ProductionLine TESTERS

SMU-410 DLL Functions

SMU-410 General Functions

SMU410_Open

Purpose:
This "opens" the board. This function is necessary for the application to obtain a "handle" from the Windows operating system. A "handle" is a pointer to a block of memory that has been assigned by Windows. Each SMU board will have its' own unique handle.

Prototype:
```
int SMU410_Open(int Slot);
```

Parameters:

Slot	int	This is the physical slot number in the PXI chassis

Returns:
An integer that equals the values which was sent (i.e. "slot")

Example:
```
nSlot = SMU410_Open(3);        // Opens the board in slot 3; nSlot should equal
                               3 if successful, 0 otherwise
```

SMU410_Close

Purpose:
This is a function that will release the Windows handle assignment, thereby freeing-up the block of memory that was initially assigned.

Prototype:
```
int SMU410_Close(int Slot);
```

Parameters:

Slot	int	Value which was returned from SMU410_Open

Returns:
0 if successful, -1otherwise

Example:
```
nSlot = SMU410_Close(3);       // Frees up the handle that was assigned to the
                               PXI board in Slot 3
```

SMU410_Reset

Purpose:
This function will reset the specified channel to its' default states.

Prototype:
int SMU410_Reset(int Slot, int ChannelNumber);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:
0 if successful, -1otherwise

Example:
nStatus = SMU410_Reset(3,2); // Resets defaults for Slot 3, Channel 2

SMU410_Init

Purpose:
This function will initialize the specified channel. This function calls SMU410_Reset to set defaults, loads the calibration constants and enables the DAC's. Note that it does NOT connect the SMU relay to the DB25 connector.

Prototype:
int SMU410_Init (int Slot, int ChannelNumber);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:
0 if successful, -1otherwise

Example:
nStatus = SMU410_Init (3,2); // Initializes Slot 3, Channel 2

SMU-410 VI-Source Functions

SMU410_Connect

Purpose:
Closes (connects) the relay that is in series between the VI-Source and the front panel DB25 connector.

Prototype:
int SMU410_Connect (int Slot, int ChannelNumber);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Default:
All channels are initially DISCONNECTED

Returns:
0 if successful, -1otherwise

Example:

```
nStatus = SMU410_Connect (3, 2);      // Closes relay, connecting the output of
                                      Slot 3, Channel 2 to the front panel
                                      connector
```

SMU410_Disconnect

Purpose:
Opens (disconnects) the relay that is in series between the VI-Source and the front panel DB25 connector.

Prototype:
int SMU410_Disconnect (int Slot, int ChannelNumber);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Default:
All channels are initially DISCONNECTED

Returns:
0 if successful, -1otherwise

Example:

```
nStatus = SMU410_Disconnect (3, 1);    // Closes relay, and connects output of
                                       Slot 3, Channel 1 to the front panel
                                       connector
```

SMU410_Enable

Purpose:

If Disabled, this function holds the voltage and current DAC's at 0, after the DAC's have already been programmed. When Enabled, the DAC outputs move up or down to their programmed value. This is useful when synchronizing multiple channels and also eliminates the spiking due to relay switching and floating outputs. (Note: this is NOT the same as opening and closing the "Connect" relay.)

Prototype:

```
int SMU410_Enable (int Slot, int ChannelNumber, int Enable);
```

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4
Enable	int	If Enable=0, voltage & current outputs are zero If Enable=1, voltage and current outputs are whatever has been programmed

Examples:

```
nStatus = SMU410_Enable(3, 2, 1);    // Enables Slot 3, channel 2
nStatus = SMU410_Enable(3, 2, 0);    // Disables Slot 3, channel 2
```

SMU410_Force_V

Purpose:

This programs the voltage DAC for each channel.

Prototype:

```
int SMU410_Force_V(int Slot, int ChannelNumber, double Voltage);
```

Parameters:

Slot	int	Slot returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4
Voltage	double	Any value between -10V and +10V

Returns:

0 if successful, -1 if Voltage is outside +/-10V range

Default:

All voltages on all channels are set to zero volts

Example:
```
nStatus = SMU410_Force_V(3, 2, 7.777);    // Programs Slot 3, channel 2
                                              to 7.777V
```

SMU410_Force_I

Purpose:
This is the default autoranging version of setting the current. It first calculates the current range and adjusts the value to scale for the appropriate range resistor. It then sets the current clamp polarity switch, closes the range relay, and programs the current DAC.

