Description

CambridgeIC's Central Tracking Unit (CTU) chips work with sensors built from PCBs to measure the position of contactless targets using resonant inductive sensing technology.

CTU chips are designed to be embedded inside electromechanical products, and communicate with a host system processor over an SPI interface.

To assist demonstration, evaluation and development, CambridgeIC provides hardware and software that works with a PC. A CTU Adapter enables a PC to communicate with CTU chips by converting between SPI and USB interfaces.

This document describes a set of LabVIEW Virtual Instruments (VIs) that communicate with CambridgeIC CTU chips using a PC connected to a CTU Adapter. These VIs can be used to create custom applications that communicate with CTU chips.

Applications

- Customised demonstrations
- Customised test systems

Features

- Simple example VI for taking measurements
- Example VI for taking averaged measurements
- Example VI for setting position triggers
- Full set of sub-VIs for advanced communications
- Approximately 100 SPI transfers per second

System Requirements

- Windows XP, Vista or Windows 7
- National Instruments' LabVIEW software (V 8.5 on)

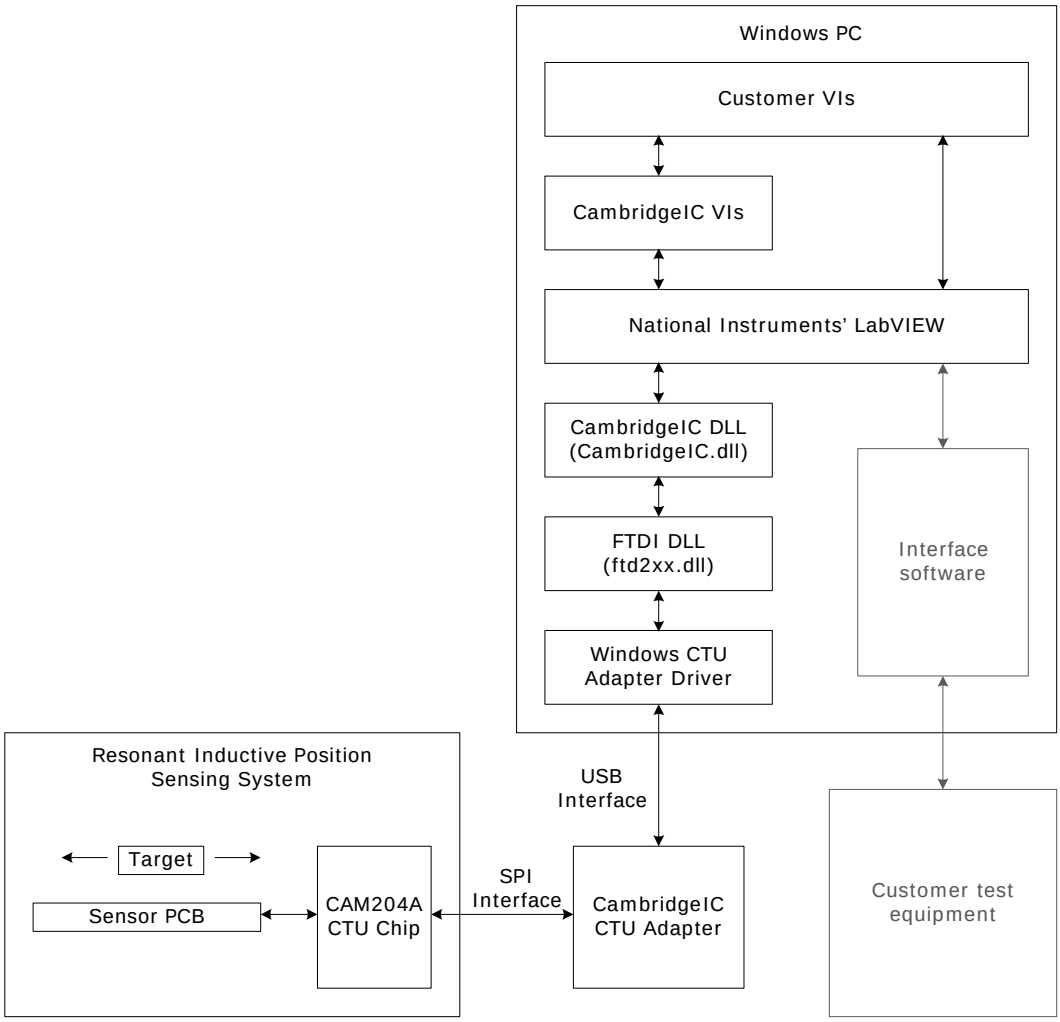

Figure 1 System block diagram

1 Introduction

1.1 Audience

This document assumes the reader is familiar with programming in National Instruments' LabVIEW.

1.2 Software Design

The software is designed to run in the environment illustrated in Figure 1. The PC is connected to a CambridgeIC CTU chip with a CTU Adapter using a USB port.

The Virtual Instruments (VIs) described in this document are coded using National Instruments' LabVIEW graphical programming language. They require LabVIEW to be installed on the target PC. LabVIEW can alternatively be used to build stand-alone applications, providing Application Builder is installed.

The VIs use two DLLs for communicating with the CTU Adapter. The higher level one is ***CambridgeIC.DLL***, and this contains all of the functions required to send data to and from a CTU chip over its SPI interface, together with functions for controlling the Adapter itself. The Adapter contains a chip for USB communication from FTDI. This has its own DLL, ***ftd2xx.DLL***, which must also be loaded on the PC to enable CambridgeIC.DLL to communicate with the Adapter. They are also included in the installer.

The Adapter requires a Windows driver, whose installation is described in the CambridgeIC CTU Software User Guide.

1.3 Communication with CambridgeIC.DLL

LabVIEW communicates with ***CambridgeIC.DLL*** using .NET controls, which can be found in the connectivity tools palette. There are sub-VIs which communicate with the DLL in this way (e.g. ***CtuPowerControl***), and it is not normally necessary to access such low-level functions directly.

All VIs that communicate through the Adapter require a valid **Adapter reference** input, which is generated by ***AdapterOpen*** or ***QuickStart***. They also output this reference, and **Adapter reference out** should be wired to the next VI that communicates through the Adapter. It is good practice to close an adapter reference before a main VI stops using ***AdapterClose***. However this is not essential.

CambridgeIC.DLL requires ***ftdxx.DLL*** for correct operation. Both may be installed alongside the CambridgeIC CTU Software using the installer described in the CambridgeIC CTU Software User Guide. They may alternatively be saved to the user's LabVIEW working directory.

1.4 CTU Datasheet

This document makes references to CTU register locations, which are used to configure CTU chips and read back data. The function of each of these registers is detailed in the relevant CTU chip datasheet.

1.5 Document Conventions

The names of VIs and DLLs are highlighted like this: ***CTU Demo***, or to the name as it appears in the VI file name such as ***CtuDemo***. The suffix *_2x* is used in the full VI file name denotes version 2.x of the software. This, and the final *.vi* in the file name, is omitted for clarity.

The names of window controls and indicators are shown like this: **Next**.

