ADCDRV\\_1ch\\_Rlt13](#)

### 5.1.1 Detailed Description

ADC functions.

### 5.1.2 Function Documentation

#### 5.1.2.1 Uint16 getDcHvVoltage ( float32 \* *dcVoltage* )

Queries the most recent voltage reading from the DC HV's associated ADC.

Parameters

out	<i>dcVoltage</i>	Address of the memory location at which to place the query result (volts).
-----	------------------	--

Returns

Error status.

#### 5.1.2.2 Uint16 getDcMidCurrent ( float32 \* *dcCurrent* )

Queries the most recent current reading from the DC Mid's associated ADC.

Parameters

out	<i>dcCurrent</i>	Address of the memory location at which to place the query result (amps).
-----	------------------	---

Returns

Error status.

#### 5.1.2.3 Uint16 getDcMidVoltage ( float32 \* *dcVoltage* )

Queries the most recent voltage reading from the DC Mid's associated ADC.

Parameters

out	<i>dcVoltage</i>	Address of the memory location at which to place the query result (volts).
-----	------------------	--

Returns

Error status.

#### 5.1.2.4 Uint16 getLoadCurrent ( loadStage *load*, float32 \* *dcCurrent* )

Queries the most recent current reading from the specified load's associated ADC.

**Parameters**

in	<i>load</i>	Specifies the load on which the reading is to be queried.
out	<i>dcCurrent</i>	Address of the memory location at which to place the query result (amps).

**Returns**

Error status.

**5.1.2.5 Uint16 getLoadVoltage ( loadStage *load*, float32 \* *dcVoltage* )**

Queries the most recent voltage reading from the specified load's associated ADC.

**Parameters**

in	<i>load</i>	Specifies the load on which the reading is to be queried.
out	<i>dcVoltage</i>	Address of the memory location at which to place the query result (volts).

**Returns**

Error status.

**5.1.2.6 void initAdc ( void )**

Configures the ADC's SOC's then calls pwmSocConfigure().

- SHOULD be run after initPWM().
- SHOULD be run before DPL\_INIT().

**5.1.3 Variable Documentation****5.1.3.1 volatile int32\* ADCDRV\_1ch\_Rlt1**

Channel 0 current sense ADC terminal pointer.

**5.1.3.2 volatile int32\* ADCDRV\_1ch\_Rlt10**

Channel 3 voltage sense ADC terminal pointer.

**5.1.3.3 volatile int32\* ADCDRV\_1ch\_Rlt11**

Interboost voltage sense ADC terminal pointer.

**5.1.3.4 volatile int32\* ADCDRV\_1ch\_Rlt12**

AC stage voltage sense ADC terminal pointer.

**5.1.3.5 volatile int32\* ADCDRV\_1ch\_Rlt13**

VMid voltage sense ADC terminal pointer.

#### 5.1.3.6 `volatile int32* ADCDRV_1ch_Rlt2`

Channel 1 current sense ADC terminal pointer.

#### 5.1.3.7 `volatile int32* ADCDRV_1ch_Rlt3`

Channel 2 current sense ADC terminal pointer.

#### 5.1.3.8 `volatile int32* ADCDRV_1ch_Rlt4`

Channel 3 current sense ADC terminal pointer.

#### 5.1.3.9 `volatile int32* ADCDRV_1ch_Rlt5`

Interboost current sense ADC terminal pointer.

#### 5.1.3.10 `volatile int32* ADCDRV_1ch_Rlt6`

AC stage current sense ADC terminal pointer.

#### 5.1.3.11 `volatile int32* ADCDRV_1ch_Rlt7`

Channel 0 voltage sense ADC terminal pointer.

#### 5.1.3.12 `volatile int32* ADCDRV_1ch_Rlt8`

Channel 1 voltage sense ADC terminal pointer.

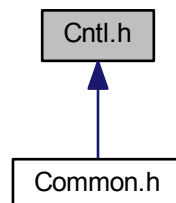
#### 5.1.3.13 `volatile int32* ADCDRV_1ch_Rlt9`

Channel 2 voltage sense ADC terminal pointer.

## 5.2 Cntl.h File Reference

DPLib CNTL Macro related helper functions.

This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef enum [coefNum](#) [cfType](#)

## Enumerations

- enum [coefNum](#) { ,  
[cMin](#) = firstCoef, [cMax](#), [cB0](#), [cB1](#),  
[cA1](#), [cB2](#), [cA2](#) }

## Functions

- void [initCoefs](#) (void)
- Uint16 [setLoadICoef](#) (loadStage load, [cfType](#) coef, float32 value)
- Uint16 [getLoadICoef](#) (loadStage load, [cfType](#) coef, float32 \*value)
- Uint16 [setAclCoef](#) ([cfType](#) coef, float32 val)
- Uint16 [getAclCoef](#) ([cfType](#) coef, float32 \*value)
- Uint16 [setAcVCoef](#) ([cfType](#) coef, float32 value)
- Uint16 [getAcVCoef](#) ([cfType](#) coef, float32 \*value)

## Variables

- struct CNTL\_2P2Z\_CoefStruct [loadICoefs](#) [numberOfLoads]
- struct CNTL\_2P2Z\_CoefStruct [aclCoefs](#)
- struct CNTL\_2P2Z\_CoefStruct [acVCoefs](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Coef1](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Coef2](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Coef3](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Coef4](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Fdbk1](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Fdbk2](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Fdbk3](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Fdbk4](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Out1](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Out2](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Out3](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Out4](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Ref1](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Ref2](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Ref3](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Ref4](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Coef5](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Fdbk5](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Out5](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Ref5](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Coef6](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Fdbk6](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Out6](#)
- volatile int32 \* [CNTL\\_2P2Z\\_Ref6](#)

### 5.2.1 Detailed Description

DPLib CNTL Macro related helper functions.

## 5.2.2 Typedef Documentation

### 5.2.2.1 typedef enum coefNum cfType

A type that allows a reference to a CNTL coefficient.

## 5.2.3 Enumeration Type Documentation

### 5.2.3.1 enum coefNum

CNTL Coefficient references

Enumerator

**cMin** Saturation minimum reference.

**cMax** Saturation maximum reference.

**cB0** B0 coefficient reference.

**cB1** B1 coefficient reference.

**cA1** A1 coefficient reference.

**cB2** B2 coefficient reference.

**cA2** A2 coefficient reference.

## 5.2.4 Function Documentation

### 5.2.4.1 Uint16 getAclCoef ( cfType coef, float32 \* value )

Queries the specified IIR filter control law coefficient for the AC current control.

Parameters

in	coef	Specifies the coefficient to be queried [cMin, cA3].
out	value	Address of the memory location at which to place the query result.

Returns

Error status.

### 5.2.4.2 Uint16 getAcVCoef ( cfType coef, float32 \* value )

Queries the specified IIR filter control law coefficient for the AC voltage control.

Parameters

in	coef	Specifies the coefficient to be queried [cMin, cA3].
out	value	Address of the memory location at which to place the query result.

Returns

Error status.

### 5.2.4.3 Uint16 getLoadICoef ( loadStage load, cfType coef, float32 \* value )

Queries the specified IIR filter control law coefficient for a specified load's current control.

**Parameters**

in	<i>load</i>	Specifies the load on which the setting is to be queried [0, NUM_CHNLS).
in	<i>coef</i>	Specifies the coefficient to be queried [cMin, cA3).
out	<i>value</i>	Address of the memory location at which to place the query result.

**Returns**

Error status.

**5.2.4.4 void initCoefs ( void )**

Initialises all IIR filter control law coefficients.

**Warning**

This function must be called before the control macros are used.

**5.2.4.5 Uint16 setAclCoef ( cfType coef, float32 val )**

Sets the specified IIR filter control law coefficient for the AC current control.

**Parameters**

in	<i>coef</i>	Specifies the coefficient to be set [cMin, cA3).
in	<i>val</i>	Specifies the coefficient value to be applied. Should be between the minimum and maximum values for the specific coefficient as defined by cfLmts[]].

**Returns**

Error status.

**5.2.4.6 Uint16 setAcVCoef ( cfType coef, float32 value )**

Sets the specified IIR filter control law coefficient for the AC voltage control.

- The actual setting in use is not updated until AFTER cntlUpdateCoefs() has been called.

**Parameters**

in	<i>coef</i>	Specifies the coefficient to be set [cMin, cA3).
in	<i>value</i>	Specifies the coefficient value to be applied. Should be between the minimum and maximum values for the specific coefficient as defined by cfLmts[]].

**Returns**

Error status.

**5.2.4.7 Uint16 setLoadICoef ( loadStage load, cfType coef, float32 value )**

Sets the specified IIR filter control law coefficient for a specified load's current control.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load number the setting is to be applied to [0, NUM_CHNLS).
<i>in</i>	<i>coef</i>	Specifies the coefficient to be set [cMin, cA3).
<i>in</i>	<i>value</i>	Specifies the coefficient value to be applied. Should be between the minimum and maximum values for the specific coefficient as defined by cfLmts[]].

**Returns**

Error status.

**5.2.5 Variable Documentation****5.2.5.1 struct CNTL\_2P2Z\_CoefStruct acLCoefs**

Structure that holds the AC I 2-pole 2-zero IIR filter control law coefficients currently in use.

**5.2.5.2 struct CNTL\_2P2Z\_CoefStruct acVCoefs**

Structure that holds the AC V 2-pole 2-zero IIR filter control law coefficients currently in use.

**5.2.5.3 volatile int32\* CNTL\_2P2Z\_Coef1**

Load 1 IIR filter control law coefficient terminal pointer.

**5.2.5.4 volatile int32\* CNTL\_2P2Z\_Coef2**

Load 2 IIR filter control law coefficient terminal pointer.

**5.2.5.5 volatile int32\* CNTL\_2P2Z\_Coef3**

Load 3 IIR filter control law coefficient terminal pointer.

**5.2.5.6 volatile int32\* CNTL\_2P2Z\_Coef4**

Load 4 IIR filter control law coefficient terminal pointer.

**5.2.5.7 volatile int32\* CNTL\_2P2Z\_Coef5**

AC stage I IIR filter control law coefficient terminal pointer.

**5.2.5.8 volatile int32\* CNTL\_2P2Z\_Coef6**

AC stage V IIR filter control law coefficient terminal pointer.

**5.2.5.9 volatile int32\* CNTL\_2P2Z\_Fdbk1**

Load 1 IIR filter control law feedback terminal pointer.



**5.2.5.10 volatile int32\* CNTL\_2P2Z\_Fdbk2**

Load 2 IIR filter control law feedback terminal pointer.

**5.2.5.11 volatile int32\* CNTL\_2P2Z\_Fdbk3**

Load 3 IIR filter control law feedback terminal pointer.

**5.2.5.12 volatile int32\* CNTL\_2P2Z\_Fdbk4**

Load 4 IIR filter control law feedback terminal pointer.

**5.2.5.13 volatile int32\* CNTL\_2P2Z\_Fdbk5**

AC stage I IIR filter control law feedback terminal pointer.

**5.2.5.14 volatile int32\* CNTL\_2P2Z\_Fdbk6**

AC stage V IIR filter control law feedback terminal pointer.

**5.2.5.15 volatile int32\* CNTL\_2P2Z\_Out1**

Load 1 IIR filter control law output terminal pointer.

**5.2.5.16 volatile int32\* CNTL\_2P2Z\_Out2**

Load 2 IIR filter control law output terminal pointer.

**5.2.5.17 volatile int32\* CNTL\_2P2Z\_Out3**

Load 3 IIR filter control law output terminal pointer.

**5.2.5.18 volatile int32\* CNTL\_2P2Z\_Out4**

Load 4 IIR filter control law output terminal pointer.

**5.2.5.19 volatile int32\* CNTL\_2P2Z\_Out5**

AC stage I IIR filter control law output terminal pointer.

**5.2.5.20 volatile int32\* CNTL\_2P2Z\_Out6**

AC stage V IIR filter control law output terminal pointer.

**5.2.5.21 volatile int32\* CNTL\_2P2Z\_Ref1**

Load 1 IIR filter control law reference terminal pointer.

**5.2.5.22 volatile int32\* CNTL\_2P2Z\_Ref2**

Load 2 IIR filter control law reference terminal pointer.

**5.2.5.23 volatile int32\* CNTL\_2P2Z\_Ref3**

Load 3 IIR filter control law reference terminal pointer.

**5.2.5.24 volatile int32\* CNTL\_2P2Z\_Ref4**

Load 4 IIR filter control law reference terminal pointer.

**5.2.5.25 volatile int32\* CNTL\_2P2Z\_Ref5**

AC stage I IIR filter control law reference terminal pointer.

**5.2.5.26 volatile int32\* CNTL\_2P2Z\_Ref6**

AC stage V IIR filter control law reference terminal pointer.

**5.2.5.27 struct CNTL\_2P2Z\_CoefStruct loadICoefs[numberOfLoads]**

Array of structures that hold the load 2-pole 2-zero IIR filter control law coefficients currently in use.

## 5.3 Common.h File Reference

Common include file for the project.

```

#include "Settings.h"
#include "PeripheralHeaderIncludes.h"
#include "DSP2802x_EPWM_defines.h"
#include "slaveMode.h"
#include "IQMathLib.h"
#include "SQMath.h"
#include "DPLib.h"
#include "sgen.h"
#include "I2c.h"
#include "Sci.h"
#include "Spi.h"
#include "MacroNets.h"
#include "Timers.h"
#include "Adc.h"
#include "Comparator.h"
#include "Pwm.h"
#include "Cntl.h"
#include "Ocp.h"
#include "Ovp.h"
#include "Opp.h"
#include "Otp.h"
#include "SlewControl.h"
#include "SineGen.h"
#include "tmp.h"
#include "EnableCtrl.h"
#include "StateMachine.h"
#include "../SCPI_Build/SCPI_Build/scpi/scpi.h"
#include "SCPI_specificCmds.h"

```

Include dependency graph for Common.h:



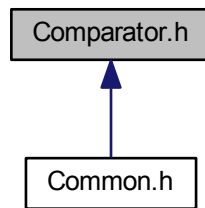
### 5.3.1 Detailed Description

Common include file for the project. All other header files used should be included within this file and this file should then be used to include them in the required source files.

## 5.4 Comparator.h File Reference

Comparator, DAC and trip zone functions.

This graph shows which files directly or indirectly include this file:



## Functions

- void [initTripZone](#) (void)
- void [initDcComparator](#) (void)
- void [rstDcTripzone](#) (void)
- void [initAcComparator](#) (void)
- Uint16 [setAcDac](#) (float32 level)
- Uint16 [getAcDac](#) (float32 \*level)
- void [rstAcTripzone](#) (void)

### 5.4.1 Detailed Description

Comparator, DAC and trip zone functions. Requires the modification of the COMP\_REGS struct of the DSP2802x\_ \_GlobalVariableDefs in the file DSP2802x\_Comp.h, to allow use of the DACCTL and ramp-related register unions. Use the equivalent file from the f2903x includes for reference.

### 5.4.2 Function Documentation

#### 5.4.2.1 Uint16 [getAcDac](#) ( float32 \* *level* )

Queries the output level setting of the DAC on the inverting input of the DC comparator. iScale MUST be set before use.

##### Parameters

<i>out</i>	<i>level</i>	Pointer to location at which to place the query result (amps).
------------	--------------	--

##### Returns

Error status.

#### 5.4.2.2 void [initAcComparator](#) ( void )

Initialises the AC comparator (COMP 1) using an internal DAC at the inverting input.

- SHOULD be called AFTER [adcSocCnf](#)().
- SHOULD be called BEFORE PWMS (SYNC) are started.
- SHOULD be called BEFORE [pwmTZConfigure](#)().

#### 5.4.2.3 void initDcComparator ( void )

Initialises the DC comparator (COMP 2) using an internal DAC at the inverting input.

- SHOULD be called AFTER adcSocCnf().
- SHOULD be called BEFORE PWMS (SYNC) are started.
- SHOULD be called BEFORE pwmTZConfigure().
- midVScale should be set before use.

#### 5.4.2.4 void initTripZone ( void )

Configures PWM trip zones for use. Requires the comparator and DAC to be configured

See Also

[adc.h](#)

#### 5.4.2.5 void rstAcTripzone ( void )

Resets the AC trip zone after an AC comparator event.

#### 5.4.2.6 void rstDcTripzone ( void )

Resets the DC trip zone after an DC comparator event.

#### 5.4.2.7 Uint16 setAcDac ( float32 level )

Sets the output level of the DAC on the inverting input of the AC comparator. The voltage and iScale MUST be set before use.

Parameters

<i>in</i>	<i>level</i>	Specifies the value of the level setting to be applied (amps).
-----------	--------------	--

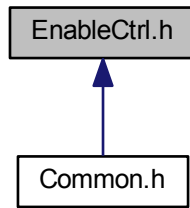
Returns

Error status.

## 5.5 EnableCtrl.h File Reference

Functions for enabling and disabling circuit sections via I2C.