Prototype:
```
int SMU410_Force_I(int Slot, int ChannelNumber, double Current);
```

Parameters:

Slot	int	Slot number of opened board
ChannelNumber	int	internal channel number 0, 1, 2, or 3
Current	double	Any value between -250mA and +250mA

Returns:
-1 if current is outside +/-250mA range
0 for range = 250nA
1 for range = 2.5uA
2 for range = 25uA
3 for range = 250uA
4 for range = 2.5mA
5 for range = 25mA
6 for range = 250mA

Default:
All currents on all channels are set to 1uA (positive)

Example:
```
nStatus = SMU410_Force_I(3, 2, 3.456e-6);      // Programs channel 2 current to
                                                  3.456 microamps
```

SMU410_Set_Current_and_Range

Purpose:
This is the manual (non-autoranging) version of setting the current. The user specifies both the current and the range. (Use SMU410_Force_I for the autoranging version). It sets the current clamp polarity switch, closes the range relay, and programs the current DAC.

Prototype:
int SMU410_Set_Current_and_Range(int Slot, int ChannelNumber, double Current, double Range);

Parameters:

Slot	int	Slot number of opened board
ChannelNumber	int	internal channel number 0,1,2, or 3
Current	double	Any value between -250mA and +250mA
Range	double	Any value between -250mA and +250mA

(NOTE: In absolute-value terms, Current must be less than Range.)

Returns:
16-bit data word that was programmed into the Current DAC

Default:
All currents on all channels are set to 1uA (positive)

Example:
```
nStatus = SMU410_Current_and_Range(3, 2, 3e-6, 1e-3);
// Programs channel 2 current to 3.0 microamps, using the 1mA current range
```

SMU-410 Measurement Functions

SMU410_Measure_V

Purpose:
This function measures the voltage for each channel. The actual measurement point is the output of the Kelvin buffer. This function includes the "Start Conversion" command, a delay for the ADC conversion time, and the reading of the 24-bit results and the calculation to map this to the +/-10V range.

Prototype:
```
double SMU410_Measure_V (int Slot, int ChannelNumber);
```

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:
The voltage value at the Kelvin buffer; Range is +/-10V

Example:
```
voltage = SMU410_Measure_V (3, 2);          // Measures the voltage on Slot 3,
                                            channel 2
```

SMU410_Measure_I

Purpose:
This function measures the current for each channel. The actual measurement point is the output of the differential amplifier across the range resistor. This function includes the "Start Conversion" command, a delay for the ADC conversion time, and the reading of the 24-bit results, and the calculation which includes the current range, using the calibrated value of the range resistor.

Prototype:
```
double SMU410_Measure_I (int Slot, int ChannelNumber);
```

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:
The calculated current value, based on the range resistor and the voltage value of the output of the differential amplifier

Example:
```
current = SMU410_Measure_I (3, 2);      // Measures current on Slot 3, channel 2
```

SMU410_Set_ADC_ConversionTime_V

Purpose:
This function programs the internal registers in the analog-to-digital converter (ADC) in the voltage path, to give the user a way to trade off conversion time with resolution. For example, if the precision of the measurement is important, one may program the ADC to a higher resolution, at the expense of having to wait a longer time to get the results. On the other hand, if the exact value is not required and a faster measurement is necessary (for example in opens/shorts testing), the ADC may be programmed at a lower resolution.

Prototype:
int SMU410_Set_ADC_ConversionTime_V (int Slot, int ChannelNumber, int conversionTime);

Parameters:

Slot	int	Slot number returned from SMU_Open.
ChannelNumber	int	Internal channel number 1, 2, 3 or 4.
conversionTime	int	Number of microseconds

This is a chart of a sampling of the Conversion Time versus Effective Resolution:

Conversion Time (usec)	Effective Resolution (bits)
2686	21.0
999 (1msec)	20.3
499	19.7
395	19.5
207	19.0
166	18.7
82	17.3

Note that other tradeoffs exist. For example, the lsb for 21 bits in a +/-10V range is less than 10uV (below the noise floor in most systems), and the serial communication between chips writing and reading the ADC registers is on the order of hundreds of microseconds.