1.6 Organisation of this Document

The following main sections illustrate how to perform common tasks using the VIs supplied by CambridgeIC for communicating with the CTU. Designs become progressively more complex, and later examples assume familiarity with earlier ones.

Section 6 includes a complete listing of the VIs provided together with a brief description. Section 7 is a troubleshooting guide in case the system does not work as expected.

1.7 Modifying CambridgeIC VIs

The example VIs below are intended to be modified to form the basis of a customer's own application. It is strongly recommended that the CambridgeIC VIs are not themselves modified, however. It is preferred to "save as" … "open an additional copy" within LabVIEW to preserve the original examples. In the more unlikely event that a sub-VI requires modification, it is also recommended to modify a separate copy and not the original.

2 Checking Communications with the Adapter

AdapterTest is a simple application that can be used to check communications with the CTU Adapter. The front panel is shown in Figure 2. When run, it configures the CTU Adapter then turns on and off power and pull-ups under control of two buttons. Their state should be reflected in the state of the LEDs on the Adapter's physical front panel.

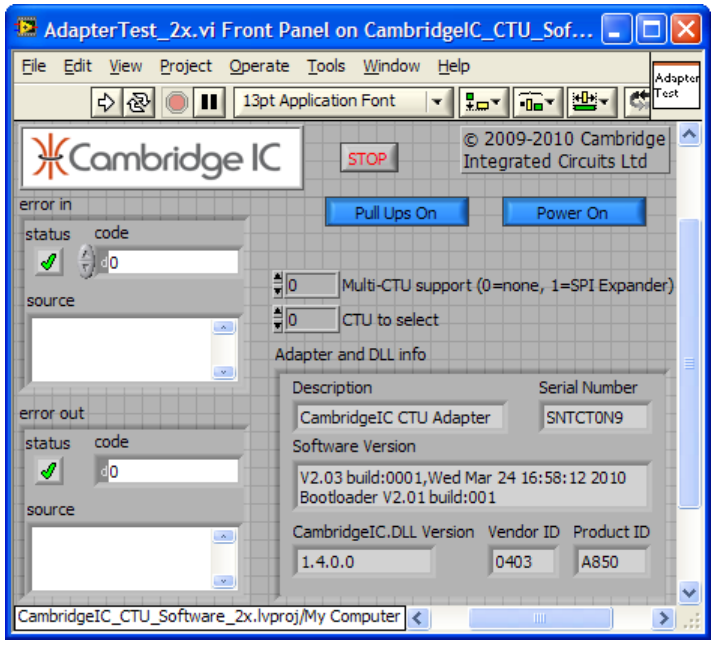

Figure 2 Adapter Test front panel

AdapterTest also includes controls for an I2C controlled SPI Expander, which may be used to select between CTU chips on the same SPI bus in multi-CTU systems. **Multi-CTU support** should be set to 1 if an SPI Expander is connected or 0 otherwise. Use the **CTU to Select** control to switch the SPI Expander between different CTU chips. Although this VI does not actually communicate with CTU chips, LEDs on the SPI Expander should light to indicate which is selected.

The block diagram is illustrated in Figure 3. Communications is first opened using ***AdapterOpen***, then the Adapter is configured using ***AdapterConfigure***. Adapter and VI information is retrieved using ***AdapterGetInfo***. ***CtuPowerConrtrol*** and ***CtuPullUpControl*** are run in a loop, and controlled with appropriately named buttons. ***SelectSpiSlave*** operates the SPI Expander, if present.

The STOP control exits this loop, whereupon ***AdapterClose*** is run to close communications with the Adapter.

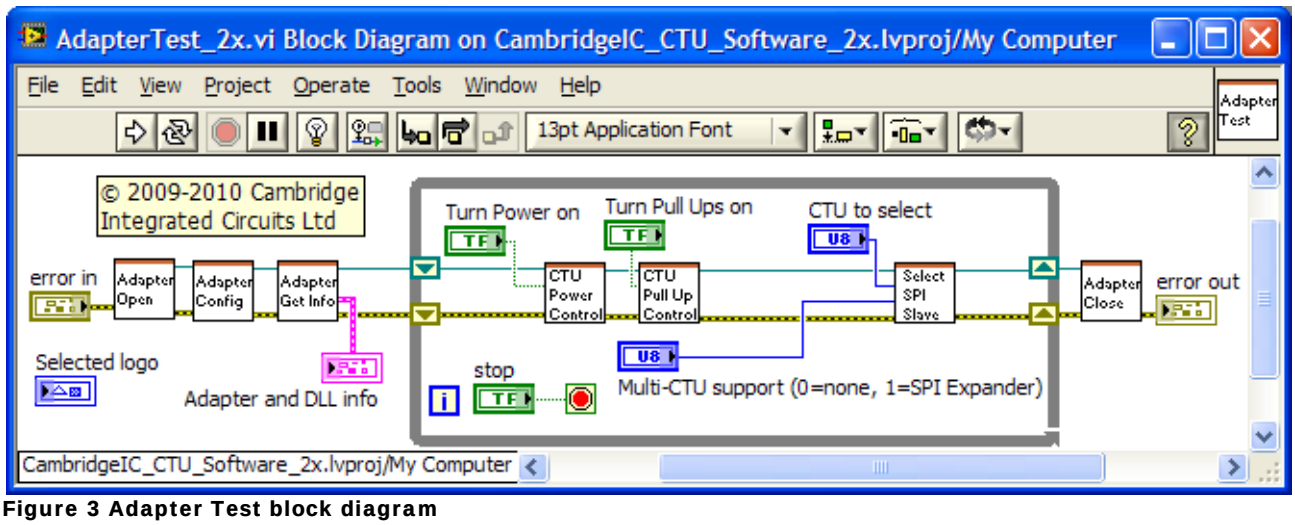

Figure 3 Adapter Test block diagram

3 Taking Single Measurements

This section describes how to use the VIs provided to take position measurements with a CTU chip. It is based on the example application ***CtuSimpleMeasureType1***. This VI can form the basis of simple demonstration and test applications.

When assessing CTU system performance, it is usually best to use a VI that takes and processes multiple samples as described in the next main section. This provides more insight into CTU performance since it measures the noise present in output data. It also yields a higher resolution position output through averaging.

Figure 4 illustrates the front panel of ***CtuSimpleMeasureType1***. Figure 5 shows the VI's block diagram. ***QuickStart*** is used to configure and test the Adapter and CTU chip. The constant **Sensor Type** is wired to ***QuickStart*** and is set for type 1 sensors. ***QuickStart*** has an output **Adapter and DLL info** which is wired to the front panel and lists version and serial number information for the Adapter and DLL. It also outputs **CTU info**, which lists version and System ID information for the CTU chip.

The **Sensor number** control selects which sensor to measure. This is wired to the VI that actually instructs the CTU to take measurements: ***CtuWriteReadResults***. This VI also requires **SCW** register contents as an input, supplied as a cluster constant on the block diagram. **GO** is set to TRUE to kick off another measurement. **CONT** is set to FALSE so that the CTU operates in single shot mode. That way, each time the PC collects one set of measurements the CTU will start measuring again. **NEW** is set to FALSE so that the sensor's NEW flag is reset. The state of the other controls within SCW do not matter for this application.