This graph shows which files directly or indirectly include this file:



## Macros

- `#define IOE_SW4A_STATE 0x00`
- `#define IOE_SW4B_STATE 0x00`
- `#define IOE_I2C_BASE_ADDR 0x20`
- `#define IOE_IODIR_ADDR 0x00`
- `#define IOE_IPOL_ADDR 0x01`
- `#define IOE_GPINTEN_ADDR 0x02`
- `#define IOE_DEFVAL_ADDR 0x03`
- `#define IOE_INTCON_ADDR 0x04`
- `#define IOE_IOCON_ADDR 0x05`
- `#define IOE_GPPU_ADDR 0x06`
- `#define IOE_INTF_ADDR 0x07`
- `#define IOE_INTCAP_ADDR 0x08`
- `#define IOE_GPIO_ADDR 0x09`
- `#define IOE_OLAT_ADDR 0x0A`
- `#define IOE_MAX_CHNL 0x08`
- `#define IOE_I2C_ADDR (IOE_I2C_BASE_ADDR | (IOE_SW4A_STATE << 2)) | (IOE_SW4B_STATE << 3)`
- `#define ALL_DISABLED_WORD`

## Typedefs

- `typedef enum ecSection circuitSection`

## Enumerations

- `enum ecSection`

## Functions

- `Uint16 initEnableControl (void)`
- `void resetEnableControl (void)`
- `Uint16 enableCircuit (circuitSection section)`
- `Uint16 disableCircuit (circuitSection section)`
- `Uint16 getCurrentState (circuitSection section, Uint16 *state)`

### 5.5.1 Detailed Description

Functions for enabling and disabling circuit sections via I2C. Circuit sections are controlled via an external I/O expander (MCP23008) that is connected to the I2C bus at address 0100xx0 where 'xx' is determined by the state of switch 4 (SW4) on the Ctrl PCB, which the user should record in the macros IOE\_SW4A\_STATE and IOE\_SW4B\_STATE.

After initialisation or reset all sections default to disabled.

#### Warning

Before any enable control functions can be used the I2C peripheral **MUST** be initialised, followed by enable control initialisation, both of which will require global interrupts to be enabled.

#### See Also

[initI2c\(\)](#)

ADS7830 Ch #	Signal	Enable bit#	Enable state
GP0	CHAN1EN	0	0
GP1	CHAN2EN	1	0
GP2	CHAN3EN	2	0
GP3	CHAN4EN	3	0
GP4	HVENABLE	4	0
GP5	ACENABLE	5	1
GP6	PSUENABLE	6	1
GP7	EXTFANEN	7	1

### 5.5.2 Macro Definition Documentation

#### 5.5.2.1 #define ALL\_DISABLED\_WORD

##### Value:

```
(~(0 | (CHAN1_EN_STATE << chan1) | (CHAN2_EN_STATE << chan2) | (CHAN3_EN_STATE << chan3) \
      | (CHAN4_EN_STATE << chan4) | (XFMR_EN_STATE << xfmrCct) | (AC_EN_STATE <<
      acCct) \
      | (PSU_EN_STATE << psu) | (EXTFAN_EN_STATE << fan))) & 0x00FF
```

Uses the defined states to define a word for the case where all circuits are disabled

#### 5.5.2.2 #define IOE\_DEFVAL\_ADDR 0x03

MCP23008 I/O expander default value register address.

#### 5.5.2.3 #define IOE\_GPINTEN\_ADDR 0x02

MCP23008 I/O expander interrupt on change enable register address.

#### 5.5.2.4 #define IOE\_GPIO\_ADDR 0x09

MCP23008 I/O expander GPIO port register address.

#### 5.5.2.5 #define IOE\_GPPU\_ADDR 0x06

MCP23008 I/O expander pull-up resistor configuration register address.

5.5.2.6 **#define IOE\_I2C\_ADDR (IOE\_I2C\_BASE\_ADDR | (IOE\_SW4A\_STATE << 2) | (IOE\_SW4B\_STATE << 3))**

MCP23008 I/O expander complete I2C address.

5.5.2.7 **#define IOE\_I2C\_BASE\_ADDR 0x20**

MCP23008 I/O expander base I2C address.

5.5.2.8 **#define IOE\_INTCAP\_ADDR 0x08**

MCP23008 I/O expander interrupt capture register address.

5.5.2.9 **#define IOE\_INTCON\_ADDR 0x04**

MCP23008 I/O expander interrupt on change control register address.

5.5.2.10 **#define IOE\_INTF\_ADDR 0x07**

MCP23008 I/O expander interrupt flag register address.

5.5.2.11 **#define IOE\_IOCON\_ADDR 0x05**

MCP23008 I/O expander configuration register address.

5.5.2.12 **#define IOE\_IODIR\_ADDR 0x00**

MCP23008 I/O expander I/O direction register address.

5.5.2.13 **#define IOE\_IPOL\_ADDR 0x01**

MCP23008 I/O expander input polarity register address.

5.5.2.14 **#define IOE\_MAX\_CHNL 0x08**

MCP23008 I/O expander number of channels (0 - 7).

5.5.2.15 **#define IOE\_OLAT\_ADDR 0x0A**

MCP23008 I/O expander output latch register address.

5.5.2.16 **#define IOE\_SW4A\_STATE 0x00**

The state of switch 4a; ON = 0x01, OFF = 0x00.

5.5.2.17 **#define IOE\_SW4B\_STATE 0x00**

The state of switch 4b; ON = 0x01, OFF = 0x00.



### 5.5.3 Typedef Documentation

#### 5.5.3.1 typedef enum ecSection circuitSection

A type that allow specification of an enable control circuit section.

### 5.5.4 Enumeration Type Documentation

#### 5.5.4.1 enum ecSection

The possible enable control circuit section selections. Also defines bit order.

### 5.5.5 Function Documentation

#### 5.5.5.1 Uint16 disableCircuit ( circuitSection *section* )

Disables the specified circuit section enable signal.

##### Parameters

<i>in</i>	<i>section</i>	Specifies the circuit section that is to be disabled.
-----------	----------------	---

##### Returns

Error status.

#### 5.5.5.2 Uint16 enableCircuit ( circuitSection *section* )

Enables the specified circuit section enable signal.

##### Parameters

<i>in</i>	<i>section</i>	Specifies the circuit section that is to be enabled.
-----------	----------------	--

##### Returns

Error status.

#### 5.5.5.3 Uint16 getCurrentState ( circuitSection *section*, Uint16 \* *state* )

Gets the current state of the specified circuit section.

##### Parameters

<i>in</i>	<i>section</i>	Specifies the circuit section for which the state is to be queried.
<i>out</i>	<i>state</i>	Pointer to location at which to replace query result (Disabled = 0, enabled = !0).

##### Returns

Error status.

#### 5.5.5.4 Uint16 initEnableControl ( void )

Initialises the enable control interface. The I2C peripheral MUST be initialised before this function is used.

##### See Also

i2cInit()

##### Returns

Error status.

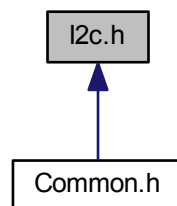
#### 5.5.5.5 void resetEnableControl ( void )

Resets and reinitialises the MCP23008 I/O Expander device.

## 5.6 I2c.h File Reference

Inter-integrated circuit communications functions.

This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [i2cMsg](#)

### Macros

- #define [I2C\\_MAX\\_BUFFER\\_SIZE](#) 0x04
- #define [I2C\\_MAX\\_PTR\\_SIZE](#) 0x02
  
- #define [I2C\\_CLR\\_AL\\_BIT](#) 0x0001
- #define [I2C\\_CLR\\_NACK\\_BIT](#) 0x0002
- #define [I2C\\_CLR\\_ARDY\\_BIT](#) 0x0004
- #define [I2C\\_CLR\\_RRDY\\_BIT](#) 0x0008
- #define [I2C\\_CLR\\_SCD\\_BIT](#) 0x0020
  
- #define [I2C\\_ARDY\\_ISRC](#) 0x0003

- `#define I2C_SCD_ISRC 0x0006`
- `#define I2C_MSGSTAT_INACTIVE 0x0000`
- `#define I2C_MSGSTAT_SEND_WITHSTOP 0x0010`
- `#define I2C_MSGSTAT_WRITE_BUSY 0x0011`
- `#define I2C_MSGSTAT_SEND_NOSTOP 0x0020`
- `#define I2C_MSGSTAT_SEND_NOSTOP_BUSY 0x0021`
- `#define I2C_MSGSTAT_RESTART 0x0022`
- `#define I2C_MSGSTAT_READ_BUSY 0x0023`

## Functions

- void `initI2c` (void)
- void `i2cPopMsg` (`i2cMsg` \*msg, Uint16 msgStatus, Uint16 slaveAddr, Uint16 numDataBytes, Uint16 numSlavePtrBytes, Uint16 slavePtrAddrHi, Uint16 slavePtrAddrLo)
- Uint16 `i2cWrite` (`i2cMsg` \*msg)
- Uint16 `i2cRead` (`i2cMsg` \*msg)

### 5.6.1 Detailed Description

Inter-integrated circuit communications functions.

#### Warning

The following bridge tracks should be cut on the TI C2000 LaunchPad XL PCB before I2C (or SPI) may be used:

Bridge to Cut	GPIO	PCB Pin
JP5	GPIO32	J2-6
JP7	GPIO33	J2-7
JP8	GPIO16	J6-7
JP10	GPIO17	J6-8

This should result in the following functionality:

Function	GPIO	PCB Pin
I2C_SDAA	GPIO32	J6-7
I2C_SCLA	GPIO33	J6-8
SPI_MOSI	GPIO16	J2-6
SPI_MISO	GPIO17	J2-7

The function `i2cInit()` MUST be called before any other public I2C function is used. This will clear any values already in the I2C registers.

Interrupts MUST be globally enabled for the functions `i2cWrite()` and `i2cRead()` to operate correctly.

#### See Also

[EnableCtrl.h](#)  
[Tmp.h](#)

### 5.6.2 Macro Definition Documentation

#### 5.6.2.1 `#define I2C_ARDY_ISRC 0x0003`

I2C Interrupt Sources Register access ready condition I2C interrupt source.

#### 5.6.2.2 `#define I2C_CLR_AL_BIT 0x0001`

I2C Status Clear Bits Arbitration lost status clear bit.

**5.6.2.3 #define I2C\_CLR\_ARDY\_BIT 0x0004**

Register access ready status clear bit.

**5.6.2.4 #define I2C\_CLR\_NACK\_BIT 0x0002**

NACK status clear bit.

**5.6.2.5 #define I2C\_CLR\_RRDY\_BIT 0x0008**

Receive data ready status clear bit.

**5.6.2.6 #define I2C\_CLR\_SCD\_BIT 0x0020**

Stop detected status clear bit.

**5.6.2.7 #define I2C\_MAX\_BUFFER\_SIZE 0x04**

Maximum I2C message buffer size in bytes, including slave register pointer bytes.

**5.6.2.8 #define I2C\_MAX\_PTR\_SIZE 0x02**

Maximum number of slave register pointer bytes.

**5.6.2.9 #define I2C\_MSGSTAT\_INACTIVE 0x0000**

I2C Message States Inactive I2C message state.

**5.6.2.10 #define I2C\_MSGSTAT\_READ\_BUSY 0x0023**

State indicating the I2C is busy with a read.

**5.6.2.11 #define I2C\_MSGSTAT\_RESTART 0x0022**

Transmit a master read with a restart.

**5.6.2.12 #define I2C\_MSGSTAT\_SEND\_NOSTOP 0x0020**

Transmit a write with no stop.

**5.6.2.13 #define I2C\_MSGSTAT\_SEND\_NOSTOP\_BUSY 0x0021**

State indicating the I2C is busy with a write with no stop.

**5.6.2.14 #define I2C\_MSGSTAT\_SEND\_WITHSTOP 0x0010**

Transmit a write with stop I2C message state.

#### 5.6.2.15 #define I2C\_MSGSTAT\_WRITE\_BUSY 0x0011

State indicating the I2C is busy with a write with a stop.

#### 5.6.2.16 #define I2C\_SCD\_ISRC 0x0006

Stop detected condition I2C interrupt source.

### 5.6.3 Function Documentation

#### 5.6.3.1 void i2cPopMsg ( i2cMsg \* msg, Uint16 msgStatus, Uint16 slaveAddr, Uint16 numDataBytes, Uint16 numSlavePtrBytes, Uint16 slavePtrAddrHi, Uint16 slavePtrAddrLo )

This function can be used to validate and populate the specified settings and values into the specified I2C message structure.

##### Parameters

out	<i>msg</i>	The I2C message structure.
in	<i>msgStatus</i>	The initial I2C message status.
in	<i>slaveAddr</i>	The slave address.
in	<i>numDataBytes</i>	The number, if any, of data bytes, above any slave register pointer bytes, in the message.
in	<i>numSlavePtrBytes</i>	The number, if any, of slave register pointer bytes.
in	<i>slavePtrAddrHi</i>	The slave register pointer high byte. If only one byte, or none, (as indicated by numSlavePtrbytes) is to be used leave this at zero.
in	<i>slavePtrAddrLo</i>	The slave register pointer low byte. If no pointer bytes (as indicated by numSlavePtrbytes) are used leave this at zero.

#### 5.6.3.2 Uint16 i2cRead ( i2cMsg \* msg )

Starts an I2C-A read using the settings specified. Read bytes are saved to the buffer msg.msgBuffer[].

##### Parameters

in	<i>msg</i>	The I2C message struct.
----	------------	-------------------------

##### Returns

Error status.

#### 5.6.3.3 Uint16 i2cWrite ( i2cMsg \* msg )

Starts an I2C-A write using the settings and values specified.

##### Parameters

in	<i>msg</i>	The I2C message structure.
----	------------	----------------------------

##### Returns

Error Status.

#### 5.6.3.4 void initI2c ( void )

Initialises the I2C-A peripheral and relevant interrupts.

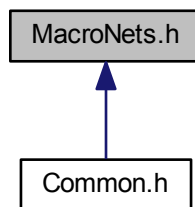
#### Warning

This function will clear any values already in the I2C peripheral registers.  
This function MUST be called before any other public I2C function.

## 5.7 MacroNets.h File Reference

DPLib macro net and value control functions.

This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [extDeviceSettings](#)
- struct [loadStageNets](#)
- struct [loadStageSettings](#)
- struct [xfmrStageNets](#)
- struct [xfmrStageSettings](#)
- struct [acStageNets](#)
- struct [acStageSettings](#)

### Macros

- #define [LOAD\\_0](#) 0
- #define [LOAD\\_1](#) 1
- #define [LOAD\\_2](#) 2
- #define [LOAD\\_3](#) 3
- #define [XFMR\\_STAGE](#) 4
- #define [AC\\_STAGE](#) 5
- #define [EXT\\_1](#) 6
- #define [EXT\\_2](#) 7

## Typedefs

- typedef enum [extSelect](#) [extSelect](#)
- typedef struct [extDeviceSettings](#) [extDeviceSettings](#)
- typedef enum [loadStage](#) [loadStage](#)
- typedef struct [loadStageNets](#) [loadStageNets](#)
- typedef struct [loadStageSettings](#) [loadStageSettings](#)
- typedef struct [xfmrStageNets](#) [xfmrStageNets](#)
- typedef struct [xfmrStageSettings](#) [xfmrStageSettings](#)
- typedef struct [acStageNets](#) [acStageNets](#)
- typedef struct [acStageSettings](#) [acStageSettings](#)

## Enumerations

- enum [extSelect](#) { [ext1](#) = 0, [ext2](#) = 1 }
- enum [loadStage](#) { [load1](#) = 0, [load2](#) = 1, [load3](#) = 2, [load4](#) = 3 }

## Functions

- void [setupNets](#) ([slaveMode](#) mode)
- void [stopAll](#) (void)
- Uint16 [runAll](#) (void)

## Variables

- Uint16 [stopAllFlag](#)
- Uint16 [enableAllFlag](#)
- [extDeviceSettings](#) [extSettings](#)
- [loadStageNets](#) [loadNets](#) [numberOfLoads]
- [loadStageSettings](#) [loadSettings](#) [numberOfLoads]
- [xfmrStageNets](#) [xfmrNets](#)
- [xfmrStageSettings](#) [xfmrSettings](#)
- [acStageNets](#) [acNets](#)
- [acStageSettings](#) [acSettings](#)

### 5.7.1 Detailed Description

DPLib macro net and value control functions.

### 5.7.2 Macro Definition Documentation

#### 5.7.2.1 #define AC\_STAGE 5

The index position for AC stage settings.

#### 5.7.2.2 #define EXT\_1 6

External temperature 1 channel reference.

#### 5.7.2.3 #define EXT\_2 7

External temperature 2 channel reference.

#### 5.7.2.4 **#define LOAD\_0 0**

The index position for Load 0 settings.

#### 5.7.2.5 **#define LOAD\_1 1**

The index position for Load 1 settings.

#### 5.7.2.6 **#define LOAD\_2 2**

The index position for Load 2 settings.

#### 5.7.2.7 **#define LOAD\_3 3**

The index position for Load 3 settings.

#### 5.7.2.8 **#define XFMR\_STAGE 4**

The index position for transformer stage settings.

### 5.7.3 **Typedef Documentation**

#### 5.7.3.1 **typedef struct acStageNets acStageNets**

A type to allow use of the AC stage nets structure.

#### 5.7.3.2 **typedef struct acStageSettings acStageSettings**

A type to allow use of the AC stage settings structure.

#### 5.7.3.3 **typedef struct extDeviceSettings extDeviceSettings**

A type to allow use of the external device settings structure.

#### 5.7.3.4 **typedef enum extSelect extSelect**

A type that allow specification of an external sensor.

#### 5.7.3.5 **typedef enum loadStage loadStage**

A type that allow specification of a load.

#### 5.7.3.6 **typedef struct loadStageNets loadStageNets**

A type to allow use of the load stage nets structure.

#### 5.7.3.7 **typedef struct loadStageSettings loadStageSettings**

A type to allow use of the load stage settings structure.