Returns:
0 if successful, -1otherwise

Default:
1000 usec (1msec, about 20.3 bits)

Example:
```
nStatus = SMU410_Set_ADC_ConversionTime_V (3, 4, 200);
// Programs the ADC for voltage measurement on Slot 3, channel 4 to 200 microseconds, which is about 19 bits of effective resolution
```

SMU410_Set_ADC_ConversionTime_I

Purpose:
This function programs the internal registers in the analog-to-digital converter (ADC) in the current path, to give the user a way to trade off conversion time with resolution. For example, if the precision of the measurement is important, one may program the ADC to a higher resolution, at the expense of having to wait a longer time to get the results. On the other hand, if the exact value is not required and a faster measurement is necessary (for example in opens/shorts testing), the ADC may be programmed at a lower resolution.

Prototype:
int SMU410_Set_ADC_ConversionTime_I (int Slot, int ChannelNumber, int conversionTime);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4
conversionTime	int	Number of microseconds

This is a chart of a sampling of the Conversion Time versus Effective Resolution:

Conversion Time (usec)	Effective Resolution (bits)
2686	21.0
999 (1msec)	20.3
499	19.7
395	19.5
207	19.0
166	18.7
82	17.3

Note that other tradeoffs exist. For example, the lsb for 21 bits in a +/-10V range is less than 10uV (below the noise floor in most systems), and the serial communication between chips writing and reading the ADC registers is on the order of hundreds of microseconds.

Returns:
0 if successful, -1otherwise

Default:
1000 usec (1msec, about 20.3 bits)

Example:
```
nStatus = SMU410_Set_ADC_ConversionTime_I (3, 4, 200);
// Programs the ADC for current measurement on Slot 3, channel 4 to 200 microseconds, which is about 19 bits of effective resolution
```

SMU410_Get_Average_V

Purpose:
This function calls "SMU410_Measure_V" repeactedly, for the programmed number of times. It reports the average reading of the multiple measurements.

Prototype:
double SMU410_Get_Average_V (int Slot, int ChannelNumber, int count);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4
count	int	Number of readings to average, up to 32767

Returns:
The average voltage value, i.e. the sum of the individual voltage readings divided by the count; Range is +/-10V

Default:
0.0V

Example:
voltage = SMU410_Get_Average_V (3, 4, 100);
// Measures the average voltage on Slot 3, channel 4, using 100 readings

SMU410_Get_Average_I

Purpose:
This function calls "SMU410_Measure_I" repeactedly, for the programmed number of times. It reports the average reading of the multiple measurements.

Prototype:
double SMU410_Get_Average_I (int Slot, int ChannelNumber, int count);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4
count	int	Number of readings to average, up to 32767

Returns:
The average current value, i.e. the sum of the individual current readings divided by the count. Range depends on which range resistor has been set. Maximum reading is +/-200mA.

Default:
0.0 amps

Example:
current = SMU410_Get_Average_I (3, 4, 100);
// Measures the average current on Slot 3, channel 4, using 100 readings

SMU410_Start_Voltage_Conversion

Purpose:
This function starts the ADC measurement cycle to read the voltage for each channel. It does NOT include any delay time.

Prototype:
int SMU410_Start_Voltage_Conversion (int Slot, int ChannelNumber);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:
Non-zero if successful, zero if failed

Example:
nStatus = SMU410_Start_Voltage_Conversion (3, 2);
// Starts the ADC conversion for Slot 3, channel 2

SMU410_Start_Current_Conversion

Purpose:
This function starts the ADC measurement cycle to read the current for each channel. It does NOT include any delay time.

Prototype:
int SMU410_Start_Current_Conversion (int Slot, int ChannelNumber);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:
Non-zero if successful, zero if failed

Example:
nStatus = SMU410_Start_Current_Conversion (3, 2);
// Starts the ADC conversion for Slot 3, channel 2

(The trigger functions below are PRELIMINARY, for information only. Please contact the factory for the release date.)

SMU-410 Trigger Functions

SMU410_Enable_Triggers

Purpose:

Selects and enables the trigger, and programs the trigger type. There are 3 registers to be programmed. For each, bits are mapped [0..7] = [Trig0..Trig7].
1.) Sync: 0=asynchronous trigger. 1=synchronous trigger.
Asynchronous means the trigger is activated according to the digital level, high or low. Synchronous means it is edge-triggered, rising or falling, with reference to the PXI-CLK10 signal (the 10MHz bus clock).
2.) Inv: 0=non-inverted trigger. 1=inverted trigger.
Triggers on the PXI bus are normally high. This means is non-inverted trigger is normally high, and is triggered when low, or on the falling edge of CLK10, depending on the sync bit. An inverted trigger is normally low, async active high and sync active on rising edge of CLK10.
3.) Enable: 0=disabled. 1=enabled.
Masking (=0) a trigger line with this bit will cause it to be ignored. Enabling a trigger line (=1) will cause the action programmed in the "SMU410_Set_Trigger_Action" function to occur when the trigger is activated.