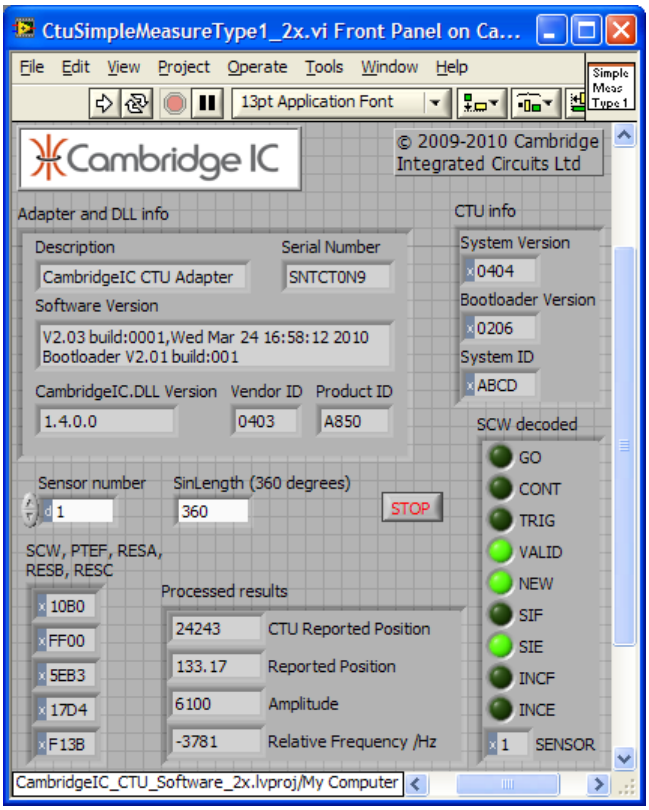

Figure 4 CTU Simple Measure Type 1 front panel

CtuWriteReadResults outputs the previous state of the following registers: **SCW, PTEF, RESA, RESB, RESC**, and these values are wired to an indicator on the front panel. LabVIEW's ***Index Array*** function is used to pick out SCW, which is passed to ***CtuInterpretScw***. This VI decodes SCW as measured back from the CTU chip and the results are displayed on the front panel.

Figure 4 shows the expected state of **SCW decoded** when ***CtuSimpleMeasureType1*** is run with a target present: **GO** FALSE (cleared by CTU following successful measurement), **NEW** TRUE (new measurement results available) and **VALID** TRUE (target present). The remainder of the bits of SCW reflect the values originally wired to ***CtuWriteReadResults***.

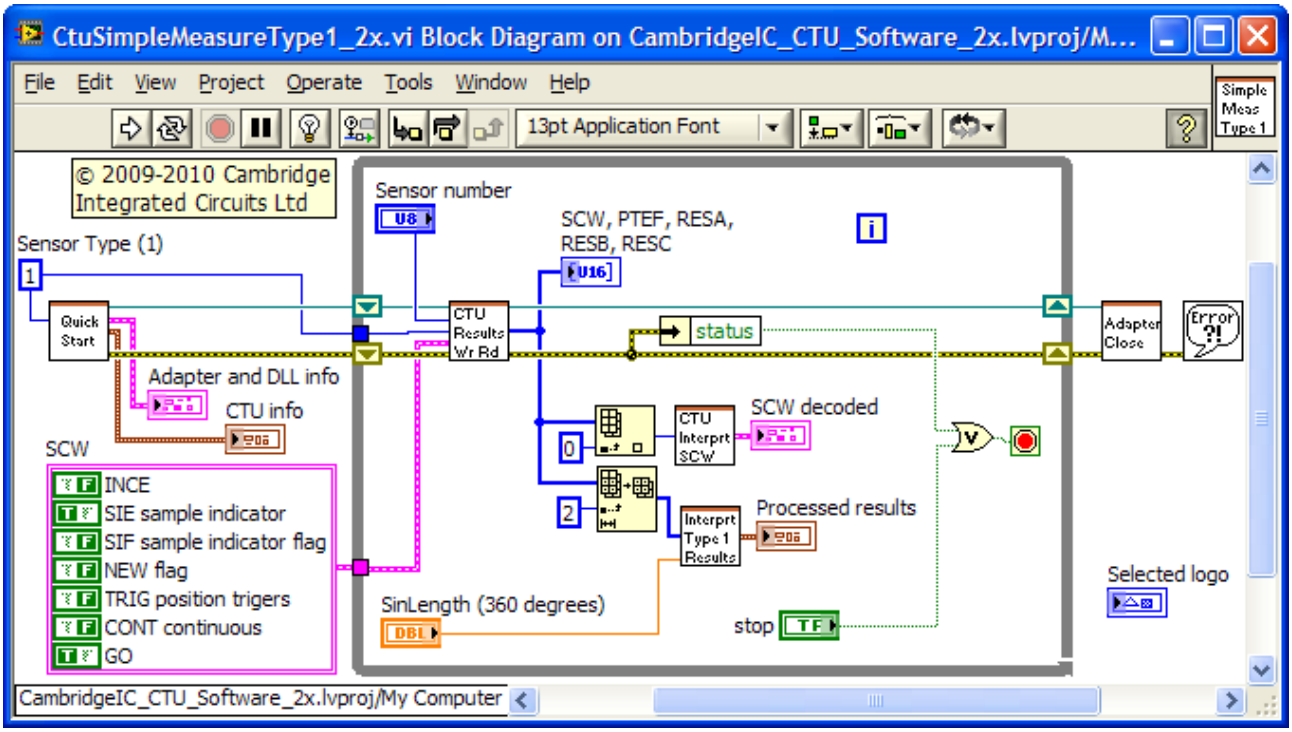

Figure 5 CTU Simple Measure Type 1 block diagram

LabVIEW's ***Array Subset*** function is used to pick out the results registers: RESA, RESB and RESC. These are wired to InterpretType1Results. This VI processes individual results and outputs them in a meaningful format as the cluster **Processed results** wired to the front panel.

CTU Reported Position is the raw CTU position output (RESA interpreted as a signed 16-bit word). This is scaled into physical units using the value of **Sin Length**. **Sin Length** is quoted in a sensor's datasheet. Set **Sin Length** to 360 for reported position in degrees when operating a 360° rotary type 1 sensor. For type 1 linear sensors, **Sin Length** is usually slightly greater than a sensor's Measuring Length. Setting the **Sin Length** control to 100 yields reported position as a percentage of the sensor's actual Sin Length.

CtuWriteReadResults is run in a while loop that is stopped with the **STOP** button or on an error. When stopped, ***AdapterClose*** is run to close communications with the Adapter. Then LabVIEW's built in VI ***Simple Error handler*** is called to highlight any errors that have occurred to the user.

CtuSimpleMeasureType1 calls ***CtuWriteReadResults*** continuously, without any time delay in the loop. Calls to ***CtuWriteReadResults*** can not be separated in time by any less than 1ms due to the PC's USB interface. Since each Type 1 measurement takes the CTU chip less than 1ms, measurement will always be complete (**NEW** TRUE) the next time the VI runs.