#### 5.7.3.8 typedef struct xfmrStageNets xfmrStageNets

A type to allow use of the transformer stage nets structure.

#### 5.7.3.9 typedef struct xfmrStageSettings xfmrStageSettings

A type to allow use of the AC stage settings structure.

### 5.7.4 Enumeration Type Documentation

#### 5.7.4.1 enum extSelect

The possible external sensor selections.

##### Enumerator

**ext1** External sensor 1.

**ext2** External sensor 2.

#### 5.7.4.2 enum loadStage

The possible load stage selections.

##### Enumerator

**load1** Load 1.

**load2** Load 2.

**load3** Load 3.

**load4** Load 4.

### 5.7.5 Function Documentation

#### 5.7.5.1 Uint16 runAll ( void )

Enables all IIR filter control law reference inputs.

##### Returns

Error status.

#### 5.7.5.2 void setupNets ( slaveMode mode )

Initialises macro settings and nets. IIR control law coefficients are handled outside of this function.

##### Warning

This MUST be called after DPL\_INIT()

This MUST be called after pwmMacroConfig()

## Parameters

<i>in</i>	<i>mode</i>	Selects the test unit mode to allow correct macro net connection.
-----------	-------------	---

## 5.7.5.3 void stopAll ( void )

Disables all circuit sections, zeroes all IIR references and flags each stage as disabled.

## 5.7.6 Variable Documentation

## 5.7.6.1 acStageNets acNets

A global instance of the AC nets structure.

## 5.7.6.2 acStageSettings acSettings

A global instance of the AC settings structure.

## 5.7.6.3 Uint16 enableAllFlag

Enable-all condition flag that allows status communication between the state machine tasks.

## 5.7.6.4 extDeviceSettings extSettings

A global instance of the external device settings structure.

## 5.7.6.5 loadStageNets loadNets[numberOfLoads]

A global array instance of the load nets structure that collects the nets for all 4 loads.

## 5.7.6.6 loadStageSettings loadSettings[numberOfLoads]

A global array instance of the load settings structure that collects the settings for all 4 loads.

## 5.7.6.7 Uint16 stopAllFlag

Stop-all condition flag that allows status communication between the state machine tasks.

## 5.7.6.8 xfmrStageNets xfmrNets

A global instance of the transformer nets structure.

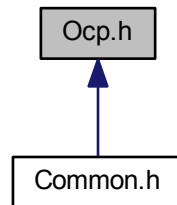
## 5.7.6.9 xfmrStageSettings xfmrSettings

A global instance of the transformer settings structure.

## 5.8 Ocp.h File Reference

Over-current protection functions.

This graph shows which files directly or indirectly include this file:



### Macros

- #define [LOAD1\\_OCP\\_TRIP](#) 1
- #define [LOAD2\\_OCP\\_TRIP](#) 2
- #define [LOAD3\\_OCP\\_TRIP](#) 4
- #define [LOAD4\\_OCP\\_TRIP](#) 8
- #define [DCMID\\_OCP\\_TRIP](#) 16
- #define [DCHV\\_OCP\\_TRIP](#) 32
- #define [AC\\_OCP\\_TRIP](#) 64

### Functions

- Uint16 [setLoadOcpLevel](#) (loadStage load, float32 dcLevel)
- Uint16 [getLoadOcpLevel](#) (loadStage load, float32 \*dcLevel)
- Uint16 [checkLoadOcp](#) (loadStage load)
- Uint16 [getLoadOcpState](#) (loadStage load)
- Uint16 [clearLoadOcp](#) (loadStage load)
- Uint16 [setDcMidOcpLevel](#) (float32 dcLevel)
- Uint16 [getDcMidOcpLevel](#) (float32 \*dcLevel)
- Uint16 [checkDcMidOcp](#) (void)
- Uint16 [getDcMidOcpState](#) (void)
- Uint16 [clearDcMidOcp](#) (void)
- Uint16 [setAcOcpLevel](#) (float32 pkLevel)
- Uint16 [getAcOcpLevel](#) (float32 \*pkLevel)
- Uint16 [tripAcOcp](#) (void)
- Uint16 [getAcOcpState](#) (void)
- Uint16 [clearAcOcp](#) (void)

#### 5.8.1 Detailed Description

Over-current protection functions. These functions require that the relevant measurement scales be set before use.

## 5.8.2 Macro Definition Documentation

### 5.8.2.1 #define AC\_OCP\_TRIP 64

OCP flag register AC bit.

### 5.8.2.2 #define DCHV\_OCP\_TRIP 32

OCP flag register DC HV bit.

### 5.8.2.3 #define DCMID\_OCP\_TRIP 16

OCP flag register DC MID bit.

### 5.8.2.4 #define LOAD1\_OCP\_TRIP 1

OCP flag register load 1 bit.

### 5.8.2.5 #define LOAD2\_OCP\_TRIP 2

OCP flag register load 2 bit.

### 5.8.2.6 #define LOAD3\_OCP\_TRIP 4

OCP flag register load 3 bit.

### 5.8.2.7 #define LOAD4\_OCP\_TRIP 8

OCP flag register load 4 bit.

## 5.8.3 Function Documentation

### 5.8.3.1 Uint16 checkDcMidOcp ( void )

Checks the current reading of the DC stage against the DC OCP limit. Raises the DC OCP flag if the reading is above the limit.

#### Returns

Error status

### 5.8.3.2 Uint16 checkLoadOcp ( loadStage load )

Checks the current reading of the specified load against the load OCP limit. Raises the load OCP flag if the reading is above the limit.

#### Parameters

---

<i>in</i>	<i>load</i>	Specifies the load on which the reading is to be tested.
-----------	-------------	--

**Returns**

Error status

**5.8.3.3 Uint16 clearAcOcp ( void )**

Clears the AC stage OCP state.

**Returns**

Error status.

**5.8.3.4 Uint16 clearDcMidOcp ( void )**

Clears the DC stage OCP state.

**Returns**

Error status.

**5.8.3.5 Uint16 clearLoadOcp ( loadStage *load* )**

Clears the OCP state for the specified load.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load for which the OCP state is to be cleared.
-----------	-------------	--

**Returns**

Error status.

**5.8.3.6 Uint16 getAcOcpLevel ( float32 \* *pkLevel* )**

Queries the over current protection level for the AC stage.

**Parameters**

<i>out</i>	<i>pkLevel</i>	Pointer to location at which to place the query result (amps).
------------	----------------	--

**Returns**

Error status.

**5.8.3.7 Uint16 getAcOcpState ( void )**

Queries the state of the AC OCP flag.

**Returns**

True if OCP flag has been raised.

5.8.3.8 Uint16 getDcMidOcpLevel ( float32 \* *dcLevel* )

Queries the over current protection level for the DC stage.

## Parameters

out	dcLevel	Pointer to location at which to place the query result (amps).
-----	---------	--

## Returns

Error status.

## 5.8.3.9 Uint16 getDcMidOcpState ( void )

Queries the state of the DC OCP flag.

## Returns

True if OCP flag has been raised.

## 5.8.3.10 Uint16 getLoadOcpLevel ( loadStage load, float32 \* dcLevel )

Queries the over current protection level for the specified load.

## Parameters

in	load	Specifies the load on which the setting is to be queried.
out	dcLevel	Pointer to location at which to place the query result (amps).

## Returns

Error status.

## 5.8.3.11 Uint16 getLoadOcpState ( loadStage load )

Queries the state of the OCP flag for the specified load.

## Parameters

in	load	Specifies the load on which to check the flag.
----	------	--

## Returns

True if OCP flag has been raised.

## 5.8.3.12 Uint16 setAcOcpLevel ( float32 pkLevel )

Sets the over current protection limit for the AC stage.

## Parameters

in	pkLevel	Specifies the peak value to be applied (amps).
----	---------	--

## Returns

Error status.

## 5.8.3.13 Uint16 setDcMidOcpLevel ( float32 dcLevel )

Sets the over current protection limit for the DC stage.

**Parameters**

in	<i>dcLevel</i>	Specifies the DC value to be applied (amps).
----	----------------	--

**Returns**

Error status.

**5.8.3.14 Uint16 setLoadOcpLevel ( loadStage *load*, float32 *dcLevel* )**

< OCP flag register. Bits are set to indicate an OCP condition has been found. Sets the over current protection level for the specified load.

**Parameters**

in	<i>load</i>	Specifies the load the setting is to be applied to.
in	<i>dcLevel</i>	Specifies the DC value to be applied (amps).

**Returns**

Error status.

**5.8.3.15 Uint16 tripAcOcp ( void )**

Trips the AC OCP. This provides an OCP function for the trip zone ISR to call when the AC comparator is triggered.

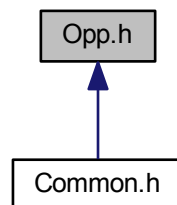
**Returns**

Error status

**5.9 Opp.h File Reference**

Over-power protection functions.

This graph shows which files directly or indirectly include this file:

**Macros**

- #define `LOAD1_OPP_TRIP` 1
- #define `LOAD2_OPP_TRIP` 2



- #define `LOAD3_OPP_TRIP` 4
- #define `LOAD4_OPP_TRIP` 8
- #define `DCMID_OPP_TRIP` 16
- #define `DCHV_OPP_TRIP` 32
- #define `AC_OPP_TRIP` 64

## Functions

- Uint16 `checkLoadOpp` (loadStage load)
- Uint16 `getLoadOppState` (loadStage load)
- Uint16 `clearLoadOpp` (loadStage load)
- Uint16 `checkAcOpp` (void)
- Uint16 `getAcOppState` (void)
- Uint16 `clearAcOpp` (void)

### 5.9.1 Detailed Description

Over-power protection functions. These functions require that the relevant measurement scales be set before use.

### 5.9.2 Macro Definition Documentation

#### 5.9.2.1 #define `AC_OPP_TRIP` 64

OPP flag register AC bit.

#### 5.9.2.2 #define `DCHV_OPP_TRIP` 32

OPP flag register DC HV bit.

#### 5.9.2.3 #define `DCMID_OPP_TRIP` 16

OPP flag register DC MID bit.

#### 5.9.2.4 #define `LOAD1_OPP_TRIP` 1

OPP flag register load 1 bit.

#### 5.9.2.5 #define `LOAD2_OPP_TRIP` 2

OPP flag register load 2 bit.

#### 5.9.2.6 #define `LOAD3_OPP_TRIP` 4

OPP flag register load 3 bit.

#### 5.9.2.7 #define `LOAD4_OPP_TRIP` 8

OPP flag register load 4 bit.

### 5.9.3 Function Documentation

#### 5.9.3.1 Uint16 checkAcOpp ( void )

Checks the current reading of the AC stage against the AC OPP limit. Raises the AC OPP flag if the reading is above the limit.

##### Returns

Error status

#### 5.9.3.2 Uint16 checkLoadOpp ( loadStage *load* )

< OPP flag register. Bits are set to indicate an OPP condition has been found. Checks the power reading of the specified load against the load OPP limit. Raises the load OPP flag if the reading is above the limit.

##### Parameters

<i>in</i>	<i>load</i>	Specifies the load on which the reading is to be tested.
-----------	-------------	--

##### Returns

Error status

#### 5.9.3.3 Uint16 clearAcOpp ( void )

Clears the AC stage OPP state.

##### Returns

Error status.

#### 5.9.3.4 Uint16 clearLoadOpp ( loadStage *load* )

Clears the OPP state for the specified load.

##### Parameters

<i>in</i>	<i>load</i>	Specifies the load for which the OPP state is to be cleared.
-----------	-------------	--

##### Returns

Error status.

#### 5.9.3.5 Uint16 getAcOppState ( void )

Queries the state of the AC OPP flag.

##### Returns

True if OPP flag has been raised.

#### 5.9.3.6 Uint16 getLoadOppState ( loadStage *load* )

Queries the state of the OPP flag for the specified load.

## Parameters

<i>in</i>	<i>load</i>	Specifies the load on which to check the flag.
-----------	-------------	--

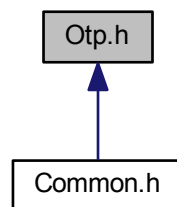
## Returns

True if OPP flag has been raised.

## 5.10 Otp.h File Reference

Over-temperature protection functions.

This graph shows which files directly or indirectly include this file:



## Macros

- `#define` [LOAD1\\_OTP\\_TRIP](#) 1
- `#define` [LOAD2\\_OTP\\_TRIP](#) 2
- `#define` [LOAD3\\_OTP\\_TRIP](#) 4
- `#define` [LOAD4\\_OTP\\_TRIP](#) 8
- `#define` [DC\\_OTP\\_TRIP](#) 16
- `#define` [RSRVD\\_OTP\\_TRIP](#) 32
- `#define` [AC\\_OTP\\_TRIP](#) 64
- `#define` [EXT1\\_OTP\\_TRIP](#) 128
- `#define` [EXT2\\_OTP\\_TRIP](#) 256

## Functions

- `Uint16` [setLoadOtpLevel](#) (`loadStage` load, `float32` level)
- `Uint16` [getLoadOtpLevel](#) (`loadStage` load, `float32` \*level)
- `Uint16` [checkLoadOtp](#) (`loadStage` load)
- `Uint16` [getLoadOtpState](#) (`loadStage` load)
- `Uint16` [clearLoadOtp](#) (`loadStage` load)
- `Uint16` [setDcOtpLevel](#) (`float32` level)
- `Uint16` [getDcOtpLevel](#) (`float32` \*level)
- `Uint16` [checkDcOtp](#) (void)
- `Uint16` [getDcOtpState](#) (void)
- `Uint16` [clearDcOtp](#) (void)
- `Uint16` [setAcOtpLevel](#) (`float32` level)
- `Uint16` [getAcOtpLevel](#) (`float32` \*level)

- Uint16 `checkAcOtp` (void)
- Uint16 `getAcOtpState` (void)
- Uint16 `clearAcOtp` (void)
- Uint16 `setExtOtpLevel` (`extSelect` ext, float32 level)
- Uint16 `getExtOtpLevel` (`extSelect` ext, float32 \*level)
- Uint16 `checkExtOtp` (`extSelect` ext)
- Uint16 `getExtOtpState` (`extSelect` ext)
- Uint16 `clearExtOtp` (`extSelect` ext)

### 5.10.1 Detailed Description

Over-temperature protection functions.

### 5.10.2 Macro Definition Documentation

#### 5.10.2.1 `#define AC_OTP_TRIP 64`

OTP flag register AC bit.

#### 5.10.2.2 `#define DC_OTP_TRIP 16`

OTP flag register DC bit.

#### 5.10.2.3 `#define EXT1_OTP_TRIP 128`

OTP flag register external 1 bit.

#### 5.10.2.4 `#define EXT2_OTP_TRIP 256`

OTP flag register external 2 bit.

#### 5.10.2.5 `#define LOAD1_OTP_TRIP 1`

OTP flag register load 1 bit.

#### 5.10.2.6 `#define LOAD2_OTP_TRIP 2`

OTP flag register load 2 bit.

#### 5.10.2.7 `#define LOAD3_OTP_TRIP 4`

OTP flag register load 3 bit.

#### 5.10.2.8 `#define LOAD4_OTP_TRIP 8`

OTP flag register load 4 bit.

#### 5.10.2.9 `#define RSRVD_OTP_TRIP 32`

Reserved bit.

### 5.10.3 Function Documentation

#### 5.10.3.1 Uint16 checkAcOtp ( void )

Checks the temperature reading of the AC stage against the AC OTP limit. Raises the AC OTP flag if the reading is above the limit.

##### Returns

Error status

#### 5.10.3.2 Uint16 checkDcOtp ( void )

Checks the temperature reading of the DC stage against the DC OTP limit. Raises the DC OTP flag if the reading is above the limit.

##### Returns

Error status

#### 5.10.3.3 Uint16 checkExtOtp ( extSelect ext )

Checks the temperature reading of the specified external sensor against the sensor OTP limit. Raises the load OTP flag if the reading is above the limit.

##### Parameters

in	ext	Specifies the sensor on which the setting is to be queried.
----	-----	---

##### Returns

Error status

#### 5.10.3.4 Uint16 checkLoadOtp ( loadStage load )

Checks the temperature reading of the specified load against the load OTP limit. Raises the load OTP flag if the reading is above the limit.

##### Parameters

in	load	Specifies the load on which the reading is to be tested.
----	------	--

##### Returns

Error status

#### 5.10.3.5 Uint16 clearAcOtp ( void )

Clears the AC stage OTP state.

##### Returns

Error status.

#### 5.10.3.6 Uint16 clearDcOtp ( void )

Clears the DC stage OTP state.

##### Returns

Error status.

#### 5.10.3.7 Uint16 clearExtOtp ( extSelect ext )

Clears the OTP state for the specified external sensor.

##### Parameters

<i>in</i>	<i>ext</i>	Specifies the external sensor for which the OTP state is to be cleared.
-----------	------------	---

##### Returns

Error status.

#### 5.10.3.8 Uint16 clearLoadOtp ( loadStage load )

Clears the OTP state for the specified load.

##### Parameters

<i>in</i>	<i>load</i>	Specifies the load for which the OTP state is to be cleared.
-----------	-------------	--

##### Returns

Error status.

#### 5.10.3.9 Uint16 getAcOtpLevel ( float32 \* level )

Queries the over temperature protection level for the AC stage.

##### Parameters

<i>out</i>	<i>level</i>	Pointer to location at which to place the query result ( ° C).
------------	--------------	--

##### Returns

Error status.

#### 5.10.3.10 Uint16 getAcOtpState ( void )

Queries the state of the AC OTP flag.