Prototype:

int SMU410_Enable_Triggers (int Slot, int Sync, int Inv, int Enable);

Parameters:

Slot	int	Slot number returned from SMU_Open
Sync	int	0=asynchronous trigger 1=synchronous trigger
Inv	int	0=non-inverted trigger 1=inverted trigger
Enable	int	0=disabled 1=enabled

Default:

Sync=0x00	(Trigger lines all asynchronous)
Inv=0x00	(Trigger lines all non-inverted)
Enable=0x00	(Trigger lines all disabled)

Returns:

0 if successful, -1otherwise

Example:

```
nStatus = SMU410_Enable_Triggers (3, 1, 0x20, 0x20, 0x38);
// Enables Trig3, 4, and 5. Only Trig5 is synchronous and inverted (i.e. Trig5 will cause an action on the rising edge of the CLK10 pulse)
```

SMU410_Set_Trigger_Action

Purpose:
When the armed trigger fires, the firmware can do several different actions, on each of the 4 channels. This function specifies what to do when a particular trigger event occurs. Upon each activation of a trigger, Channel "Chn" will do one "Action".

Prototype:
int SMU410_Set_Trigger_Action (int Slot, int ChannelNumber, int Trigger, int Action);

Parameters:

Slot	int	Slot numberreturned from SMU_Open
Chn	int	Internal channel number 1, 2, 3 or 4
Trigger	int	Trig[0-7]=bit0-bit7
Action	int	0 = No action 1 = disable VI Source output 2 = enable VI Source output 4 = begin ADC conversion for current 8 = begin ADC conversion for voltage

Default:
Action = 0x0000 (No Trigger actions on any trigger)

Returns:
0 if successful, -1otherwise

Example:
SMU410_Set_Trigger_Action (3, 2, 0x0041, 0x0002);
// Upon a trigger event on either Trig6 or Trig0, the VI-Source on Slot 3, Channel 2 will be enabled

//---------- do not use Read_Triggers ----------//

SMU410_Read_Triggers

Purpose:
This function reads back the status of the trigger lines. It does not perform any actions (from "SMU410_Set_Trigger_Actions") and it does not change any programmed registers (from "SMU410_Enable_Triggers"). It is therefore more of a status type function for information about the trigger lines.

Prototype:
int SMU410_Read_Triggers(int Slot)

Parameters:

Slot	int	Slot number returned from SMU_Open

Default:

0 Nothing is read

Returns:

Trigger line levels Bits [0..7] correspond to Trig[0] .. Trig[7]

Example:

```
nTriggerStatus = SMU410_Read_Triggers (3);
// Trigger lines are read from the card in Slot 3
//-----------------------------------------------------------------------------------------------------//
```

SMU-410 Calibration Functions

SMU410_Load_Resistor_Cal_Const

Purpose:
This function loads the calibration constants that are saved in the EEPROM registers in each of the channels. These are loaded into arrays that are used when forcing and measuring currents.

Prototype:
int SMU410_Load_Resistor_Cal_Const (int Slot, int ChannelNumber);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:
0 if successful, -1 otherwise

Default:
Nominal values are pre-loaded. The 10ohms and 10M range resistors are 1%, and the others are 0.1%. The accuracy specifications for the VI-Sources are only valid if this function is called upon initialization.

Example:
nStatus = SMU410_Load_Resistor_Cal_Const (3, 1);
// Loads the factory-calibrated constants into an array for each of the range resistors in Slot 3, Channel 1

SMU410_Get_Cal_Const

Purpose:
This function retrieves the existing calibration constant located in EEPROM on the circuit board.

Prototype:
int SMU410_Get_Cal_Const (int Slot, int ChannelNumber, int Cal_Const_Index);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4
Cal_Const_Index	int	Index of the desired constant. See Table 1.

Returns:
8-bit calibration constant

Example:

```
nCalConst = SMU410_Get_Cal_Const (3, 2, 27);        // Returns constant 27
                                                    from Slot 3 Channel 2
```

SMU410_Set_Cal_Const

Purpose:

This function sends a calibration constant from the user interface to the microcontroller, and sets the variable within the micorcontroller code. It does NOT write the constant to the FLASH EEPROM.