4 Taking and Processing Multiple Measurements

CtuMeasureAndAnalyse takes multiple measurements from a selected sensor and processes the results to yield a set of **Measurement statistics** including **Position Average** and **Position Std Dev** (standard deviation).

The number of measurements is set by **Requested Num VALID**. A larger number yields more averaging, so that the output **Position Average** has lower noise and higher resolution. For every factor of 4 increase in **Requested Num VALID**, resolution improves by approximately 1 bit.

This VI is recommended for measuring CTU system performance, so that systematic errors and noise can be treated separately. It is also recommended for single sensor demonstration applications that do not need the maximum update rate of 100 per second.

CtuMeasureAndAnalyse is called by ***CtuRepeatAverage***, whose front panel is illustrated in Figure 6 and the block diagram in Figure 7. The design is similar to ***CtuSimpleMeasureType1*** described in section 3, and the description below highlights only the main differences.

QuickStart is called at the start of execution to configure the CTU Adapter and CTU. This time two additional controls are wired to the VI. **Multi CTU Support** should be set to 0 if the system is configured normally, with the CTU Adapter addressing a single CTU. If an SPI Expander is attached, set **Multi CTU Support** to 1 and **CTU to Select** to the appropriate CTU number.

Sensor Type is a front panel control whose value must match the type of sensor connected to the CTU's **Sensor number** input for correct operation. **Sin Length** scales the position values reported by the CTU to physical units used for **Position Average**, **Position Std Dev** and **Position Pk Dev From Ave** and **Position results**. Its value should match the value quoted in the sensor's datasheet (the value of Fine Pitch A in the case of Type 2 sensors).

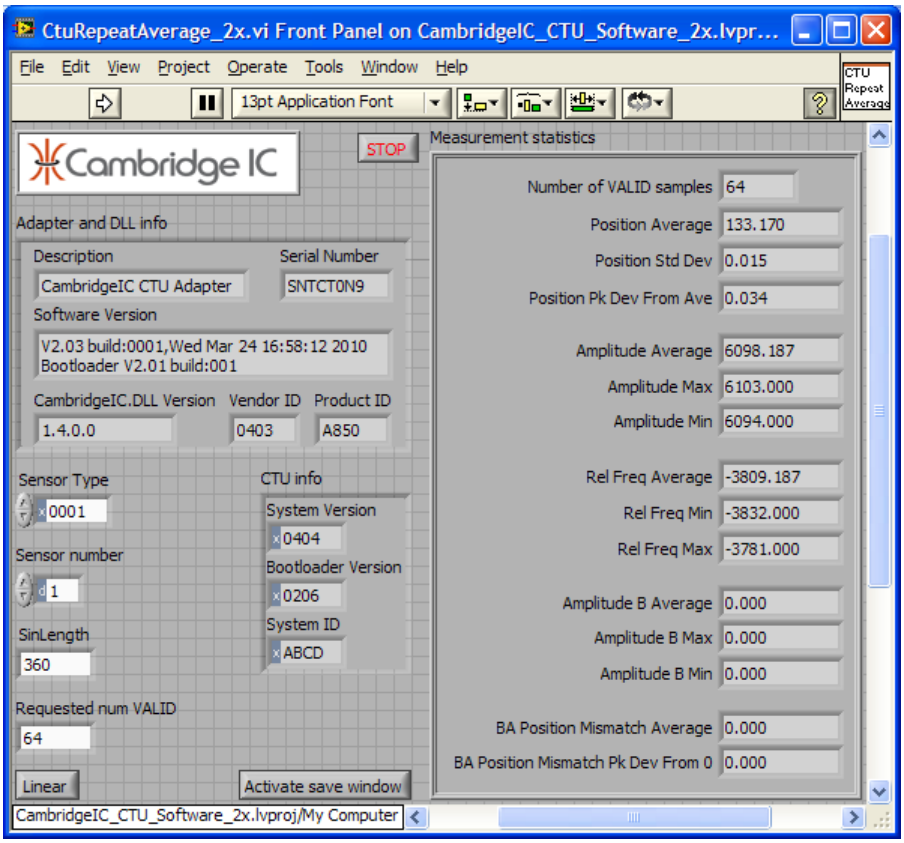

Figure 6 CTU Repeat Average front panel

The **Measurement statistics** cluster contains some indicators that are specific to certain sensor types, which are set to 0 when not applicable. For example, the CTU does not report **Amplitude B** or **BA Position Mismatch** values for a Type 1 sensor.

The **Measurement statistics** cluster includes **Number of VALID samples**. This will normally equal **Requested Num VALID** if a target is in range, otherwise 0.

CtuMeasureAndAnalyse also outputs individual processed measurements in the form of arrays, once point for each CTU measurement: **Position results**, **Amplitude results** and **Relative frequency results**. The indicators are hidden on the block diagram of Figure 6, but visible on the block diagram of Figure 7. Note that the time interval between individual measurements can be highly variable since it is based on PC timings.

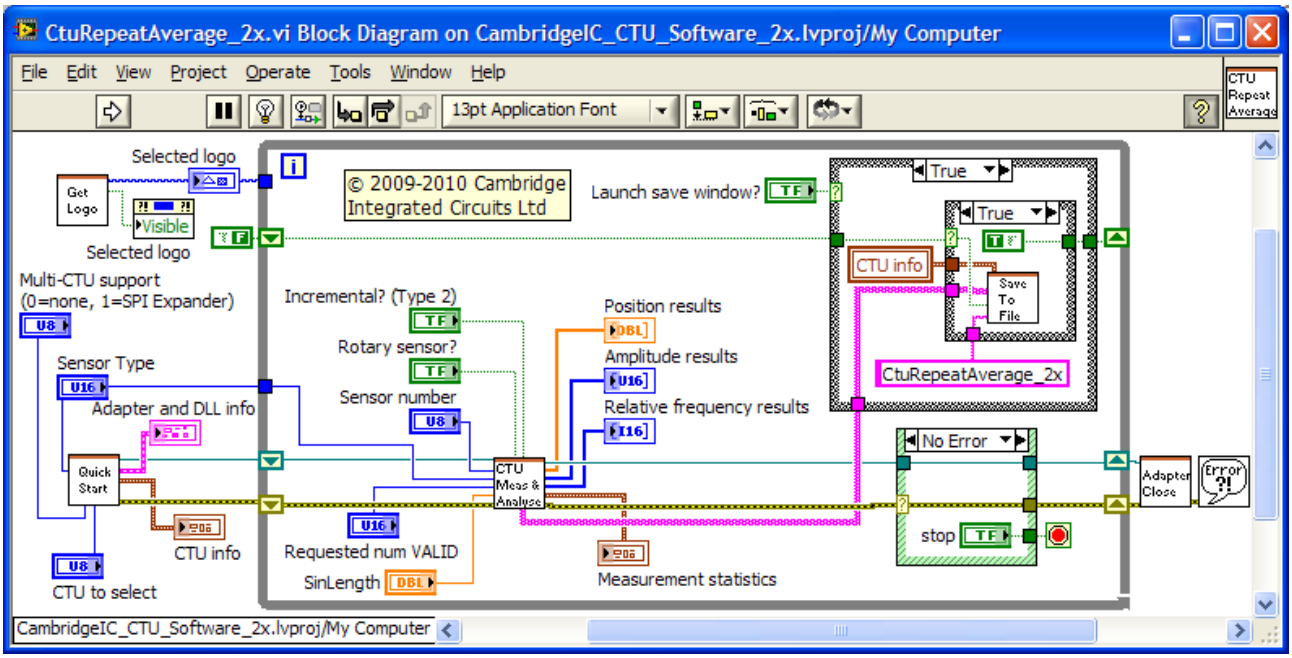

Figure 7 CTU Repeat Average block diagram

VIs which are distributed as CambridgeIC CTU Software applications, including CTU Repeat Average, use the sub-VI ***GetLogo*** to retrieve an image of the CambridgeIC logo from the file logo.bmp. Other VIs display a fixed, default logo.