##### Returns

True if OTP flag has been raised.

#### 5.10.3.11 Uint16 getDcOtpLevel ( float32 \* level )

Queries the over current protection level for the DC stage.

**Parameters**

<i>out</i>	<i>level</i>	Pointer to location at which to place the query result ( ° C).
------------	--------------	--

**Returns**

Error status.

**5.10.3.12 Uint16 getDcOtpState ( void )**

Queries the state of the DC OTP flag.

**Returns**

True if OTP flag has been raised.

**5.10.3.13 Uint16 getExtOtpLevel ( extSelect *ext*, float32 \* *level* )**

Queries the over temperature protection level for the specified external sensor.

**Parameters**

<i>in</i>	<i>ext</i>	Specifies the external sensor on which the setting is to be queried.
<i>out</i>	<i>level</i>	Pointer to location at which to place the query result ( ° C).

**Returns**

Error status.

**5.10.3.14 Uint16 getExtOtpState ( extSelect *ext* )**

Queries the state of the OTP flag for the specified external sensor.

**Parameters**

<i>in</i>	<i>ext</i>	Specifies the sensor on which to check the flag.
-----------	------------	--

**Returns**

True if OTP flag has been raised.

**5.10.3.15 Uint16 getLoadOtpLevel ( loadStage *load*, float32 \* *level* )**

Queries the over temperature protection level for the specified load.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load on which the setting is to be queried.
<i>out</i>	<i>level</i>	Pointer to location at which to place the query result ( ° C).

**Returns**

Error status.

**5.10.3.16 Uint16 getLoadOtpState ( loadStage *load* )**

Queries the state of the OTP flag for the specified load.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load on which to check the flag.
-----------	-------------	--

**Returns**

True if OTP flag has been raised.

**5.10.3.17 Uint16 setAcOtpLevel ( float32 level )**

Sets the over current protection limit for the AC stage.

**Parameters**

<i>in</i>	<i>level</i>	Specifies the value to be applied ( ° C).
-----------	--------------	---

**Returns**

Error status.

**5.10.3.18 Uint16 setDcOtpLevel ( float32 level )**

Sets the over temperature protection limit for the DC stage.

**Parameters**

<i>in</i>	<i>level</i>	Specifies the value to be applied ( ° C).
-----------	--------------	---

**Returns**

Error status.

**5.10.3.19 Uint16 setExtOtpLevel ( extSelect ext, float32 level )**

Sets the over temperature protection level for the specified external sensor.

**Parameters**

<i>in</i>	<i>ext</i>	Specifies the sensor the setting is to be applied to.
<i>in</i>	<i>level</i>	Specifies the value to be applied ( ° C).

**Returns**

Error status.

**5.10.3.20 Uint16 setLoadOtpLevel ( loadStage load, float32 level )**

< OTP flag register. Bits are set to indicate an OTP condition has been found. Sets the over temperature protection level for the specified load.

**Parameters**



in	<i>load</i>	Specifies the load the setting is to be applied to.
in	<i>level</i>	Specifies the value to be applied ( ° C).

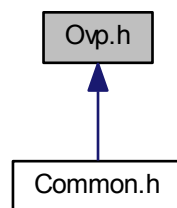
**Returns**

Error status.

## 5.11 Ovp.h File Reference

Over-voltage protection functions.

This graph shows which files directly or indirectly include this file:

**Macros**

- #define [LOAD1\\_OVP\\_TRIP](#) 1
- #define [LOAD2\\_OVP\\_TRIP](#) 2
- #define [LOAD3\\_OVP\\_TRIP](#) 4
- #define [LOAD4\\_OVP\\_TRIP](#) 8
- #define [DCMID\\_OVP\\_TRIP](#) 16
- #define [DCHV\\_OVP\\_TRIP](#) 32
- #define [AC\\_OVP\\_TRIP](#) 64

**Functions**

- Uint16 [checkLoadOvp](#) (loadStage load)
- Uint16 [getLoadOvpState](#) (loadStage load)
- Uint16 [clearLoadOvp](#) (loadStage load)
- Uint16 [tripDcMidOvp](#) (void)
- Uint16 [getDcMidOvpState](#) (void)
- Uint16 [clearDcMidOvp](#) (void)
- Uint16 [setDcHvOvpLevel](#) (float32 dcLevel)
- Uint16 [getDcHvOvpLevel](#) (float32 \*dcLevel)
- Uint16 [checkDcHvOvp](#) (void)
- Uint16 [getDcHvOvpState](#) (void)
- Uint16 [clearDcHvOvp](#) (void)
- Uint16 [setAcOvpLevel](#) (float32 pkLevel)
- Uint16 [getAcOvpLevel](#) (float32 \*pkLevel)
- Uint16 [checkAcOvp](#) (void)
- Uint16 [getAcOvpState](#) (void)
- Uint16 [clearAcOvp](#) (void)

### 5.11.1 Detailed Description

Over-voltage protection functions. These functions require that the relevant measurement scales be set before use.

### 5.11.2 Macro Definition Documentation

#### 5.11.2.1 `#define AC_OVP_TRIP 64`

OVP flag register AC bit.

#### 5.11.2.2 `#define DCHV_OVP_TRIP 32`

OVP flag register DC HV bit.

#### 5.11.2.3 `#define DCMID_OVP_TRIP 16`

OVP flag register DC MID bit.

#### 5.11.2.4 `#define LOAD1_OVP_TRIP 1`

OVP flag register load 1 bit.

#### 5.11.2.5 `#define LOAD2_OVP_TRIP 2`

OVP flag register load 2 bit.

#### 5.11.2.6 `#define LOAD3_OVP_TRIP 4`

OVP flag register load 3 bit.

#### 5.11.2.7 `#define LOAD4_OVP_TRIP 8`

OVP flag register load 4 bit.

### 5.11.3 Function Documentation

#### 5.11.3.1 `Uint16 checkAcOvp ( void )`

Checks the voltage reading of the AC stage against the AC OVP limit. Raises the AC OVP flag if the reading is above the limit.

##### Returns

Error status

#### 5.11.3.2 `Uint16 checkDcHvOvp ( void )`

Checks the voltage reading of the DC HV stage against the DC HV OVP limit. Raises the DC HV OVP flag if the reading is above the limit.

**Returns**

Error status

**5.11.3.3 Uint16 checkLoadOvp ( loadStage *load* )**

< OVP flag register. Bits are set to indicate an OVP condition has been found. Checks the voltage reading of the specified load against the load OVP limit. Raises the load OVP flag if the reading is above the limit.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load on which the reading is to be tested.
-----------	-------------	--

**Returns**

Error status

**5.11.3.4 Uint16 clearAcOvp ( void )**

Clears the AC stage OVP state.

**Returns**

Error status.

**5.11.3.5 Uint16 clearDcHvOvp ( void )**

Clears the DC HV stage OVP state.

**Returns**

Error status.

**5.11.3.6 Uint16 clearDcMidOvp ( void )**

Clears the DC Mid stage OVP state.

**Returns**

Error status.

**5.11.3.7 Uint16 clearLoadOvp ( loadStage *load* )**

Clears the OVP state for the specified load.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load for which the OVP state is to be cleared.
-----------	-------------	--

**Returns**

Error status.

**5.11.3.8 Uint16 getAcOvpLevel ( float32 \* *pkLevel* )**

Queries the over voltage protection level for the AC stage.

**Parameters**

<code>out</code>	<code>pkLevel</code>	Pointer to location at which to place the query result (volts).
------------------	----------------------	---

**Returns**

Error status.

**5.11.3.9 Uint16 getAcOvpState ( void )**

Queries the state of the AC OVP flag.

**Returns**

True if OVP flag has been raised.

**5.11.3.10 Uint16 getDcHvOvpLevel ( float32 \* *dcLevel* )**

Queries the over voltage protection level for the DC HV stage.

**Parameters**

<code>out</code>	<code>dcLevel</code>	Pointer to location at which to place the query result (volts).
------------------	----------------------	---

**Returns**

Error status.

**5.11.3.11 Uint16 getDcHvOvpState ( void )**

Queries the state of the DC HV OVP flag.

**Returns**

True if OVP flag has been raised.

**5.11.3.12 Uint16 getDcMidOvpState ( void )**

Queries the state of the DC Mid OVP flag.

**Returns**

True if OVP flag has been raised.

**5.11.3.13 Uint16 getLoadOvpState ( loadStage *load* )**

Queries the state of the OVP flag for the specified load.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load on which to check the flag.
-----------	-------------	--

**Returns**

True if OVP flag has been raised.

**5.11.3.14 Uint16 setAcOvpLevel ( float32 *pkLevel* )**

Sets the over voltage protection limit for the AC stage.

**Parameters**

<i>in</i>	<i>pkLevel</i>	Specifies the peak value to be applied (volts).
-----------	----------------	---

**Returns**

Error status.

**5.11.3.15 Uint16 setDcHvOvpLevel ( float32 *dcLevel* )**

Sets the over voltage protection limit for the DC HV stage.

**Parameters**

<i>in</i>	<i>dcLevel</i>	Specifies the DC value to be applied (volts).
-----------	----------------	---

**Returns**

Error status.

**5.11.3.16 Uint16 tripDcMidOvp ( void )**

Trips the DC Mid OVP. This provides an OVP function for the trip zone ISR to call when the DC comparator is triggered.

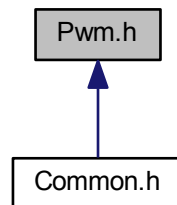
**Returns**

Error status

## 5.12 Pwm.h File Reference

PWM and related functions.

This graph shows which files directly or indirectly include this file:

**Macros**

- `#define PWM_1_PRD` 462
- `#define PWM_2_PRD` 462
- `#define PWM_3_PRD` 462

**Functions**

- void `initPwm` (void)
- interrupt void `DPL_ISR` (void)

**Variables**

- volatile int32 \* `PWMDRV_2ch_UpCnt_Duty1A`
- volatile int32 \* `PWMDRV_2ch_UpCnt_Duty1B`
- volatile int32 \* `PWMDRV_2ch_UpCnt_Duty2A`
- volatile int32 \* `PWMDRV_2ch_UpCnt_Duty2B`
- volatile int32 \* `PWMDRV_2ch_UpCnt_Duty3A`
- volatile int32 \* `PWMDRV_2ch_UpCnt_Duty3B`

### 5.12.1 Detailed Description

PWM and related functions.

### 5.12.2 Macro Definition Documentation

#### 5.12.2.1 `#define PWM_1_PRD` 462

Defines the period setting for PWM1.  $60\text{MHz} / 462 = 130\text{kHz}$ .

#### 5.12.2.2 `#define PWM_2_PRD 462`

Defines the period setting for PWM1.  $60\text{MHz} / 462 = 130\text{kHz}$ .

#### 5.12.2.3 `#define PWM_3_PRD 462`

Defines the period setting for PWM1.  $60\text{MHz} / 462 \sim 130\text{kHz}$ .

### 5.12.3 Function Documentation

#### 5.12.3.1 `interrupt void DPL_ISR ( void )`

Digital power control loop interrupt service routine Located in the assembly file `BurnInUnit_ISR.asm`

#### 5.12.3.2 `void initPwm ( void )`

Initialises PWM macros, SOC's and ISR (DPL trigger).

### 5.12.4 Variable Documentation

#### 5.12.4.1 `volatile int32* PWMDRV_2ch_UpCnt_Duty1A`

Channel 0 PWM terminal pointer.

#### 5.12.4.2 `volatile int32* PWMDRV_2ch_UpCnt_Duty1B`

Channel 1 PWM terminal pointers.

#### 5.12.4.3 `volatile int32* PWMDRV_2ch_UpCnt_Duty2A`

Channel 2 PWM terminal pointer.

#### 5.12.4.4 `volatile int32* PWMDRV_2ch_UpCnt_Duty2B`

Channel 3 PWM terminal pointer.

#### 5.12.4.5 `volatile int32* PWMDRV_2ch_UpCnt_Duty3A`

Interboost PWM terminal pointer.

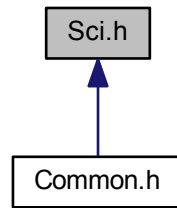
#### 5.12.4.6 `volatile int32* PWMDRV_2ch_UpCnt_Duty3B`

AC stage PWM terminal pointer.

## 5.13 Sci.h File Reference

Serial communications interface functions.

This graph shows which files directly or indirectly include this file:



## Macros

- #define [SCIBAUD\\_MIN](#) 29
- #define [SCIFFRX\\_INT\\_LVL](#) 1
- #define [SCIFFTX\\_INT\\_LVL](#) 0
- #define [SCIFFTX\\_FILL\\_LVL](#) 4

## Functions

- Uint16 [scilnit](#) (Uint32 baud)
- void [sciTx](#) (void)

### 5.13.1 Detailed Description

Serial communications interface functions.

On the TI C2000 LaunchPad XL:

Function	GPIO	PCB Pin
SCI_RX	GPIO28	J1-3
SCI_TX	GPIO29	J1-4

## Warning

On the TI C2000 LaunchPad XL the switch 4 (S4) should be in the OFF position to communicate with devices other than the on-board USB controller.

### 5.13.2 Macro Definition Documentation

#### 5.13.2.1 #define SCIBAUD\_MIN 29

Minimum allowable value of SCI Baud.

#### 5.13.2.2 #define SCIFFRX\_INT\_LVL 1

Interrupt level for receiving FIFO.



## 5.13.2.3 #define SCIFFTX\_FILL\_LVL 4

Fill level for transmission FIFO.

## 5.13.2.4 #define SCIFFTX\_INT\_LVL 0

Interrupt level for transmission FIFO.

## 5.13.3 Function Documentation

5.13.3.1 Uint16 scilnit ( Uint32 *baud* )

Initialises the SCI(A) peripheral and relevant interrupts.

Parameters

<i>in</i>	<i>baud</i>	The baud rate that the SCI should use minimum value is set by SCIBAUD_MIN.
-----------	-------------	--

Returns

Error status.

Warning

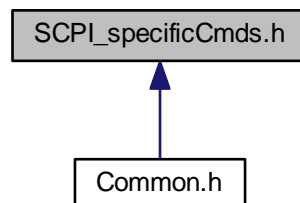
This function will clear any values already in the SCI peripheral registers.  
This function MUST be called before any other SCI function.

## 5.13.3.2 void sciTx ( void )

Transmits whatever data is on the SCPI output queue if SCI has been selected as the external communications type.

## 5.14 SCPI\_specificCmds.h File Reference

This graph shows which files directly or indirectly include this file:



### 5.14.1 Detailed Description

This file includes the functions for registering the application specific commands and the related callback functions

These commands and their associated callback functions can be edited to suit the application in use, though they should follow the guidelines and conventions laid out in IEEE 488.2-1992 and SCPI-1999

Please follow the information in the parser documentation and the file SCPI\_specificCmds.c on how to add or change message headers, commands and parameters

To register a new command device specific tree node, add a registerChild() call to the function registerSpecificCommands() with the form "err += registerChild(CHILD\_LONG\_NAME, PARENT\_SHORT\_NAME, BOOL\_IS\_HEADER\_ONLY, BOOL\_IS\_QUERY, CALLBACK\_HANDLE);", where:

- CHILD\_LONG\_NAME is the long form of the name of the child node to be registered.
- PARENT\_SHORT\_NAME is the short form of the name of the parent of the child node to be registered. This must be an already existing node.
- BOOL\_IS\_HEADER\_ONLY is a boolean value that indicates if the node being registered is a header only (i.e. is not a command) or not.
- BOOL\_IS\_QUERY is a boolean value that indicates if the node being registered may be queried or not.
- CALLBACK\_HANDLE is a pointer to the function to be associated with this child if it is a command, or NULL if it is a header only.

For further information on the operation of registerChild() please see the function description.

All string literals MUST be in upper case and enclosed in quotation marks.

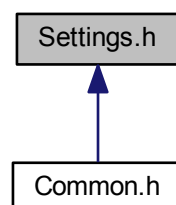
Node long and short names must conform to the standard as outlined by SCPI-99 in 6.2.1

All tree nodes are children of at least "ROOT". Changes to the number of nodes registered should be reflected by the user in the number defined for TREE\_CHILD\_LIMIT in scpi.h

## 5.15 Settings.h File Reference

Major build definitions and settings for the project.

This graph shows which files directly or indirectly include this file:



### Macros

- #define INCR\_BUILD 2
- #define DEBUG

- #define LOG\_SIN
  - #define DUAL\_CNTL\_AC
  - #define VSSA 0l
  - #define VDDA 3300l
  - #define LOAD\_I\_SCALE 0.25f
  - #define LOAD\_V\_R1 18.0f
  - #define LOAD\_V\_R2 1.0f
  - #define HV\_V\_R1 540.0f
  - #define HV\_V\_R2 4.3f
  - #define MID\_I\_SCALE 0.25f
  - #define MID\_V\_R1 18.0f
  - #define MID\_V\_R2 1.0f
  - #define AC\_I\_SCALE 0.25f
  - #define AC\_V\_R1 540.0f
  - #define AC\_V\_R2 4.3f
  - #define LOAD\_IDCLVL\_MAX 35
  - #define LOAD\_VDCLVL\_FIX 60
  - #define LOAD\_PWRLVL\_FIX 200
  - #define DCMID\_VDCLVL\_FIX 400
  - #define DCMIDV\_STRTUP\_LMT 10.0f
  - #define DCHV\_UVLCK\_LMT 330.0f
  - #define AC\_IRMSLV\_L\_MAX 10
  - #define AC\_VRMSLV\_L\_MAX 250
  - #define AC\_PWRLVL\_FIX 800
  - #define XFMR\_PWMF\_FIX 130
  - #define SQRT\_2 1.41429
  - #define RECP\_SQRT\_2 0.70711
- 
- #define CHANNEL\_OOB 0x10
  - #define VALUE\_OOB 0x11
  - #define OCP\_TRIP 0x12
  - #define OVP\_TRIP 0x13
  - #define OPP\_TRIP 0x14
  - #define OTP\_TRIP 0x15
  - #define UVLKT\_TRIP 0x16
  - #define TRIP\_ALERT 0x17
  - #define I2C\_READ\_WRONG\_MSG 0x20
  - #define I2C\_WRITE\_WRONG\_MSG 0x21
  - #define I2C\_STP\_NOT\_READY 0x22
  - #define I2C\_BUS\_BUSY 0x23
  - #define I2C\_INVALID\_ISRC 0x24

### 5.15.1 Detailed Description

Major build definitions and settings for the project.