Prototype:

```
int SMU410_Set_Cal_Const (int Slot, int ChannelNumber, int Cal_Const_Index,
int Cal_Const);
```

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4
Cal_Const_Index	int	Index of the desired constant. See Table 1.
Cal_Const	int	The desired constant.

Returns:

zero

Example:

```
nCalConst = SMU410_Set_Cal_Const (3, 2, 27, 15);
// Sets the calibration constant, Slot 3 Channel 2, constant number 27, to 15
```

SMU410_Save_All_Cal_Const

Purpose:

After writing the constants to the microcontroller with the Set_Cal_Const function above, this Save_All function writes the calibration constants into the FLASH EEPROM. The new constants are written over the old ones. The remaining existing constants stay the same.

Prototype:

```
int SMU410_Save_All_Cal_Const (int Slot, int ChannelNumber);
```

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:

zero

Example:

```
nCalConst = SMU410_Save_All_Cal_Const (3, 2, 27);
// Writes the constants into the FLASH on the microcontroller
```

Cal_Const_Index	Function
0	Voltage DAC zero offset adjustment
1	Voltage DAC full scale gain adjustment
2	Current DAC zero offset adjustment
3	Current DAC full scale gain adjustment
4	(Not used)
5	(Not used)
6	(Not used)
7	(Not used)
8	10M current range resistor delta
9	1M current range resistor delta
10	100K current range resistor delta
11	10K current range resistor delta
12	1K current range resistor delta
13	100 ohms current range resistor delta
14	10 ohms current range resistor delta
15	(Not used)
16	I-ADC zero offset most significant byte
17	I-ADC zero offset middle byte
18	I-ADC zero offset least significant byte
19	V-ADC zero offset most significant byte
20	V-ADC zero offset middle byte
21	V-ADC zero offset least significant byte
22	I-ADC full scale gain adjustment most significant byte
23	I-ADC full scale gain adjustment middle byte
24	I-ADC full scale gain adjustment least significant byte
25	V-ADC full scale gain adjustment most significant byte
26	V-ADC full scale gain adjustment middle byte
27	V-ADC full scale gain adjustment least significant byte

Table 1 --- Calibration Constant Index

SMU410_Clear_Cal_Block

Purpose:
This function erases all the calibration constants for the addressed channel. It will clear the constants from the microcontrollers FLASH EEPROM, and replace them with 0xFF. This also includes a 100msec delay while the microcontroller is busy. Use caution with this function.

Prototype:
int SMU410_Clear_Cal_Block (int Slot, int ChannelNumber);

Parameters:

Slot	int	Slot number returned from SMU_Open
ChannelNumber	int	Internal channel number 1, 2, 3 or 4

Returns:
non-zero internal flag

Example:

```
nFlag = SMU410_Clear_Cal_Block (3, 2);      // Erases all calibration constants
                                            in Slot 3 Channel 2
```

SMU-410 Utility Functions

SMU410_Read_PXI_Slot

Purpose:
This function reads the PXI physical slot. The PXI bus has 5 lines that are tied high and low to indicate the address of the physical slot on the PXI bus. This slot number is the same as the one printed on the chassis directly below the card. Note that this is NOT the "logical" slot that is sometimes referred to in the software.

Prototype:
int SMU410_Read_PXI_Slot (int Slot);

Parameters:

Slot	int	Slot number returned from SMU_Open

Returns: The physical slot number of the card.

Example:
```
nSlot = SMU410_Read_PXI_Slot(3);
// Reads the physical slot number of the card that has been previously open using the SMU410_Open function
```

SMU410_Get_Vendor_Id

Purpose:
This function reads the Vendor Number from the EEPROM used for the PLX9030 chip that interfaces to the PXI bus.

Prototype:
int SMU410_Get_Vendor_Id (int Slot);

Parameters:

Slot	int	Slot number returned from SMU_Open

Returns:
The Vendor Id number

Example:
```
nVendor = SMU410_Get_Vendor_Id (3);
// Reads the Vendor Number of the manufacturer of the card that has been previously open using SMU410_Open
```

SMU410_Get_Instrument_Id

Purpose:
This function reads the Instrument Model Number from the EEPROM.

Prototype:
int SMU410_Get_Instrument_Id (int Slot);

Parameters:

Slot	int	Slot number returned from SMU_Open.

Returns:
The Instrument Model Id number

Example:
```
nInstrument = SMU410_Get_Instrument_Id (3);
// Reads the Instrument Model Number of the card that has been opened using
SMU410_Open
```