5 Reading Multiple Sensors

CtuDemo can read measurement results from multiple sensors connected to the CAM204A CTU chip, providing the **Sensor Type** allows it. Its operation is detailed in the CambridgeIC CTU Software User Guide.

Unlike previous examples, ***CtuDemo*** can operate in the CTU's continuous mode (**CONT** = TRUE) as well as single shot mode (**CONT** = FALSE). When operating in continuous mode, the VI uses the CTU's sample indicators to signal when all active sensors have completed measurement. Using sample indicators requires the CTU's SIC register to be configured. The **SIC contents** cluster constant determines settings. **SMAP** maps each sensor's sample indicator to the user IO pin of the same number. **SCTRL** = 1 means sample indicators are triggered on NEW measurements. **SAUTOCLR** = 0 means the CTU does not clear sample indicators itself. Instead, they are cleared by writing FALSE to the sample indicator flag **SIF** within each sensor's **SCW** register. These values are written to each sensor's SIC registers using ***CtuSetMultiSensorSampleIndicators***.

Using continuous mode also requires a write to the CTU's **SYSI** register to change the interval between samples on each sensor timed by the CTU. ***CtuSetSystemInterval*** is called from inside a case structure which is only TRUE when the value of **SYSI** changes.

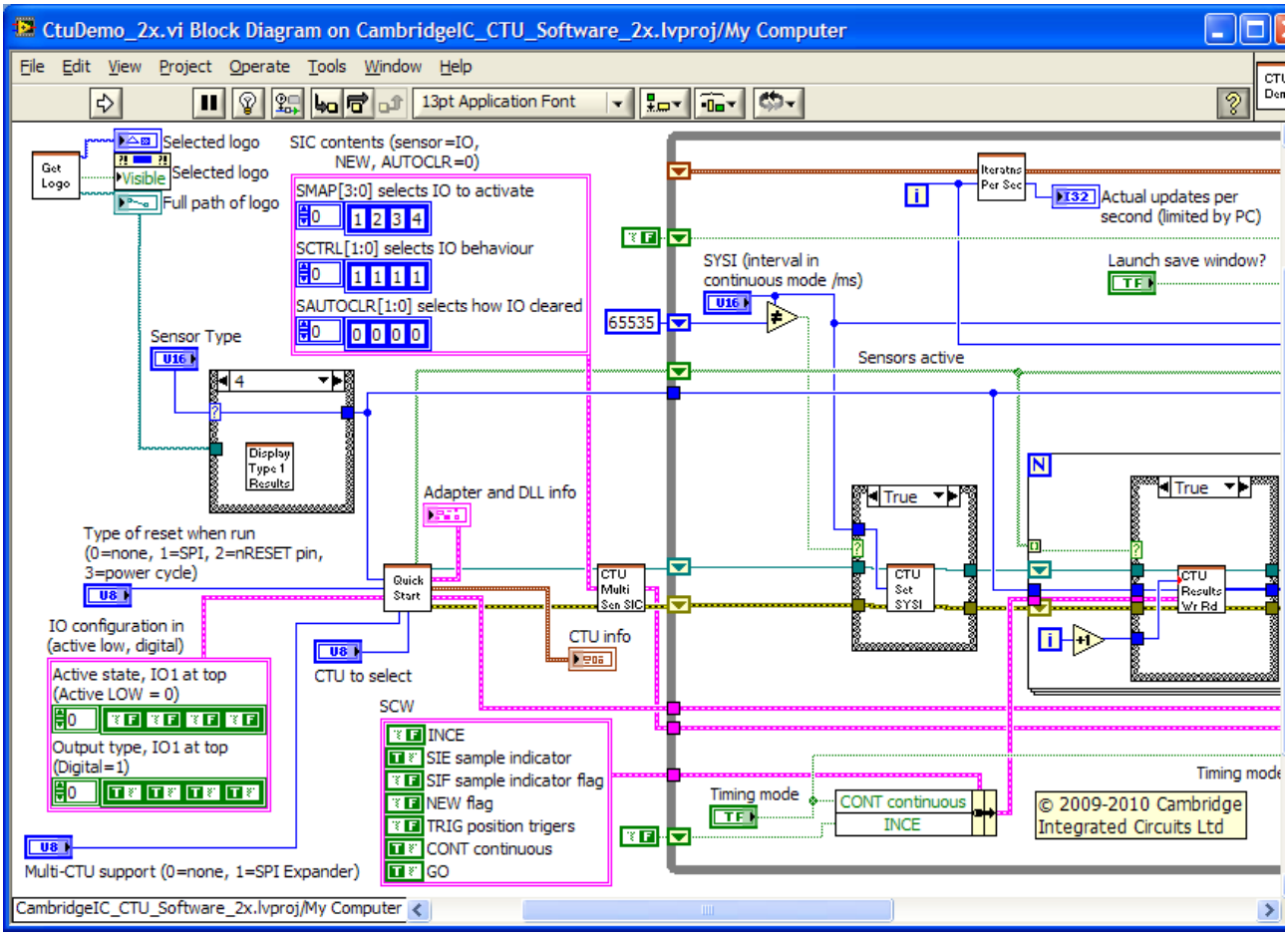

Figure 8 CTU Demo block diagram (left hand end)

QuickStart 's outputs include Sensors active, a boolean array with one entry per sensor which is TRUE for active and FALSE for inactive. This VI determines the default states (all active), and the number of array entries (the maximum number for the **Sensor Type**).

CtuWriteReadResults is called to collect the results of each measurement from each sensor. It is called from within a for loop, once for each entry in the Sensors active array (each possible sensor). It is also inside a case loop controlled by Sensors active, so that it only runs if the selected sensor is active.

The outputs of ***CtuWriteReadResults*** are passed to either ***DisplayType1Results*** or ***DisplayType2Results*** for display, depending on **Sensor Type**. They are shown in Figure 9. These two VIs are configured to launch when CtuDemo is run (right click on the VI in the block diagram, click on **SubVI Node Setup** and select **show front panel**

when called). The results of Type 1 sensor measurement are a 2D array (5 registers x 4 sensors) while the results of Type 2 sensor measurement are a 1D array (8 registers x 1 sensor).