#### Warning

This file is included and referenced by ISR.asm, main() and mnConnectNets().  
When changes are made to this file please use rebuild all.

## 5.15.2 Macro Definition Documentation

### 5.15.2.1 `#define AC_I_SCALE 0.25f`

Scaling factor for AC current ADC. Volts/amp.

### 5.15.2.2 `#define AC_IRMSLVL_MAX 10`

The maximum allowable value, in amps (RMS), for the AC current level.

### 5.15.2.3 `#define AC_PWRLVL_FIX 800`

The fixed value, in watts, for the AC OPP level.

### 5.15.2.4 `#define AC_V_R1 540.0f`

Scaling voltage divider R1 resistor value for AC voltage ADC. kOhms.

### 5.15.2.5 `#define AC_V_R2 4.3f`

Scaling voltage divider R2 resistor value for AC voltage ADC. kOhms.

### 5.15.2.6 `#define AC_VRMSLVL_MAX 250`

The maximum allowable value, in volts (RMS), for the AC voltage level.

### 5.15.2.7 `#define CHANNEL_OOB 0x10`

Channel out of bounds error code.

### 5.15.2.8 `#define DCHV_UVLCK_LMT 330.0f`

The DC HV under-voltage lockout limit level.

### 5.15.2.9 `#define DCMID_VDCLVL_FIX 400`

The fixed maximum value, in volts, for the DC mid voltage level.

### 5.15.2.10 `#define DCMIDV_STRTPUP_LMT 10.0f`

The maximum limit for DC MID voltage at to allow transformer stage startup.

### 5.15.2.11 `#define DEBUG`

Includes and makes functions and variables public that are used only for debugging purposes.

### 5.15.2.12 `#define DUAL_CNTL_AC`

Uses the dual CNTL AC control instead of single VCtrl. Cannot be used if PID is still in use.

**5.15.2.13 #define HV\_V\_R1 540.0f**

Scaling voltage divider R1 resistor value for HV voltage ADC. kOhms.

**5.15.2.14 #define HV\_V\_R2 4.3f**

Scaling voltage divider R2 resistor value for HV voltage ADC. kOhms.

**5.15.2.15 #define I2C\_BUS\_BUSY 0x23**

I2C bus already busy error code.

**5.15.2.16 #define I2C\_INVALID\_ISRC 0x24**

Invalid I2C interrupt source error code.

**5.15.2.17 #define I2C\_READ\_WRONG\_MSG 0x20**

Incorrect type I2C message read error code.

**5.15.2.18 #define I2C\_STP\_NOT\_READY 0x22**

I2C stop bit was not yet received error code.

**5.15.2.19 #define I2C\_WRITE\_WRONG\_MSG 0x21**

Incorrect type I2C write message error code.

**5.15.2.20 #define INCR\_BUILD 2**

Alters the digital power control loop between closed or open. Open-Loop: 1. Closed-loop: 2.

**5.15.2.21 #define LOAD\_I\_SCALE 0.25f**

Scaling factor for load current ADC.

**5.15.2.22 #define LOAD\_IDCLVL\_MAX 35**

The maximum allowable value, in amps, for the load current levels.

**5.15.2.23 #define LOAD\_PWRLVL\_FIX 200**

The fixed maximum value, in watts, for the load power levels.

**5.15.2.24 #define LOAD\_V\_R1 18.0f**

Scaling voltage divider R1 resistor value for load voltage ADC. kOhms.

5.15.2.25 **#define LOAD\_V\_R2 1.0f**

Scaling voltage divider R2 resistor value for load voltage ADC. kOhms.

5.15.2.26 **#define LOAD\_VDCLVL\_FIX 60**

The fixed maximum value, in volts, for the load voltage levels.

5.15.2.27 **#define LOG\_SIN**

Logs the sine wave VOut and SignOut data into data-log buffers for debug graphing.

5.15.2.28 **#define MID\_I\_SCALE 0.25f**

Scaling factor for MID current ADC. Volts/amp.

5.15.2.29 **#define MID\_V\_R1 18.0f**

Scaling voltage divider R1 resistor value for MID voltage ADC. kOhms.

5.15.2.30 **#define MID\_V\_R2 1.0f**

Scaling voltage divider R2 resistor value for MID voltage ADC. kOhms.

5.15.2.31 **#define OCP\_TRIP 0x12**

Over-current protection trip error code.

5.15.2.32 **#define OPP\_TRIP 0x14**

Over-power protection trip error code.

5.15.2.33 **#define OTP\_TRIP 0x15**

Over-temperature protection trip error code.

5.15.2.34 **#define OVP\_TRIP 0x13**

Over-voltage protection trip error code.

5.15.2.35 **#define RECP\_SQRT\_2 0.70711**

1/sqrt(2) constant used for RMS calculations.

5.15.2.36 **#define SQRT\_2 1.41429**

Sqrt(2) constant used for RMS calculations.

**5.15.2.37 #define TRIP\_ALERT 0x17**

General purpose alert trip error code.

**5.15.2.38 #define UVLKT\_TRIP 0x16**

Under-voltage lock-out trip error code.

**5.15.2.39 #define VALUE\_OOB 0x11**

Value out of bounds error code.

**5.15.2.40 #define VDDA 3300l**

System VMAXREF (millivolts).

**5.15.2.41 #define VSSA 0l**

System VLOWREF (millivolts).

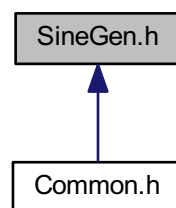
**5.15.2.42 #define XFMR\_PWMF\_FIX 130**

The fixed frequency (kHz) setting for the transformer PWM (at 50% duty).

## 5.16 SineGen.h File Reference

Signal generator functions.

This graph shows which files directly or indirectly include this file:



### Macros

- #define [SIN\\_F\\_ISR](#) 60000000.0f/PWM\_1\_PRD
- #define [SIN\\_F\\_ISR\\_DVSR](#) 6
- #define [SIN\\_F\\_MAX](#) 200u
- #define [SIN\\_F\\_REQ](#) 50.0
- #define [SIN\\_OFST](#) 0

- `#define SIN_PHSE 0`
- `#define SIN_DFLT_GAIN 0.0`
- `#define SIN_DFLT_RCTFY TRUE`
- `#define SIN_CHANNEL AC_STAGE`

## Functions

- void `initSine` (Uint16 enablePhaseOut)
- void `updateSineGain` (void)
- void `updateSineSignal` (void)
- void `enableSinePhaseOut` (void)
- void `disableSinePhaseOut` (void)
- Uint16 `setSineState` (Uint16 state)
- Uint16 `getSineState` (Uint16 \*state)
- Uint16 `setSineRmsTarget` (float32 rmsTarget)
- Uint16 `setSineGainTarget` (float32 target)
- Uint16 `getSineGainTarget` (float32 \*target)
- Uint16 `setSineGainStep` (float32 step)
- Uint16 `getSineGainStep` (float32 \*step)

## Variables

- volatile int32 \* `SGENTI_1ch_VOut`

### 5.16.1 Detailed Description

Signal generator functions. `sgInit()` must be called before any other signal generator functions are used. Note that the frequency resolution is determined by the maximum frequency and the step max. For further details, see the signal generator library documentation (Texas Instruments Signal Generator Library Module user's Guide).

#### Warning

This file is included by the file `ISR.asm`.

### 5.16.2 Macro Definition Documentation

#### 5.16.2.1 `#define SIN_CHANNEL AC_STAGE`

Defines which channel enable controls the generator output.

#### 5.16.2.2 `#define SIN_DFLT_GAIN 0.0`

Initial gain setting [0.0, 1.0].

#### 5.16.2.3 `#define SIN_DFLT_RCTFY TRUE`

Initial rectification setting [TRUE | FALSE].

#### 5.16.2.4 `#define SIN_F_ISR 60000000.0f/PWM_1_PRD`

Sampling frequency. I.e. the frequency at which the DPL ISR task runs.



5.16.2.5 `#define SIN_F_ISR_DVSR 6`

Sampling frequency divisor. E.g.  $F\_ISR = 130\text{kHz}$ , but the DPL ISR only runs on every third trigger plus there are two alternating tasks so,  $3 \times 2 = 6$

5.16.2.6 `#define SIN_F_MAX 200u`

Maximum frequency setting (hertz).

5.16.2.7 `#define SIN_F_REQ 50.0`

Required frequency setting (hertz).

5.16.2.8 `#define SIN_OFST 0`

Offset setting  $[-0.5, +0.5]$ , IQ15.

5.16.2.9 `#define SIN_PHSE 0`

Initial phase setting  $[0, 360]$ , IQ16.

## 5.16.3 Function Documentation

5.16.3.1 `void disableSinePhaseOut ( void )`

Disables the use of the AC phase output signal on GPIO12.

5.16.3.2 `void enableSinePhaseOut ( void )`

Enables the use of the AC phase output signal on GPIO12.

5.16.3.3 `uint16_t getSineGainStep ( float32 * step )`

Queries the current gain step setting.

Parameters

<code>out</code>	<code>step</code>	Pointer to location at which to place the query result.
------------------	-------------------	---

Returns

Error status.

5.16.3.4 `uint16_t getSineGainTarget ( float32 * target )`

Queries the current target gain setting.

Parameters

<i>out</i>	<i>target</i>	Pointer to location at which to place the query result.
------------	---------------	---

**Returns**

Error status.

**5.16.3.5 Uint16 getSineState ( Uint16 \* state )**

Queries the current state of the generator output.

**Parameters**

<i>out</i>	<i>state</i>	Pointer to location at which to place the query result {1:ON   0:OFF}.
------------	--------------	--

**Returns**

Error status.

**5.16.3.6 void initSine ( Uint16 enablePhaseOut )**

Sets the initial generator values and disables the output. This function MUST be called before any other signal generator function.

**Parameters**

<i>in</i>	<i>enablePhaseOut</i>	Determines if the AC phase signal is enabled as an output or disabled and switched to an input.
-----------	-----------------------	---

**5.16.3.7 Uint16 setSineGainStep ( float32 step )**

Sets the gain step of the sine signal.

**Parameters**

<i>in</i>	<i>step</i>	Specifies the value of the gain step to be applied.
-----------	-------------	---

**Returns**

Error status.

**5.16.3.8 Uint16 setSineGainTarget ( float32 target )**

Sets the target gain of the signal.

**Parameters**

<i>in</i>	<i>target</i>	Gain target value [0.0, 1.0).
-----------	---------------	-------------------------------

**Returns**

Error status.

**5.16.3.9 Uint16 setSineRmsTarget ( float32 rmsTarget )**

Sets the target RMS voltage of the signal.

**Parameters**

<i>in</i>	<i>rmsTarget</i>	RMS voltage target value.
-----------	------------------	---------------------------

**Returns**

Error status.

**5.16.3.10 Uint16 setSineState ( Uint16 *state* )**

Enables or disables the output of the generator onto the connected net

**Parameters**

<i>in</i>	<i>state</i>	Output enable state {1:ON   0:OFF}.
-----------	--------------	-------------------------------------

**Returns**

Error status.

**5.16.3.11 void updateSineGain ( void )**

Updates the gain value to create a slow-start ramp. This should be called at the same time and similarly to the DC slew update.

**5.16.3.12 void updateSineSignal ( void )**

Generates the next signal data point and loads it onto the VOut terminal. If the point is positive the sign terminal is set, otherwise it is cleared. If rectify is enabled, the VOut value produced will be an absolute value. This is called by the DP\_LIB asm ISR.

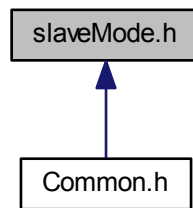
**5.16.4 Variable Documentation****5.16.4.1 volatile int32\* SGENT1\_1ch\_VOut**

Voltage output terminal.

**5.17 slaveMode.h File Reference**

Functions and variables used in detection of the mode of the unit.

This graph shows which files directly or indirectly include this file:



### Typedefs

- typedef enum [slaveMode](#) [slaveMode](#)

### Enumerations

- enum [slaveMode](#)

### Functions

- [slaveMode](#) [detectSlaveMode](#) (void)

### Variables

- [slaveMode](#) [slaveModeStatus](#)

## 5.17.1 Detailed Description

Functions and variables used in detection of the mode of the unit. These functions determine the mode of the burn in unit and should be run after system start up. The detection is managed using GPIO6 and GPIO7. A unit may be in one of four modes: a single unit, a master unit, a slave unit or alternatively a unit may be in the undetected mode which is the initial mode of a unit upon startup.

### Warning

For the mode to be correctly detected the two units should be connected while powered off, then the slave unit should be powered on first, followed by powering on the master unit second. Once detected at startup, any further changes to the connected mode will not be detected and thus the detected mode reported will not reflect such changes to the connection mode of the unit. To change the connection mode of a unit, it, and any slave or master unit must be powered off again.

Also note that, as mentioned, the mode detection algorithm uses GPIO6. However the during initialisation and use of the SPI peripheral GPIO6 is used as a slave service request line. Thus even if the unit mode is not used in conjunction with the SPI functions they should still only be used after the slave mode has been detected.

### 5.17.2 Typedef Documentation

#### 5.17.2.1 typedef enum slaveMode slaveMode

A type to allow specification of the unit mode

### 5.17.3 Enumeration Type Documentation

#### 5.17.3.1 enum slaveMode

The possible unit modes

### 5.17.4 Function Documentation

#### 5.17.4.1 slaveMode detectSlaveMode ( void )

Detects the unit mode. This should only be run once, at startup, to allow the mode to be provided to any other function initialisations that may rely on it.

When a slave unit is powered on the state of GPIO7 is read. If the reading is LOW, this indicates that the unit is a slave and subsequently GPIO6 is set HIGH. If the GPIO7 reading indicated that the unit is a master unit the state of GPIO6 is then read and if it is LOW it determines that the unit is a single unit.

#### Returns

The detected slave mode. If the mode has already been detected the state will be available by referencing the slaveModeStatus variable instead.

### 5.17.5 Variable Documentation

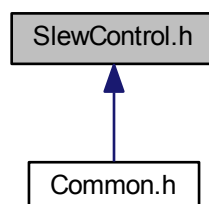
#### 5.17.5.1 slaveMode slaveModeStatus

A global variable that holds the detected mode of the unit, allowing the mode to be queried.

## 5.18 SlewControl.h File Reference

Slew control functions.

This graph shows which files directly or indirectly include this file:



## Functions

- void [updateLoadSlew](#) (void)
- Uint16 [setLoadSlewTarget](#) (loadStage load, float32 target)
- Uint16 [setLoadSlewStep](#) (loadStage load, float32 step)
- Uint16 [setLoadState](#) (loadStage load, Uint16 state)
- Uint16 [scSetLoadStateAll](#) (Uint16 state)
- Uint16 [getLoadSlewTarget](#) (loadStage load, float32 \*target)
- Uint16 [getLoadSlewStep](#) (loadStage load, float32 \*step)
- Uint16 [getLoadState](#) (loadStage load, Uint16 \*state)

### 5.18.1 Detailed Description

Slew control functions.

### 5.18.2 Function Documentation

#### 5.18.2.1 Uint16 getLoadSlewStep ( loadStage load, float32 \* step )

Queries the current slew step size of the specified channel.

##### Parameters

in	<i>load</i>	Specifies the load the setting is to be read from.
out	<i>step</i>	Pointer to location at which to place the query result (amps).

##### Returns

Error status.

#### 5.18.2.2 Uint16 getLoadSlewTarget ( loadStage load, float32 \* target )

Queries the current slew target setting for the specified channel.

##### Parameters

in	<i>load</i>	Specifies the load the setting is to be read from.
out	<i>target</i>	Pointer to location at which to place the query result (amps).

##### Returns

Error status.

#### 5.18.2.3 Uint16 getLoadState ( loadStage load, Uint16 \* state )

Queries the current reference net enable state for the specified channel.

##### Parameters

in	<i>load</i>	Specifies the load the setting is to be read from.
out	<i>state</i>	Pointer to location at which to place the query result (0:OFF   non-zero:ON).

##### Returns

Error status.

#### 5.18.2.4 Uint16 scSetLoadStateAll ( Uint16 *state* )

Sets all channels' reference net enable state.

**Parameters**

<i>in</i>	<i>state</i>	Specifies the reference net state to be applied (0:OFF   non-zero:ON).
-----------	--------------	--

**Returns**

Error status.

**5.18.2.5 Uint16 setLoadSlewStep ( loadStage *load*, float32 *step* )**

Sets the slew step size for the specified load.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load the setting is to be applied to.
<i>in</i>	<i>step</i>	Specifies the value of the slew step size to be applied (amps).

**Returns**

Error status.

**5.18.2.6 Uint16 setLoadSlewTarget ( loadStage *load*, float32 *target* )**

Sets the slew target for the specified load.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load the setting is to be applied to.
<i>in</i>	<i>target</i>	Specifies the value of the slew target to be applied (amps).

**Returns**

Error status.

**5.18.2.7 Uint16 setLoadState ( loadStage *load*, Uint16 *state* )**

Sets the reference net enable state for the specified load.

**Parameters**

<i>in</i>	<i>load</i>	Specifies the load the setting is to be applied to.
<i>in</i>	<i>state</i>	Specifies the reference net state to be applied (0:OFF   non-zero:ON).

**Returns**

Error Status.

**5.18.2.8 void updateLoadSlew ( void )**

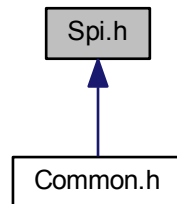
Advances the slew ramps for all loads.



## 5.19 Spi.h File Reference

Serial peripheral interface communications functions.

This graph shows which files directly or indirectly include this file:



### Macros

- #define `SPI_DFLT_BAUD` 0x3D090
- #define `SPI_DFLT_CPOL` 0x00
- #define `SPI_DFLT_CPHA` 0x01
- #define `SPI_FFRX_INTLVL` 0x01
- #define `SPI_FFTX_INTLVL` 0x00
- #define `SPI_FFTX_FILLVL` 0x04

### Typedefs

- typedef enum `spiMode` `spiMode`
- typedef enum `spiLpbk` `spiLpbk`
- typedef enum `transPol` `transPol`
- typedef enum `spiCPha` `spiCPha`

### Enumerations

- enum `spiMode`
- enum `spiLpbk`
- enum `transPol`
- enum `spiCPha`

### Functions

- Uint16 `spiInit` (`spiMode` mode, Uint32 baud, `spiLpbk` loopback, `transPol` cPol, `spiCPha` cPha)
- void `spiTx` (void)

#### 5.19.1 Detailed Description

Serial peripheral interface communications functions. As a master GPIO6 is setup as external interrupt 2 for the slave service request line, set as active low. As a master GPIO6 is setup as an active low output.

**Warning**

The following bridge tracks should be cut on the TI C2000 LaunchPad XL PCB before SPI (or I2C) may be used:

Bridge to Cut	GPIO	PCB Pin
JP5	GPIO32	J2-6
JP7	GPIO33	J2-7
JP8	GPIO16	J6-7
JP10	GPIO17	J6-8

This should result in the following functionality:

Function	GPIO	PCB Pin
I2C_SDAA	GPIO32	J6-7
I2C_SCLA	GPIO33	J6-8
SPI_MOSI	GPIO16	J2-6
SPI_MISO	GPIO17	J2-7
SPI_CLK	GPIO18	J1-18
SPI_STE	GPIO19	J2-1

**5.19.2 Macro Definition Documentation****5.19.2.1 #define SPI\_DFLT\_BAUD 0x3D090**

SPI default baud rate.

**5.19.2.2 #define SPI\_DFLT\_CPHA 0x01**

SPI default clock phase. 0 = Without delay, 1 = with delay.

**5.19.2.3 #define SPI\_DFLT\_CPOL 0x00**

SPI default clock polarity. 0 = Rising edge, 1 = falling edge.

**5.19.2.4 #define SPI\_FFRX\_INTLVL 0x01**

SPI Rx FIFO interrupt level.

**5.19.2.5 #define SPI\_FFTX\_FILLVL 0x04**

SPI Tx FIFO fill level.

**5.19.2.6 #define SPI\_FFTX\_INTLVL 0x00**

SPI Tx FIFO interrupt level.

**5.19.3 Typedef Documentation****5.19.3.1 typedef enum spiCPha spiCPha**

A type to allow specification of the SPI clock phase setting.

**5.19.3.2 typedef enum spiLpbk spiLpbk**

A type to allows specification of the SPI loop-back setting.

## 5.19.3.3 typedef enum spiMode spiMode

A type to allow specification of the SPI mode.

## 5.19.3.4 typedef enum transPol transPol

A type to allow specification of the edge transition polarity setting.

## 5.19.4 Enumeration Type Documentation

## 5.19.4.1 enum spiCPha

The possible SPI clock phase settings.

## 5.19.4.2 enum spiLpbk

The possible SPI loop-back settings.

## 5.19.4.3 enum spiMode

The possible SPI Modes.

## 5.19.4.4 enum transPol

The possible edge transition polarity settings.

## 5.19.5 Function Documentation

## 5.19.5.1 Uint16 spilnit ( spiMode mode, Uint32 baud, spiLpbk loopback, transPol cPol, spiCPha cPha )

Initialises the SPI-A peripheral and relevant interrupts

## Parameters

in	<i>mode</i>	Specifies if the device is in slave or master mode, values are master or slave.
in	<i>baud</i>	Specifies the baud rate to be used minimum value is 117,187, maximum is 3,750,000
in	<i>loopback</i>	Specifies if the SPI module should be run in loop-back mode, values are enable or disable.
in	<i>cPol</i>	Specifies the clock polarity, values are risingEdge or fallingEdge.
in	<i>cPha</i>	Specifies the clock phase, values are withDelay or withoutDelay.

## Returns

Error status.

## Warning

This function will clear any values already in the SPI peripheral registers.  
This function MUST be called before any other public SPI function.

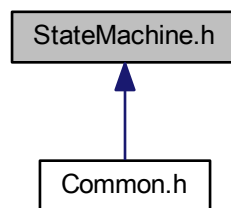
### 5.19.5.2 void spiTx ( void )

Transmits whatever data is on the SCPI output queue if SPI has been selected as the external communications type.

## 5.20 StateMachine.h File Reference

State machine functions.

This graph shows which files directly or indirectly include this file:



### Functions

- void [initStateMachine](#) (void)

### Variables

- void(\* [Alpha\\_State\\_Ptr](#))(void)

### 5.20.1 Detailed Description

State machine functions.

### 5.20.2 Function Documentation

#### 5.20.2.1 void initStateMachine ( void )

Sets up the state machine (including timers) ready for use.

### 5.20.3 Variable Documentation

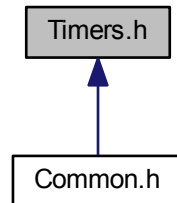
#### 5.20.3.1 void(\* Alpha\_State\_Ptr)(void)

Runs the next iteration of the state machine. Should be called from the main super-loop.

## 5.21 Timers.h File Reference

Real and virtual timer functions.

This graph shows which files directly or indirectly include this file:



### Functions

- void [timersSetupReal](#) (void)

#### 5.21.1 Detailed Description

Real and virtual timer functions. These functions should be run as part of the state machine setup.

See Also

[StateMachine.h](#)

#### 5.21.2 Function Documentation

##### 5.21.2.1 void timersSetupReal ( void )

Sets up the real timers that run the state machine This should be called as part of the state machine initialisation.

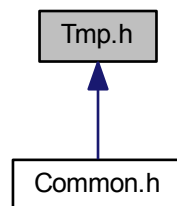
## See Also

smInit()

## 5.22 Tmp.h File Reference

Temperature sensor functions.

This graph shows which files directly or indirectly include this file:



### Macros

- `#define ADC_SW3A_STATE 0x00`
- `#define ADC_SW3B_STATE 0x00`
- `#define ADC_I2C_BASE_ADDR 0x48`
- `#define ADC_I2C_ADDR (ADC_I2C_BASE_ADDR | ADC_SW3A_STATE) | (ADC_SW3B_STATE << 1)`
- `#define ADC_NUM_CHNL 0x08`
- `#define ADC_VREF 5.0`
- `#define ADC_STPS 256`
- `#define TMP_V0C_OFST 0.4`
- `#define TMP_SCL_OFST TMP_V0C_OFST * ADC_STPS / ADC_VREF`
- `#define TMP_E_T_COLD 1.5`

### Functions

- Uint16 `initTemperature` (void)
- Uint16 `readTemperature` (Uint16 chnl, float32 \*tmpDest)

#### 5.22.1 Detailed Description

Temperature sensor functions. The temperature sensor (MCP9701) output is read via an external ADC (ADS7830) that is connected to the I2C bus at address 10010xx where 'xx' is dependent upon the state of switch 3 (SW3) on the Ctrl PCB, which the user should record in the macros `ADC_SW3A_STATE` and `ADC_SW3B_STATE`.

All temperatures are in degrees Celsius.

#### Warning

Before any temperature functions can be used the I2C peripheral **MUST** be initialised, using `i2cInit()`, and `initTemperature()` must be run - `initTemperature()` will require the interrupts to be enabled globally.

See Also

i2cInit()

ADS7830 Ch #	Signal	Chnl selector	ADC Sel Code
CH0	TEMP1	load1 (0)	0
CH1	TEMP2	load2 (1)	4
CH2	TEMP3	load3 (2)	1
CH3	TEMP4	load4 (3)	5
CH4	TEMP5	XFMR_STAGE (4)	2
CH5	TEMP6	AC_STAGE (5)	6
CH6	TEMP_EXT1	EXT_1 (6)	3
CH7	TEMP_EXT2	EXT_2 (7)	7

## 5.22.2 Macro Definition Documentation

**5.22.2.1 #define ADC\_I2C\_ADDR (ADC\_I2C\_BASE\_ADDR | ADC\_SW3A\_STATE) | (ADC\_SW3B\_STATE << 1)**

ADS7830 ADC complete I2C address.

**5.22.2.2 #define ADC\_I2C\_BASE\_ADDR 0x48**

ADS7830 ADC base I2C address.

**5.22.2.3 #define ADC\_NUM\_CHNL 0x08**

ADS7830 ADC number of temperature channels.

**5.22.2.4 #define ADC\_STPS 256**

ADS7830 ADC number of steps.

**5.22.2.5 #define ADC\_SW3A\_STATE 0x00**

The state of switch 3a; ON = 0x01, OFF = 0x00.

**5.22.2.6 #define ADC\_SW3B\_STATE 0x00**

The state of switch 3b; ON = 0x01, OFF = 0x00.

**5.22.2.7 #define ADC\_VREF 5.0**

ADS7830 ADC reference voltage (volts).

**5.22.2.8 #define TMP\_E\_T\_COLD 1.5**

MCP9701 Error at lowest operating temperature ( ° C), calculated as shown in Microchip AN1001.

**5.22.2.9 #define TMP\_SCL\_OFST TMP\_V0C\_OFST \* ADC\_STPS / ADC\_VREF**

Scaled temperature offset.

5.22.2.10 `#define TMP_V0C_OFST 0.4`

MCP9701 Temperature sensor  $V_{0^{\circ}C}$  (volts).

### 5.22.3 Function Documentation

5.22.3.1 `Uint16 initTemperature ( void )`

Initialises the system for temperature readings. The I2C peripheral must be initialised before this function is used

See Also

`i2cInit()`.

Returns

Error status.

5.22.3.2 `Uint16 readTemperature ( Uint16 chnl, float32 * tmpDest )`

Queries the current temperature on the specified ADC channel.

Parameters

<code>in</code>	<code><i>chnl</i></code>	Specifies the channel the temperature is to be read from.
<code>out</code>	<code><i>tmpDest</i></code>	Address of the memory location at which to place the query result ( $^{\circ}C$ ).

Returns

Error status.



# Index

AC\_I\_SCALE  
    Settings.h, [80](#)  
AC\_IRMSLV\_MAX  
    Settings.h, [80](#)  
AC\_OCP\_TRIP  
    Ocp.h, [56](#)  
AC\_OPP\_TRIP  
    Opp.h, [61](#)  
AC\_OTP\_TRIP  
    Otp.h, [64](#)  
AC\_OVP\_TRIP  
    Ovp.h, [70](#)  
AC\_PWRLVL\_FIX  
    Settings.h, [80](#)  
AC\_STAGE  
    MacroNets.h, [51](#)  
AC\_V\_R1  
    Settings.h, [80](#)  
AC\_V\_R2  
    Settings.h, [80](#)  
AC\_VRMSLV\_MAX  
    Settings.h, [80](#)  
ADC\_I2C\_ADDR  
    Tmp.h, [99](#)  
ADC\_I2C\_BASE\_ADDR  
    Tmp.h, [99](#)  
ADC\_NUM\_CHNL  
    Tmp.h, [99](#)  
ADC\_STPS  
    Tmp.h, [99](#)  
ADC\_SW3A\_STATE  
    Tmp.h, [99](#)  
ADC\_SW3B\_STATE  
    Tmp.h, [99](#)  
ADC\_VREF  
    Tmp.h, [99](#)  
ADCDRV\_1ch\_Rlt1  
    Adc.h, [31](#)  
ADCDRV\_1ch\_Rlt10  
    Adc.h, [31](#)  
ADCDRV\_1ch\_Rlt11  
    Adc.h, [31](#)  
ADCDRV\_1ch\_Rlt12  
    Adc.h, [31](#)  
ADCDRV\_1ch\_Rlt13  
    Adc.h, [31](#)  
ADCDRV\_1ch\_Rlt2  
    Adc.h, [31](#)  
ADCDRV\_1ch\_Rlt3  
    Adc.h, [32](#)  
ADCDRV\_1ch\_Rlt4  
    Adc.h, [32](#)  
ADCDRV\_1ch\_Rlt5  
    Adc.h, [32](#)  
ADCDRV\_1ch\_Rlt6  
    Adc.h, [32](#)  
ADCDRV\_1ch\_Rlt7  
    Adc.h, [32](#)  
ADCDRV\_1ch\_Rlt8  
    Adc.h, [32](#)  
ADCDRV\_1ch\_Rlt9  
    Adc.h, [32](#)  
ALL\_DISABLED\_WORD  
    EnableCtrl.h, [43](#)  
acFrequency  
    acStageSettings, [20](#)  
acCoefs  
    Cntl.h, [36](#)  
acNets  
    MacroNets.h, [54](#)  
acSettings  
    MacroNets.h, [54](#)  
acStageNets, [19](#)  
    iFdbkNet, [19](#)  
    iFiltOutNet, [19](#)  
    iRefNet, [19](#)  
    MacroNets.h, [52](#)  
    vFdbkNet, [19](#)  
    vRefNet, [19](#)  
acStageSettings, [20](#)  
    acFrequency, [20](#)  
    enable, [20](#)  
    gainRate, [20](#)  
    gainTarget, [20](#)  
    iMaxRms, [20](#)  
    iScale, [21](#)  
    MacroNets.h, [52](#)  
    mode, [21](#)  
    ocpLevel, [21](#)  
    otpLevel, [21](#)  
    ovpLevel, [21](#)  
    vGainLmt, [21](#)  
    vMaxRms, [21](#)  
    vScale, [21](#)  
acVCoefs  
    Cntl.h, [36](#)  
Adc.h, [29](#)  
    ADCDRV\_1ch\_Rlt1, [31](#)