DisplayType1Results includes a button array allowing the user to control which sensors are active, and it passes this array back to ***CtuDemo*** to update the state of the Sensors active array accordingly. Both display VIs also pass back the state of an alternative **STOP** button located on their front panels.

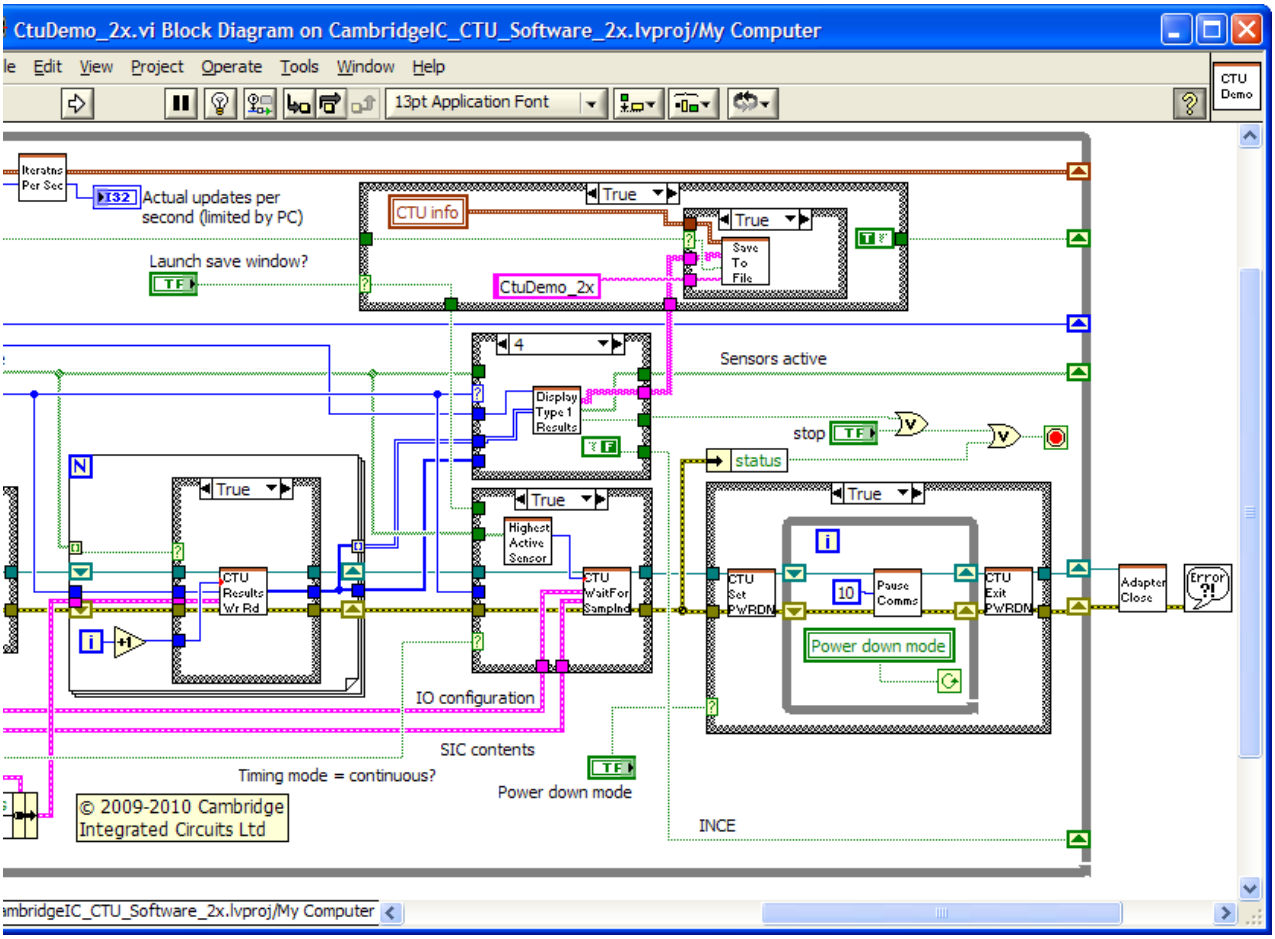

Figure 9 CTU Demo block diagram (right hand end)

If the system is running in continuous mode, ***CtuDemo*** now waits for the sample indicator of the highest numbered active sensor to become active, signaling that new measurements are ready and the main while loop can execute again to collect this next set of measurements. ***HighestActiveSensor*** takes the Sensors active array and determines which is the highest numbered active sensor. It then passes this value to ***CtuWaitForSampleIndicator***, which runs until that sample indicator becomes active.

CtuDemo includes a front panel button which appears as **Power down mode** in Figure 9 and allows the user to control the CTU's PWRDN bit in its SYSCW register. This causes the CTU to cease measurements in progress and enter a low power mode. ***CtuSetPwrdn*** is used to enter this mode, and ***CtuExitPwrdn*** is used to exit.

6 VI Descriptions

The following table includes a list of VIs in alphabetical order, and a description of each one. For readability, the highlighting conventions of section 1.5 have not been applied.

File name (minus _2x.vi)	Description
AdapterClose	It is good practice to include this VI at the end of an application to close communications with the Adapter.
AdapterConfigure	Configures the adapter. Must be run after AdapterOpen and before any other VIs. Included in QuickStart.
AdapterGetInfo	Retrieves Adapter and DLL version information. Included in QuickStart.
AdapterInternalReset	Resets the Adapter itself. Not normally required. AdapterConfigure must be run afterwards for correct operation.
AdapterOpen	Opens communications with Adapter and creates an Adapter Reference (see section 1.3). Included in QuickStart.
AdapterReadCharacters	Communicates with the Adapter at a low level. Not normally required.
AdapterTest	Small application for testing communication with the Adapter on its own without a CTU chip attached. See section 2.
AdapterWriteCharacters	Communicates with the Adapter at a low level. Not normally required.
AdapterWriteReadCharacters	Communicates with the Adapter at a low level. Not normally required.
AnalyseMultiCtuResults	Processes a set of raw, valid CTU measurements and outputs processed data series plus statistics. Called by CtuMultiMeasureAndAnalyse.
BuildScwArray	Used to create an array of SCW register controls, one for each sensor, for passing to CtuWriteReadResults.
CalculatePositionStats	As below, but calls UnwrapPhaseAngle to handle the discontinuity between ±180° if rotary.
CalculateStats	Used by AnalyseMultiCtuResults to calculate the average, standard deviation and peak deviation from average of the input array.
CtuBuildPTEF	Calculates the contents of the CTU's PTEF register from arrays containing individual enables and flags. Called by CtuWriteReadResults.
CtuBuildSCW	Calculates the contents of the CTU's SCW register from its individual contents formatted as a cluster. Called by CtuWriteReadResults.
CtuChangeSysID	Performs a register write read to the SYSID register location to change the value read out during the address word of all subsequent SPI data transactions. This VI is used to test for successful CTU reset within CtuResetAndCheck. All other VIs that communicate with the CTU check for the default value of SYSID, and will normally return an error if the value has been changed. When run, the VI returns the previous value of SYSID before the change, as reported over the SPI interface, and not the newly programmed value.
CtuCheckSysID	Checks whether its SYSID input matches the expected value. Generates errors if not, or if the CTU is waiting for firmware to be loaded.
CtuConfigureIOs	Configures the CTU's IOs for active high or low, and open drain or digital. Converts cluster of input settings to appropriate SYSINT value and writes to the CTU's SYSINT register.
CtuConfigureMultiSensor PositionTriggers	Calls CtuConfigurePositionTriggers multiple times to configure position triggers for all sensors.
CtuConfigurePositionTriggers	Configures the CTU's position trigger registers. Input parameters are processed to determine register settings, then these are written to the CTU. Called by CtuConfigureMultiSensorPositionTriggers.
CtuDemo	Main demo application documented in the CambridgeIC Software User Guide and in section 5. Displays measurement results in a separate window which depends on Sensor Type.