- ADCDRV\_1ch\_Rlt10, [31](#)
- ADCDRV\_1ch\_Rlt11, [31](#)
- ADCDRV\_1ch\_Rlt12, [31](#)
- ADCDRV\_1ch\_Rlt13, [31](#)
- ADCDRV\_1ch\_Rlt2, [31](#)
- ADCDRV\_1ch\_Rlt3, [32](#)
- ADCDRV\_1ch\_Rlt4, [32](#)
- ADCDRV\_1ch\_Rlt5, [32](#)
- ADCDRV\_1ch\_Rlt6, [32](#)
- ADCDRV\_1ch\_Rlt7, [32](#)
- ADCDRV\_1ch\_Rlt8, [32](#)
- ADCDRV\_1ch\_Rlt9, [32](#)
- getDcHvVoltage, [30](#)
- getDcMidCurrent, [30](#)
- getDcMidVoltage, [30](#)
- getLoadCurrent, [30](#)
- getLoadVoltage, [31](#)
- initAdc, [31](#)
- Alpha\_State\_Ptr
  - StateMachine.h, [96](#)
- cA1
  - Cntl.h, [34](#)
- cA2
  - Cntl.h, [34](#)
- cB0
  - Cntl.h, [34](#)
- cB1
  - Cntl.h, [34](#)
- cB2
  - Cntl.h, [34](#)
- cMax
  - Cntl.h, [34](#)
- cMin
  - Cntl.h, [34](#)
- CHANNEL\_OOB
  - Settings.h, [80](#)
- CNTL\_2P2Z\_Coef1
  - Cntl.h, [36](#)
- CNTL\_2P2Z\_Coef2
  - Cntl.h, [36](#)
- CNTL\_2P2Z\_Coef3
  - Cntl.h, [36](#)
- CNTL\_2P2Z\_Coef4
  - Cntl.h, [36](#)
- CNTL\_2P2Z\_Coef5
  - Cntl.h, [36](#)
- CNTL\_2P2Z\_Coef6
  - Cntl.h, [36](#)
- CNTL\_2P2Z\_Fdbk1
  - Cntl.h, [36](#)
- CNTL\_2P2Z\_Fdbk2
  - Cntl.h, [36](#)
- CNTL\_2P2Z\_Fdbk3
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Fdbk4
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Fdbk5
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Fdbk6
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Out1
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Out2
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Out3
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Out4
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Out5
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Out6
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Ref1
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Ref2
  - Cntl.h, [37](#)
- CNTL\_2P2Z\_Ref3
  - Cntl.h, [38](#)
- CNTL\_2P2Z\_Ref4
  - Cntl.h, [38](#)
- CNTL\_2P2Z\_Ref5
  - Cntl.h, [38](#)
- CNTL\_2P2Z\_Ref6
  - Cntl.h, [38](#)
- cfType
  - Cntl.h, [34](#)
- checkAcOpp
  - Opp.h, [62](#)
- checkAcOtp
  - Otp.h, [65](#)
- checkAcOvp
  - Ovp.h, [70](#)
- checkDcHvOvp
  - Ovp.h, [70](#)
- checkDcMidOcp
  - Ocp.h, [56](#)
- checkDcOtp
  - Otp.h, [65](#)
- checkExtOtp
  - Otp.h, [65](#)
- checkLoadOcp
  - Ocp.h, [56](#)
- checkLoadOpp
  - Opp.h, [62](#)
- checkLoadOtp
  - Otp.h, [65](#)
- checkLoadOvp
  - Ovp.h, [71](#)
- circuitSection
  - EnableCtrl.h, [45](#)
- clearAcOcp
  - Ocp.h, [57](#)
- clearAcOpp
  - Opp.h, [62](#)
- clearAcOtp
  - Otp.h, [65](#)

clearAcOvp  
     Ovp.h, 71  
 clearDcHvOvp  
     Ovp.h, 71  
 clearDcMidOcp  
     Ocp.h, 57  
 clearDcMidOvp  
     Ovp.h, 71  
 clearDcOtp  
     Otp.h, 65  
 clearExtOtp  
     Otp.h, 66  
 clearLoadOcp  
     Ocp.h, 57  
 clearLoadOpp  
     Opp.h, 62  
 clearLoadOtp  
     Otp.h, 66  
 clearLoadOvp  
     Ovp.h, 71  
 Cntl.h  
     cA1, 34  
     cA2, 34  
     cB0, 34  
     cB1, 34  
     cB2, 34  
     cMax, 34  
     cMin, 34  
 Cntl.h, 32  
     acICoefs, 36  
     acVCoefs, 36  
     CNTL\_2P2Z\_Coef1, 36  
     CNTL\_2P2Z\_Coef2, 36  
     CNTL\_2P2Z\_Coef3, 36  
     CNTL\_2P2Z\_Coef4, 36  
     CNTL\_2P2Z\_Coef5, 36  
     CNTL\_2P2Z\_Coef6, 36  
     CNTL\_2P2Z\_Fdbk1, 36  
     CNTL\_2P2Z\_Fdbk2, 36  
     CNTL\_2P2Z\_Fdbk3, 37  
     CNTL\_2P2Z\_Fdbk4, 37  
     CNTL\_2P2Z\_Fdbk5, 37  
     CNTL\_2P2Z\_Fdbk6, 37  
     CNTL\_2P2Z\_Out1, 37  
     CNTL\_2P2Z\_Out2, 37  
     CNTL\_2P2Z\_Out3, 37  
     CNTL\_2P2Z\_Out4, 37  
     CNTL\_2P2Z\_Out5, 37  
     CNTL\_2P2Z\_Out6, 37  
     CNTL\_2P2Z\_Ref1, 37  
     CNTL\_2P2Z\_Ref2, 37  
     CNTL\_2P2Z\_Ref3, 38  
     CNTL\_2P2Z\_Ref4, 38  
     CNTL\_2P2Z\_Ref5, 38  
     CNTL\_2P2Z\_Ref6, 38  
     cfType, 34  
     coefNum, 34  
     getAcICoef, 34  
     getAcVCoef, 34  
     getLoadICoef, 34  
     initCoefs, 35  
     loadICoefs, 38  
     setAcICoef, 35  
     setAcVCoef, 35  
     setLoadICoef, 35  
 coefNum  
     Cntl.h, 34  
 Common.h, 38  
 Comparator.h, 39  
     getAcDac, 40  
     initAcComparator, 40  
     initDcComparator, 40  
     initTripZone, 41  
     rstAcTripzone, 41  
     rstDcTripzone, 41  
     setAcDac, 41  
 DC\_OTP\_TRIP  
     Otp.h, 64  
 DCHV\_OCP\_TRIP  
     Ocp.h, 56  
 DCHV\_OPP\_TRIP  
     Opp.h, 61  
 DCHV\_OVP\_TRIP  
     Ovp.h, 70  
 DCHV\_UVLCK\_LMT  
     Settings.h, 80  
 DCMID\_OCP\_TRIP  
     Ocp.h, 56  
 DCMID\_OPP\_TRIP  
     Opp.h, 61  
 DCMID\_OVP\_TRIP  
     Ovp.h, 70  
 DCMID\_VDCLVL\_FIX  
     Settings.h, 80  
 DCMIDV\_STRTUP\_LMT  
     Settings.h, 80  
 DEBUG  
     Settings.h, 80  
 DPL\_ISR  
     Pwm.h, 75  
 DUAL\_CNTL\_AC  
     Settings.h, 80  
 detectSlaveMode  
     slaveMode.h, 89  
 disableCircuit  
     EnableCtrl.h, 45  
 disableSinePhaseOut  
     SineGen.h, 85  
 EXT1\_OTP\_TRIP  
     Otp.h, 64  
 EXT2\_OTP\_TRIP  
     Otp.h, 64  
 EXT\_1  
     MacroNets.h, 51  
 EXT\_2

- MacroNets.h, 51
- ecSection
  - EnableCtrl.h, 45
- enable
  - acStageSettings, 20
  - loadStageSettings, 24
  - xfrmStageSettings, 27
- enableAllFlag
  - MacroNets.h, 54
- enableCircuit
  - EnableCtrl.h, 45
- EnableCtrl.h, 41
  - ALL\_DISABLED\_WORD, 43
  - circuitSection, 45
  - disableCircuit, 45
  - ecSection, 45
  - enableCircuit, 45
  - getCurrentState, 45
  - IOE\_DEFVAL\_ADDR, 43
  - IOE\_GPINTEN\_ADDR, 43
  - IOE\_GPIO\_ADDR, 43
  - IOE\_GPPU\_ADDR, 43
  - IOE\_I2C\_ADDR, 43
  - IOE\_I2C\_BASE\_ADDR, 44
  - IOE\_INTCAP\_ADDR, 44
  - IOE\_INTCON\_ADDR, 44
  - IOE\_INTF\_ADDR, 44
  - IOE\_IOCON\_ADDR, 44
  - IOE\_IODIR\_ADDR, 44
  - IOE\_IPOL\_ADDR, 44
  - IOE\_MAX\_CHNL, 44
  - IOE\_OLAT\_ADDR, 44
  - IOE\_SW4A\_STATE, 44
  - IOE\_SW4B\_STATE, 44
  - initEnableControl, 45
  - resetEnableControl, 46
- enableSinePhaseOut
  - SineGen.h, 85
- ext1
  - MacroNets.h, 53
- ext1OtpLevel
  - extDeviceSettings, 22
- ext2
  - MacroNets.h, 53
- ext2OtpLevel
  - extDeviceSettings, 22
- extDeviceSettings, 21
  - ext1OtpLevel, 22
  - ext2OtpLevel, 22
  - extFanEnable, 22
  - extPsuEnable, 22
  - MacroNets.h, 52
- extFanEnable
  - extDeviceSettings, 22
- extPsuEnable
  - extDeviceSettings, 22
- extSelect
  - MacroNets.h, 52, 53
- extSettings
  - MacroNets.h, 54
- gainRate
  - acStageSettings, 20
- gainTarget
  - acStageSettings, 20
- getAcDac
  - Comparator.h, 40
- getAcClCoef
  - Cntl.h, 34
- getAcOcpLevel
  - Ocp.h, 57
- getAcOcpState
  - Ocp.h, 57
- getAcOppState
  - Opp.h, 62
- getAcOtpLevel
  - Otp.h, 66
- getAcOtpState
  - Otp.h, 66
- getAcOvpLevel
  - Ovp.h, 71
- getAcOvpState
  - Ovp.h, 72
- getAcVCoef
  - Cntl.h, 34
- getCurrentState
  - EnableCtrl.h, 45
- getDcHvOvpLevel
  - Ovp.h, 72
- getDcHvOvpState
  - Ovp.h, 72
- getDcHvVoltage
  - Adc.h, 30
- getDcMidCurrent
  - Adc.h, 30
- getDcMidOcpLevel
  - Ocp.h, 57
- getDcMidOcpState
  - Ocp.h, 59
- getDcMidOvpState
  - Ovp.h, 72
- getDcMidVoltage
  - Adc.h, 30
- getDcOtpLevel
  - Otp.h, 66
- getDcOtpState
  - Otp.h, 67
- getExtOtpLevel
  - Otp.h, 67
- getExtOtpState
  - Otp.h, 67
- getLoadCurrent
  - Adc.h, 30
- getLoadIcoef
  - Cntl.h, 34
- getLoadOcpLevel
  - Ocp.h, 59

getLoadOcpState  
     Ocp.h, 59  
 getLoadOppState  
     Opp.h, 62  
 getLoadOtpLevel  
     Otp.h, 67  
 getLoadOtpState  
     Otp.h, 67  
 getLoadOvpState  
     Ovp.h, 72  
 getLoadSlewStep  
     SlewControl.h, 90  
 getLoadSlewTarget  
     SlewControl.h, 90  
 getLoadState  
     SlewControl.h, 90  
 getLoadVoltage  
     Adc.h, 31  
 getSineGainStep  
     SineGen.h, 85  
 getSineGainTarget  
     SineGen.h, 85  
 getSineState  
     SineGen.h, 86  
  
 HV\_V\_R1  
     Settings.h, 80  
 HV\_V\_R2  
     Settings.h, 81  
 hvOvpLevel  
     xfmrStageSettings, 27  
 hvVMax  
     xfmrStageSettings, 27  
 hvVScale  
     xfmrStageSettings, 27  
 hvVSnsNet  
     xfmrStageNets, 26  
  
 I2C\_ARDY\_ISRC  
     I2c.h, 47  
 I2C\_BUS\_BUSY  
     Settings.h, 81  
 I2C\_CLR\_AL\_BIT  
     I2c.h, 47  
 I2C\_CLR\_ARDY\_BIT  
     I2c.h, 47  
 I2C\_CLR\_NACK\_BIT  
     I2c.h, 48  
 I2C\_CLR\_RRDY\_BIT  
     I2c.h, 48  
 I2C\_CLR\_SCD\_BIT  
     I2c.h, 48  
 I2C\_INVALID\_ISRC  
     Settings.h, 81  
 I2C\_MAX\_BUFFER\_SIZE  
     I2c.h, 48  
 I2C\_MAX\_PTR\_SIZE  
     I2c.h, 48  
 I2C\_MSGSTAT\_INACTIVE  
     I2c.h, 48  
 I2C\_MSGSTAT\_RESTART  
     I2c.h, 48  
 I2C\_READ\_WRONG\_MSG  
     Settings.h, 81  
 I2C\_SCD\_ISRC  
     I2c.h, 49  
 I2C\_STP\_NOT\_READY  
     Settings.h, 81  
 I2C\_WRITE\_WRONG\_MSG  
     Settings.h, 81  
 I2c.h, 46  
     I2C\_ARDY\_ISRC, 47  
     I2C\_CLR\_AL\_BIT, 47  
     I2C\_CLR\_ARDY\_BIT, 47  
     I2C\_CLR\_NACK\_BIT, 48  
     I2C\_CLR\_RRDY\_BIT, 48  
     I2C\_CLR\_SCD\_BIT, 48  
     I2C\_MAX\_BUFFER\_SIZE, 48  
     I2C\_MAX\_PTR\_SIZE, 48  
     I2C\_MSGSTAT\_INACTIVE, 48  
     I2C\_MSGSTAT\_RESTART, 48  
     I2C\_SCD\_ISRC, 49  
     i2cPopMsg, 49  
     i2cRead, 49  
     i2cWrite, 49  
     initI2c, 49  
 i2cMsg, 22  
     msgBuffer, 23  
     msgStatus, 23  
     numOfBytes, 23  
     numSlavePtrBytes, 23  
     slaveAddress, 23  
     slavePtrAddrHigh, 23  
     slavePtrAddrLow, 23  
 i2cPopMsg  
     I2c.h, 49  
 i2cRead  
     I2c.h, 49  
 i2cWrite  
     I2c.h, 49  
 iFdbkNet  
     acStageNets, 19  
     loadStageNets, 24  
 iFiltOutNet  
     acStageNets, 19  
     loadStageNets, 24  
 iMax  
     loadStageSettings, 24  
     xfmrStageSettings, 27  
 iMaxRms  
     acStageSettings, 20  
 INCR\_BUILD  
     Settings.h, 81  
 IOE\_DEFVAL\_ADDR  
     EnableCtrl.h, 43  
 IOE\_GPINTEN\_ADDR  
     EnableCtrl.h, 43

---

IOE\_GPIO\_ADDR  
     EnableCtrl.h, 43  
 IOE\_GPPU\_ADDR  
     EnableCtrl.h, 43  
 IOE\_I2C\_ADDR  
     EnableCtrl.h, 43  
 IOE\_I2C\_BASE\_ADDR  
     EnableCtrl.h, 44  
 IOE\_INTCAP\_ADDR  
     EnableCtrl.h, 44  
 IOE\_INTCON\_ADDR  
     EnableCtrl.h, 44  
 IOE\_INTF\_ADDR  
     EnableCtrl.h, 44  
 IOE\_IOCON\_ADDR  
     EnableCtrl.h, 44  
 IOE\_IODIR\_ADDR  
     EnableCtrl.h, 44  
 IOE\_IPOL\_ADDR  
     EnableCtrl.h, 44  
 IOE\_MAX\_CHNL  
     EnableCtrl.h, 44  
 IOE\_OLAT\_ADDR  
     EnableCtrl.h, 44  
 IOE\_SW4A\_STATE  
     EnableCtrl.h, 44  
 IOE\_SW4B\_STATE  
     EnableCtrl.h, 44  
 iRefNet  
     acStageNets, 19  
     loadStageNets, 24  
 iScale  
     acStageSettings, 21  
     loadStageSettings, 25  
     xfmrStageSettings, 27  
 iSnsNet  
     xfmrStageNets, 26  
 initAcComparator  
     Comparator.h, 40  
 initAdc  
     Adc.h, 31  
 initCoefs  
     Cntl.h, 35  
 initDcComparator  
     Comparator.h, 40  
 initEnableControl  
     EnableCtrl.h, 45  
 initI2c  
     I2c.h, 49  
 initPwm  
     Pwm.h, 75  
 initSine  
     SineGen.h, 86  
 initStateMachine  
     StateMachine.h, 96  
 initTemperature  
     Tmp.h, 100  
 initTripZone  
     Comparator.h, 41  
 LOAD1\_OCP\_TRIP  
     Ocp.h, 56  
 LOAD1\_OPP\_TRIP  
     Opp.h, 61  
 LOAD1\_OTP\_TRIP  
     Otp.h, 64  
 LOAD1\_OVP\_TRIP  
     Ovp.h, 70  
 LOAD2\_OCP\_TRIP  
     Ocp.h, 56  
 LOAD2\_OPP\_TRIP  
     Opp.h, 61  
 LOAD2\_OTP\_TRIP  
     Otp.h, 64  
 LOAD2\_OVP\_TRIP  
     Ovp.h, 70  
 LOAD3\_OCP\_TRIP  
     Ocp.h, 56  
 LOAD3\_OPP\_TRIP  
     Opp.h, 61  
 LOAD3\_OTP\_TRIP  
     Otp.h, 64  
 LOAD3\_OVP\_TRIP  
     Ovp.h, 70  
 LOAD4\_OCP\_TRIP  
     Ocp.h, 56  
 LOAD4\_OPP\_TRIP  
     Opp.h, 61  
 LOAD4\_OTP\_TRIP  
     Otp.h, 64  
 LOAD4\_OVP\_TRIP  
     Ovp.h, 70  
 LOAD\_0  
     MacroNets.h, 51  
 LOAD\_1  
     MacroNets.h, 52  
 LOAD\_2  
     MacroNets.h, 52  
 LOAD\_3  
     MacroNets.h, 52  
 LOAD\_I\_SCALE  
     Settings.h, 81  
 LOAD\_IDCLVL\_MAX  
     Settings.h, 81  
 LOAD\_PWRLVL\_FIX  
     Settings.h, 81  
 LOAD\_V\_R1  
     Settings.h, 81  
 LOAD\_V\_R2  
     Settings.h, 81  
 LOAD\_VDCLVL\_FIX  
     Settings.h, 82  
 LOG\_SIN  
     Settings.h, 82  
 load1  
     MacroNets.h, 53  
 load2

---

- MacroNets.h, 53
- load3
  - MacroNets.h, 53
- load4
  - MacroNets.h, 53
- loadlCoefs
  - Cntl.h, 38
- loadNets
  - MacroNets.h, 54
- loadSettings
  - MacroNets.h, 54
- loadStage
  - MacroNets.h, 52, 53
- loadStageNets, 23
  - iFdbkNet, 24
  - iFiltOutNet, 24
  - iRefNet, 24
  - MacroNets.h, 52
  - vFdbkNet, 24
- loadStageSettings, 24
  - enable, 24
  - iMax, 24
  - iScale, 25
  - MacroNets.h, 52
  - ocpLevel, 25
  - oppLevel, 25
  - otpLevel, 25
  - ovpLevel, 25
  - slewRate, 25
  - slewTarget, 25
  - vMax, 25
  - vScale, 25
- MID\_I\_SCALE
  - Settings.h, 82
- MID\_V\_R1
  - Settings.h, 82
- MID\_V\_R2
  - Settings.h, 82
- MacroNets.h
  - ext1, 53
  - ext2, 53
  - load1, 53
  - load2, 53
  - load3, 53
  - load4, 53
- MacroNets.h, 50
  - AC\_STAGE, 51
  - acNets, 54
  - acSettings, 54
  - acStageNets, 52
  - acStageSettings, 52
  - EXT\_1, 51
  - EXT\_2, 51
  - enableAllFlag, 54
  - extDeviceSettings, 52
  - extSelect, 52, 53
  - extSettings, 54
  - LOAD\_0, 51
  - LOAD\_1, 52
  - LOAD\_2, 52
  - LOAD\_3, 52
  - loadNets, 54
  - loadSettings, 54
  - loadStage, 52, 53
  - loadStageNets, 52
  - loadStageSettings, 52
  - runAll, 53
  - setupNets, 53
  - stopAll, 54
  - stopAllFlag, 54
  - XFMR\_STAGE, 52
  - xfrmNets, 54
  - xfrmSettings, 54
  - xfrmStageNets, 52
  - xfrmStageSettings, 53
- midVMax
  - xfrmStageSettings, 27
- midVScale
  - xfrmStageSettings, 27
- midVSnsNet
  - xfrmStageNets, 26
- mode
  - acStageSettings, 21
- msgBuffer
  - i2cMsg, 23
- msgStatus
  - i2cMsg, 23
- numOfBytes
  - i2cMsg, 23
- numSlavePtrBytes
  - i2cMsg, 23
- OCP\_TRIP
  - Settings.h, 82
- OPP\_TRIP
  - Settings.h, 82
- OTP\_TRIP
  - Settings.h, 82
- OVP\_TRIP
  - Settings.h, 82
- Ocp.h, 55
  - AC\_OCP\_TRIP, 56
  - checkDcMidOcp, 56
  - checkLoadOcp, 56
  - clearAcOcp, 57
  - clearDcMidOcp, 57
  - clearLoadOcp, 57
  - DCHV\_OCP\_TRIP, 56
  - DCMID\_OCP\_TRIP, 56
  - getAcOcpLevel, 57
  - getAcOcpState, 57
  - getDcMidOcpLevel, 57
  - getDcMidOcpState, 59
  - getLoadOcpLevel, 59
  - getLoadOcpState, 59
  - LOAD1\_OCP\_TRIP, 56

- LOAD2\_OCP\_TRIP, 56
- LOAD3\_OCP\_TRIP, 56
- LOAD4\_OCP\_TRIP, 56
- setAcOcpLevel, 59
- setDcMidOcpLevel, 59
- setLoadOcpLevel, 60
- tripAcOcp, 60
- ocpLevel
  - acStageSettings, 21
  - loadStageSettings, 25
  - xfmrStageSettings, 27
- Opp.h, 60
  - AC\_OPP\_TRIP, 61
  - checkAcOpp, 62
  - checkLoadOpp, 62
  - clearAcOpp, 62
  - clearLoadOpp, 62
  - DCHV\_OPP\_TRIP, 61
  - DCMID\_OPP\_TRIP, 61
  - getAcOppState, 62
  - getLoadOppState, 62
  - LOAD1\_OPP\_TRIP, 61
  - LOAD2\_OPP\_TRIP, 61
  - LOAD3\_OPP\_TRIP, 61
  - LOAD4\_OPP\_TRIP, 61
- oppLevel
  - loadStageSettings, 25
- Otp.h, 63
  - AC\_OTP\_TRIP, 64
  - checkAcOtp, 65
  - checkDcOtp, 65
  - checkExtOtp, 65
  - checkLoadOtp, 65
  - clearAcOtp, 65
  - clearDcOtp, 65
  - clearExtOtp, 66
  - clearLoadOtp, 66
  - DC\_OTP\_TRIP, 64
  - EXT1\_OTP\_TRIP, 64
  - EXT2\_OTP\_TRIP, 64
  - getAcOtpLevel, 66
  - getAcOtpState, 66
  - getDcOtpLevel, 66
  - getDcOtpState, 67
  - getExtOtpLevel, 67
  - getExtOtpState, 67
  - getLoadOtpLevel, 67
  - getLoadOtpState, 67
  - LOAD1\_OTP\_TRIP, 64
  - LOAD2\_OTP\_TRIP, 64
  - LOAD3\_OTP\_TRIP, 64
  - LOAD4\_OTP\_TRIP, 64
  - RSRVD\_OTP\_TRIP, 64
  - setAcOtpLevel, 68
  - setDcOtpLevel, 68
  - setExtOtpLevel, 68
  - setLoadOtpLevel, 68
- otpLevel
  - acStageSettings, 21
  - loadStageSettings, 25
  - xfmrStageSettings, 27
- Ovp.h, 69
  - AC\_OVP\_TRIP, 70
  - checkAcOvp, 70
  - checkDcHvOvp, 70
  - checkLoadOvp, 71
  - clearAcOvp, 71
  - clearDcHvOvp, 71
  - clearDcMidOvp, 71
  - clearLoadOvp, 71
  - DCHV\_OVP\_TRIP, 70
  - DCMID\_OVP\_TRIP, 70
  - getAcOvpLevel, 71
  - getAcOvpState, 72
  - getDcHvOvpLevel, 72
  - getDcHvOvpState, 72
  - getDcMidOvpState, 72
  - getLoadOvpState, 72
  - LOAD1\_OVP\_TRIP, 70
  - LOAD2\_OVP\_TRIP, 70
  - LOAD3\_OVP\_TRIP, 70
  - LOAD4\_OVP\_TRIP, 70
  - setAcOvpLevel, 73
  - setDcHvOvpLevel, 73
  - tripDcMidOvp, 73
- ovpLevel
  - acStageSettings, 21
  - loadStageSettings, 25
- PWM\_1\_PRD
  - Pwm.h, 74
- PWM\_2\_PRD
  - Pwm.h, 74
- PWM\_3\_PRD
  - Pwm.h, 75
- PWMDRV\_2ch\_UpCnt\_Duty1A
  - Pwm.h, 75
- PWMDRV\_2ch\_UpCnt\_Duty1B
  - Pwm.h, 75
- PWMDRV\_2ch\_UpCnt\_Duty2A
  - Pwm.h, 75
- PWMDRV\_2ch\_UpCnt\_Duty2B
  - Pwm.h, 75
- PWMDRV\_2ch\_UpCnt\_Duty3A
  - Pwm.h, 75
- PWMDRV\_2ch\_UpCnt\_Duty3B
  - Pwm.h, 75
- Pwm.h, 74
  - DPL\_ISR, 75
  - initPwm, 75
  - PWM\_1\_PRD, 74
  - PWM\_2\_PRD, 74
  - PWM\_3\_PRD, 75
  - PWMDRV\_2ch\_UpCnt\_Duty1A, 75
  - PWMDRV\_2ch\_UpCnt\_Duty1B, 75
  - PWMDRV\_2ch\_UpCnt\_Duty2A, 75
  - PWMDRV\_2ch\_UpCnt\_Duty2B, 75



PWMDRV\_2ch\_UpCnt\_Duty3A, 75  
 PWMDRV\_2ch\_UpCnt\_Duty3B, 75  
 pwmDutyNet  
     xfmrStageNets, 26  
  
 RECP\_SQRT\_2  
     Settings.h, 82  
 RSRVD\_OTP\_TRIP  
     Otp.h, 64  
 readTemperature  
     Tmp.h, 100  
 resetEnableControl  
     EnableCtrl.h, 46  
 rstAcTripzone  
     Comparator.h, 41  
 rstDcTripzone  
     Comparator.h, 41  
 runAll  
     MacroNets.h, 53  
  
 SCIBAUD\_MIN  
     Sci.h, 76  
 SCIFFRX\_INT\_LVL  
     Sci.h, 76  
 SCIFFTX\_FILL\_LVL  
     Sci.h, 76  
 SCIFFTX\_INT\_LVL  
     Sci.h, 77  
 SCPI\_specificCmds.h, 77  
 SGENTI\_1ch\_VOut  
     SineGen.h, 87  
 SIN\_CHANNEL  
     SineGen.h, 84  
 SIN\_DFLT\_GAIN  
     SineGen.h, 84  
 SIN\_DFLT\_RCTFY  
     SineGen.h, 84  
 SIN\_F\_ISR  
     SineGen.h, 84  
 SIN\_F\_ISR\_DVSR  
     SineGen.h, 84  
 SIN\_F\_MAX  
     SineGen.h, 85  
 SIN\_F\_REQ  
     SineGen.h, 85  
 SIN\_OFST  
     SineGen.h, 85  
 SIN\_PHSE  
     SineGen.h, 85  
 SPI\_DFLT\_BAUD  
     Spi.h, 94  
 SPI\_DFLT\_CPHA  
     Spi.h, 94  
 SPI\_DFLT\_CPOL  
     Spi.h, 94  
 SPI\_FFRX\_INTLVL  
     Spi.h, 94  
 SPI\_FFTX\_FILLVL  
     Spi.h, 94  
  
 SPI\_FFTX\_INTLVL  
     Spi.h, 94  
 SQRT\_2  
     Settings.h, 82  
 scSetLoadStateAll  
     SlewControl.h, 90  
 Sci.h, 75  
     SCIBAUD\_MIN, 76  
     SCIFFRX\_INT\_LVL, 76  
     SCIFFTX\_FILL\_LVL, 76  
     SCIFFTX\_INT\_LVL, 77  
     sciInit, 77  
     sciTx, 77  
 sciInit  
     Sci.h, 77  
 sciTx  
     Sci.h, 77  
 setAcDac  
     Comparator.h, 41  
 setAcCoef  
     Cntl.h, 35  
 setAcOcpLevel  
     Ocp.h, 59  
 setAcOtpLevel  
     Otp.h, 68  
 setAcOvpLevel  
     Ovp.h, 73  
 setAcVCoef  
     Cntl.h, 35  
 setDcHvOvpLevel  
     Ovp.h, 73  
 setDcMidOcpLevel  
     Ocp.h, 59  
 setDcOtpLevel  
     Otp.h, 68  
 setExtOtpLevel  
     Otp.h, 68  
 setLoadICoef  
     Cntl.h, 35  
 setLoadOcpLevel  
     Ocp.h, 60  
 setLoadOtpLevel  
     Otp.h, 68  
 setLoadSlewStep  
     SlewControl.h, 92  
 setLoadSlewTarget  
     SlewControl.h, 92  
 setLoadState  
     SlewControl.h, 92  
 setSineGainStep  
     SineGen.h, 86  
 setSineGainTarget  
     SineGen.h, 86  
 setSineRmsTarget  
     SineGen.h, 86  
 setSineState  
     SineGen.h, 87  
 Settings.h, 78

- AC\_I\_SCALE, 80
- AC\_IRMSLV\_MAX, 80
- AC\_PWRLVL\_FIX, 80
- AC\_V\_R1, 80
- AC\_V\_R2, 80
- AC\_VRMSLV\_MAX, 80
- CHANNEL\_OOB, 80
- DCHV\_UVLCK\_LMT, 80
- DCMID\_VDCLVL\_FIX, 80
- DCMIDV\_STRTUP\_LMT, 80
- DEBUG, 80
- DUAL\_CNTL\_AC, 80
- HV\_V\_R1, 80
- HV\_V\_R2, 81
- I2C\_BUS\_BUSY, 81
- I2C\_INVALID\_ISRC, 81
- I2C\_READ\_WRONG\_MSG, 81
- I2C\_STP\_NOT\_READY, 81
- I2C\_WRITE\_WRONG\_MSG, 81
- INCR\_BUILD, 81
- LOAD\_I\_SCALE, 81
- LOAD\_IDCLVL\_MAX, 81
- LOAD\_PWRLVL\_FIX, 81
- LOAD\_V\_R1, 81
- LOAD\_V\_R2, 81
- LOAD\_VDCLVL\_FIX, 82
- LOG\_SIN, 82
- MID\_I\_SCALE, 82
- MID\_V\_R1, 82
- MID\_V\_R2, 82
- OCP\_TRIP, 82
- OPP\_TRIP, 82
- OTP\_TRIP, 82
- OVP\_TRIP, 82
- RECP\_SQRT\_2, 82
- SQRT\_2, 82
- TRIP\_ALERT, 82
- UVLKT\_TRIP, 83
- VALUE\_OOB, 83
- VDDA, 83
- VSSA, 83
- XFMR\_PWMF\_FIX, 83
- setupNets
  - MacroNets.h, 53
- SineGen.h, 83
  - disableSinePhaseOut, 85
  - enableSinePhaseOut, 85
  - getSineGainStep, 85
  - getSineGainTarget, 85
  - getSineState, 86
  - initSine, 86
  - SGENTI\_1ch\_VOut, 87
  - SIN\_CHANNEL, 84
  - SIN\_DFLT\_GAIN, 84
  - SIN\_DFLT\_RCTFY, 84
  - SIN\_F\_ISR, 84
  - SIN\_F\_ISR\_DVSR, 84
  - SIN\_F\_MAX, 85
  - SIN\_F\_REQ, 85
  - SIN\_OFST, 85
  - SIN\_PHSE, 85
  - setSineGainStep, 86
  - setSineGainTarget, 86
  - setSineRmsTarget, 86
  - setSineState, 87
  - updateSineGain, 87
  - updateSineSignal, 87
- slaveAddress
  - i2cMsg, 23
- slaveMode
  - slaveMode.h, 89
- slaveMode.h, 87
  - detectSlaveMode, 89
  - slaveMode, 89
  - slaveModeStatus, 89
- slaveModeStatus
  - slaveMode.h, 89
- slavePtrAddrHigh
  - i2cMsg, 23
- slavePtrAddrLow
  - i2cMsg, 23
- SlewControl.h, 89
  - getLoadSlewStep, 90
  - getLoadSlewTarget, 90
  - getLoadState, 90
  - scSetLoadStateAll, 90
  - setLoadSlewStep, 92
  - setLoadSlewTarget, 92
  - setLoadState, 92
  - updateLoadSlew, 92
- slewRate
  - loadStageSettings, 25
- slewTarget
  - loadStageSettings, 25
- Spi.h, 93
  - SPI\_DFLT\_BAUD, 94
  - SPI\_DFLT\_CPHA, 94
  - SPI\_DFLT\_CPOL, 94
  - SPI\_FFRX\_INTLV, 94
  - SPI\_FFTX\_FILLVL, 94
  - SPI\_FFTX\_INTLV, 94
  - spiCPha, 94, 95
  - spiInit, 95
  - spiLpbk, 94, 95
  - spiMode, 94, 95
  - spiTx, 95
  - transPol, 95
- spiCPha
  - Spi.h, 94, 95
- spiInit
  - Spi.h, 95
- spiLpbk
  - Spi.h, 94, 95
- spiMode
  - Spi.h, 94, 95
- spiTx

- Spi.h, 95
- StateMachine.h, 96
  - Alpha\_State\_Ptr, 96
  - initStateMachine, 96
- stopAll
  - MacroNets.h, 54
- stopAllFlag
  - MacroNets.h, 54
- TMP\_E\_T\_COLD
  - Tmp.h, 99
- TMP\_SCL\_OFST
  - Tmp.h, 99
- TMP\_V0C\_OFST
  - Tmp.h, 99
- TRIP\_ALERT
  - Settings.h, 82
- Timers.h, 97
  - timersSetupReal, 97
- timersSetupReal
  - Timers.h, 97
- Tmp.h, 98
  - ADC\_I2C\_ADDR, 99
  - ADC\_I2C\_BASE\_ADDR, 99
  - ADC\_NUM\_CHNL, 99
  - ADC\_STPS, 99
  - ADC\_SW3A\_STATE, 99
  - ADC\_SW3B\_STATE, 99
  - ADC\_VREF, 99
  - initTemperature, 100
  - readTemperature, 100
  - TMP\_E\_T\_COLD, 99
  - TMP\_SCL\_OFST, 99
  - TMP\_V0C\_OFST, 99
- transPol
  - Spi.h, 95
- tripAcOcp
  - Ocp.h, 60
- tripDcMidOvp
  - Ovp.h, 73
- UVLKT\_TRIP
  - Settings.h, 83
- updateLoadSlew
  - SlewControl.h, 92
- updateSineGain
  - SineGen.h, 87
- updateSineSignal
  - SineGen.h, 87
- VALUE\_OOB
  - Settings.h, 83
- VDDA
  - Settings.h, 83
- vFdbkNet
  - acStageNets, 19
  - loadStageNets, 24
- vGainLmt
  - acStageSettings, 21
- vMax
  - loadStageSettings, 25
- vMaxRms
  - acStageSettings, 21
- vRefNet
  - acStageNets, 19
- VSSA
  - Settings.h, 83
- vScale
  - acStageSettings, 21
  - loadStageSettings, 25
- XFMR\_PWMF\_FIX
  - Settings.h, 83
- XFMR\_STAGE
  - MacroNets.h, 52
- xfmrNets
  - MacroNets.h, 54
- xfmrSettings
  - MacroNets.h, 54
- xfmrStageNets, 25
  - hvVSnsNet, 26
  - iSnsNet, 26
  - MacroNets.h, 52
  - midVSnsNet, 26
  - pwmDutyNet, 26
- xfmrStageSettings, 26
  - enable, 27
  - hvOvpLevel, 27
  - hvVMax, 27
  - hvVScale, 27
  - iMax, 27
  - iScale, 27
  - MacroNets.h, 53
  - midVMax, 27
  - midVScale, 27
  - ocpLevel, 27
  - otpLevel, 27