File name (minus _2x.vi)	Description
CtuExitPWRDN	Execute this VI to exit the CTUs' power down mode. Used in CtuDemo, see section 5.
CtuInterpretSCW	The CTU's sensor control word register contains bits for controlling the CTU's measurements on that sensor (e.g. GO) and single bit results (e.g. VALID). This VI converts an SCW value to a cluster representing each bit. Used where SCW decoding is required, e.g. CtuMultiWriteReadResults.
CtuMultiMeasureAndAnalyse	Takes a set of measurements and analyses them using AnalyseMultiCtuResults. Used by CtuRepeatAverage, see section 4.
CtuMultiWriteReadResults	Calls CtuWriteReadResults multiple times to collect a set of results for a single sensor. Discards invalid data. Called by CtuMultiMeasureAndAnalyse.
CtuNumberResults	Lists how many results to collect, which differs by sensor type. Called by CtuWriteReadResults.
CtuPowerControl	The Adapter has a power supply output to the CTU. This VI controls whether power is on or off. Called by QuickStart. Also called by AdapterTest, see section 2.
CtuPullUpControl	The Adapter has switchable pull-ups for the CTU's SPI lines. This VI controls whether they are active or not. They should be set to active for correct operation, unless pull-ups are connected by another means. QuickStart calls this VI and sets pull-ups to active. Also called by AdapterTest, see section 2.
CtuReadIOs	This VI reads the state of the CTU's user IO outputs. For correct operation, they must be set to digital, e.g. with CtuConfigureIOs, or pull-up resistors should be fitted. There are no switchable pull-ups in the Adapter; the pull-ups controlled by CtuPullUpControl are the SPI interface lines only. CtuReadIOs is called by CtuWaitForSampleIndicator to read the CTU's user IOs without performing an SPI transaction.
CtuReadRegister	This VI performs a read from the selected CTU register address. Also returns IO states.
CtuReadRegisters	This VI performs a read from multiple CTU addresses. The start address and number of addresses to read from are specified. Called by CtuReadVerAndID. Also returns IO states.
CtuReadVerAndID	Reads bootloader and system versions, and system ID. Uses CtuReadRegisters to read from the 3 adjacent CTU registers involved. Called by CtuResetAndCheck, which is called by QuickStart.
CtuRepeatAverage	Documented in the CambridgeIC CTU Software User Guide, and in section 4. Also demonstrates use of CtuMultiMeasureAndAnalyse.
CtuResetAndCheck	Performs a CTU reset. Checks that a reset has happened by checking that the SYSID register returns to its default value. Returns CTU version information. Called by QuickStart. By default, this VI uses CtuResetWithSpi to reset the CTU chip (**Type of reset** = 1). If **Type of reset** is instead set to 2, the Adapter uses CtuToggleResetPin instead, which will only work if the Adapter's nRESET pin is connected to the CTU.
CtuResetWithSpi	Performs a CTU reset over the SPI interface by writing to the SYSCW register.
CtuSetMultiSensorSample Indicators	Sets each sensor's SIC register by repeatedly calling CtuSetMultiSensorSampleIndicators. Called by CtuDemo, see section 5.
CtuSetPwrdn	Execute this VI to enter the CTUs' power down mode. Used in CtuDemo, see section 5.
CtuSetSampleIndicatorControl	Sets the CTU's SIC register of a single sensor from its individual components supplied as a cluster. Called by CtuSetMultiSensorSampleIndicators.
CtuSetSensorType	Writes to the CTU's TYPE register location for a selected sensor to configure that sensor for the specified Sensor Type. Called by CtuSetSensorTypeMultiSensor.
CtuSetSensorTypeMultiSensor	The CTU can be configured to measure multiple sensors, and that number depends on Sensor type. This VI writes the same Sensor TYPE to all possible

File name (minus _2x.vi)	Description
	sensors for that type. Also lists which sensors are possible, in the form of a Boolean array. Calls CtuSetSensorType. Called by QuickStart.
CtuSetSystemInterval	Writes to the CTU's SYSI register, which specifies the interval between measurements on a particular sensor when the sensor is operating in continuous mode. Called by CTU demo, see section 5.
CtuSimpleMeasureType1	Example VI described in section 3. Takes measurements from a selected Type 1 sensor.
CtuToggleResetPin	This VI toggles the Adapter's nRESET output pin low for a short period, which will reset the CTU if nRESET is connected to it.
CtuWaitForSampleIndicator	When called, this VI runs until a sensor's sample indicator becomes valid. It is used by CtuDemo to pause communications until the CTU reports that new measurements are available using sample indicators, see section 5. This VI requires SIC register contents for each active sensor as an input, so that it knows which IO and to test and which polarity to test for.
CtuWriteReadRegister	This VI writes new data to the selected CTU register address. It outputs the previous register contents (NOT the newly written data). It also returns the states of the CTU's user IO pins. Called by several VIs including CtuConfigureIOs.
CtuWriteReadRegisters	This VI writes data to adjacent CTU registers, starting at the specified address. It outputs the previous contents of the registers (NOT the newly written data). It also returns the states of the CTU's user IO pins. Called by CtuWriteReadResults.
CtuWriteReadResults	This VI performs a write read SPI transaction to the selected sensor's SCW, PTEF and results registers. This single transaction can be used to read the most recent results form that sensor, and to configure and kick off another measurement on the sensor. It also manipulates the flags and enables present in the SCW and PTEF registers to control the CTU's user IOs. Called by all VIs that perform measurements, for example CtuSimpleMeasureType1, see section 3.
CtuWriteReadSpi	This VI performs an SPI write read, in 16-bit word sized chunks. It is not normally required, since all CTU communication is register based and higher level VIs for register access such as CtuWriteReadRegisters are available.
DisplayType1Results	Used by CtuDemo to display the results of measurement on up to 4 Type 1 sensors, see section 5. Also used for Type 4 sensors. The Sin Length parameter sets the scaling factor from CTU units to physical units such as mm or degrees. This VI also returns the states of active/inactive buttons for each sensor which appear on its window.
DisplayType2Results	Used by CtuDemo to display the results of a measurement of a Type 2 sensor, see section 5.
DisplayType3Results	Used by CtuDemo to display the results of a measurement of a Type 3 sensor.
ErrorCodeInterpreter	This VI interprets the numeric error code returned by CambridgeIC.DLL. It is called by every VI that communicates directly with the Adapter using that DLL. It is normally used together with LabVIEW's built in VI "Error Cluster From Error Cod" to signal an error to the application. AdapterOpen is a typical example.
GetDateAndTime	Used by SaveToFile to time and date stamp measurements.
GetLogo	Retrieves logo image from file (logo.bmp).
HighestActiveSensor	This VI is used by CtuDemo to work out which sensor's sample indicator to wait for. See section 5.
InterpretType1Results	This VI is used to interpret raw results register CTU contents for a Type 1 sensor. It scales raw CTU position measurement into physical units such as mm or degrees using the Sin Length input parameter. Usually called after CtuWriteReadRegisters to interpret results. For example in CtuSimpleMeasureType1, see section 3. Also used for Type 4 sensors, which have the same results register format as Type 1.

File name (minus _2x.vi)	Description
InterpretType2Results	As above, but for a Type 2 sensor. Also works for Type 3.
IterationsPerSec	This VI can be used to determine an approximate number of iterations per second using the PC's internal clock. Called by CtuDemo to measure update rate, see section 5.
PauseComms	This VI pauses operation for the specified number of milliseconds. It is used where a delay is required, for example within CtuPowerControl to ensure that the change in power state has time to take effect before the VI finishes. It may also be used in loops to prevent LabVIEW from executing the loop so rapidly that it consumes all of the PC's CPU cycles.
PauseSingleShot	Pauses for a length of time which depends on sensor Type, to allow the next CTU measurement to complete before the PC attempts to read its results.
QuickStart	This VI performs commonly required configuration tasks for the CTU Adapter and the CTU itself. It opens and configures the Adapter, activates the power supply and SPI pull-ups, checks communications with the CTU including a reset, configures the CTU's outputs and sets sensor type(s). See sections 3, 4, 5 for examples.
SaveToFile	Saves measurements to file. Called by CtuDemo and CtuRepeatAverage when "Save to File" is activated. Takes data in the form of a text array. First column is data label, second is the data itself.
SelectSpiSlave	Selects which CTU device to select when a system has more than one on the SPI bus. Requires an I2C controlled SPI bus Expander. Used by CtuRepeatAverage among others. See section 5. Included in QuickStart.
U16toI16	This VI converts a raw unsigned 16-bit word (U16) to a signed 16-bit word (I16). It is called by InterpretType1Results to convert appropriate register contents to signed quantities.
UnwrapPhaseAngle	Takes an input array of phase angles and "wraps" it to remove artificial phase jumps around +/- 180 degrees. This is important prior to performing an average and any other function that requires an input that is continuous.

7 Troubleshooting Guide

Symptom	Possible cause	Rectification
Error 5001: DEVICE NOT FOUND	Adapter not plugged into USB port	Plug it in
	Adapter driver not installed	Install driver, please refer to the CambridgeIC CTU Software User Guide
	Adapter driver disabled	Re-Enable driver, please refer to the CambridgeIC CTU Software User Guide
	Adapter not yet enumerated	Wait 5 seconds between plugging in the Adapter and running any application that uses it
Error 5002: DEVICE NOT OPEN	A VI has been called with an **Adapter reference** input that has already been closed	Call OpenAdapter to create a new **Adapter reference**.
Error 5004: Read timeout…	Adapter power selector link missing	Check there is a link selecting between 2V7, 3V3 and 3V6 fitted through the Adapter's rear panel.
Error 5008: Adapter Error	Adapter internal error	Unplug the Adapter from the PC's USB port then plug it back in again and wait 5 seconds.
Error 5010: Exceeded max iterations waiting for active IO	Sample indicators not correctly configured	Check the sensor's SIC register contents. The AUTOCLR bit should be FALSE, and SCTRL TRUE.
	SIC and IO configuration not wired to CtuWaitForSampleIndicator	Wire the current multi-channel SIC contents and IO configuration to the VI.
	The CTU's user IOs are not connected to the CTU	Connect them, or Use single shot mode instead of continuous
Error 5012: Did not get expected SYSID from CTU	Adapter to CTU SPI interface not properly connected	Check SPI connections
	CTU chip not powered	Check power is supplied to the CTU by the Adapter or other means
Error 5013: SYSID = 0x10AD: CTU firmware missing	CTU firmware update was interrupted while in progress	Try uploading CTU firmware again
Error 5107: Failed mid-stream		Upload CTU firmware
Error 1172 occurred at Error calling method…	A VI communicating with the Adapter does not have a valid Adapter reference input	Wire the **Adapter reference** output of the previous VI that communicates with the Adapter, or to OpenAdapter.
CTU does not report VALID when a target is present	A sensor or excitation coil connection is missing or shorted	Check sensor and its connections
	A Type 2 sensor is being used with Sensor Type set to 1	Change **Sensor Type** to match the sensor type connected
	The CTU is measuring a different Type 1 sensor	Check that **Sensor number** matches the CTU input number that the sensor is connected to
	The CTU is measuring an invalid Type 2 sensor number	Check that **Sensor number** is set to 1 for a Type 2 sensor
	The CTU's excitation circuit is not working	Check the circuit conforms to the CTU datasheet
	The target is not working	Try a different target
	The target's frequency has been changed by nearby metal beyond the CTU's tuning range	Check for metal closer to target and sensor than specified in their datasheets
	Misaligned target	Check sensor datasheet for correct alignment of

Symptom	Possible cause	Rectification
		sensor and target
Reported position does not change smoothly with target position	A sensor coil connection is missing or shorted	Check sensor and its connections
	A Type 1 sensor is being used with Sensor Type set to 2	Change **Sensor Type** to match the sensor type connected
Reported position in mm or degrees incorrect	Incorrect value of Sin Length for the sensor	Check that the value used matches the appropriate value from the sensor's datasheet
Position noise different to that expected		
First few measurements with target in-range are invalid	This is normal; it can take up to 10 samples before the CTU locks on to the target's frequency	Consider using CtuMultiMeasureAndAnalyse to collect position measurements. This VI discards any initial invalid samples.
Front panel controls unresponsive when running	The VI is consuming all of the PC's CPU resources sitting in a loop	Add a time delay to the loop, e.g. with PauseComms.
		Stop the VI, change the control and run it again.
Other errors	A VI has an unwired input and the default value is not appropriate	Wire correct values to the VI's input
	Unknown LabVIEW error	Try closing and re-starting LabVIEW.
	Unknown Adapter error	Try unplugging the Adapter from the USB port and plugging it back in again.

8 Document History

Revision	Date	Description
0001	17 August 2009	First draft
0002	29 January 2010	Updated logo and style
0003	22 July 2010	Changed to reflect CTU Software version 2.x

9 Contact Information

Cambridge Integrated Circuits Ltd
21 Sedley Taylor Road
Cambridge
CB2 8PW
UK

Tel: +44 (0) 1223 413500

info@cambridgeic.com

10 Legal