GENÈVE, SUISSE
GENEVA, SWITZERLAND

Organisation Européenne pour la Recherche Nucléaire

European Organization for Nuclear Research

AB-PO Technical Note N°11

The FGC Analogue Measurement Calibration System

Quentin King

Miguel Cerqueira Bastos

Many CERN power converters are controlled by an embedded computer called a Function Generator/Controller (FGC). It must measure the current in the magnet circuit using DCCTs connected to ADCs. This note describes the calibration system for this analogue measurement chain.

Table of Contents

1. Introduction to the FGC magnet current analogue acquisition chain

The measurement of the current in a magnet circuit requires a chain of three devices:

1. DCCT head
2. DCCT electronics
3. ADC

Each of these devices has a nominal behaviour and an error which has been modelled and measured during the testing of the devices.

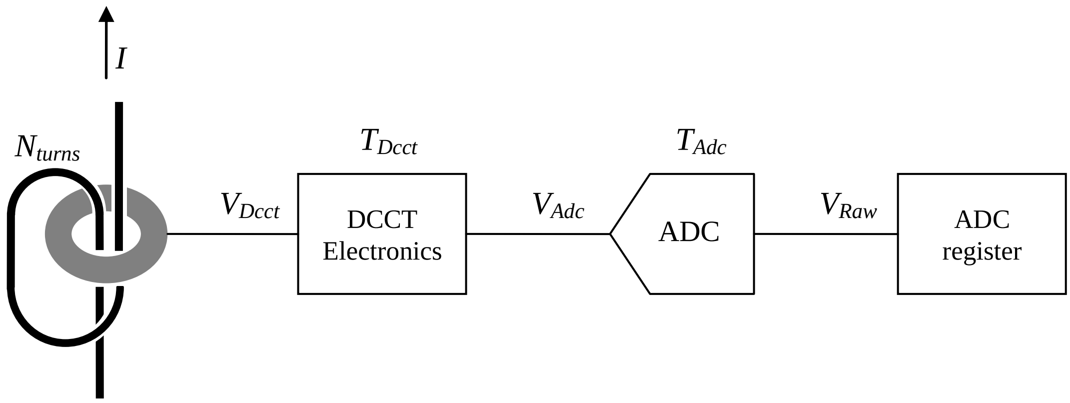

Figure 1: The analogue acquisition chain

Figure 1 shows the measurement chain and the virtual values V_{Dcct} and V_{Raw} and the real signal V_{Adc}. The virtual signals do not physically correspond to a signal at that stage in the chain, but are useful for the maths.

Only one LHC power converter has more than one primary turn through the DCCT head and that is the 60A 8V. It has two turns through the 120A DCCT that is also used by the 120A converters.

In the case of the 600A and multi-kiloamp DCCTs, a temperature sensor is included in the electronics which is filtered to derive T_{Dcct}. In addition, the internal ADCs within the FGC are near the inlet temperature sensor which is filtered to derive T_{Adc}. The 120A DCCTs do not have temperature sensors so the FGC inlet temperature is used as the source of T_{Dcct}. The filtering is first order and the time constants for T_{Dcct} and T_{Adc} can be defined in separate properties.

The high precision 22-bit external ADCs have a temperature sensor but their operating temperature is internally regulated and in the most demanding cases, they will be mounted in a temperature controlled cabinet. For this reason, temperature compensation is implemented for the DCCTs and internal ADCs but not for the external ADCs.

The nominal absolute full-scale voltage for the ADCs and DCCTs (V_{Nom}) is 10V.

Table 1: DCCT related calibration properties

Calibration property (* = A or B)	**Global config?**	**Contents**
DCCT.PRIMARY_TURNS	Yes	Number of primary turns through the DCCT, N_{turns}
DCCT.TAU_TEMP	Yes	Time constant of the first order filter used to derive T_{Dcct} from TEMP.*.DCCT.ELEC (600A or kiloamp DCCTs) or TEMP.FGC.IN (120A DCCTs).
DCCT.*.GAIN	Yes	Nominal DCCT gain G_{Dcct} in amps per volt per turn (I/V_{Dcct})
CAL.*.DCCT.HEADERR	Yes	Head gain error $E_{DcctHead}$ in ppm of $1/G_{Dcct}$
CAL.*.DCCT.ERR[6]	Yes	DCCT electronics error: $E_{DcctOffset}$, $E_{DcctGain+}$, $E_{DcctGain-}$ where $E_{DcctOffset}$ is in ppm of full scale (10V) and $E_{DcctGain+/-}$ are in ppm of nominal gain (1), all at 23°C. ERR[3,4,5] contain the time and temperature at the time of the last calibration.
CAL.*.DCCT.TC[3]	Yes	First order temperature coefficients for DCCT electronics errors $E_{DcctOffset}$, $E_{DcctGain+}$ and $E_{DcctGain-}$ in ppm/°C
CAL.*.DCCT.DTC[3]	Yes	Second order correction to the temperature coefficients for DCCT electronics errors $E_{DcctOffset}$, $E_{DcctGain+}$ and $E_{DcctGain-}$ in ppm at 28°C

1.1 FGC calibration and temperature measurement properties

Many of the calibration and temperature properties for the DCCT/ADC chain include the channel in the name: ".A." or ".B." Thus, in tables 1–3, the * character is used to refer to the channel "A" or "B".

1.2 DCCT and ADC temperature compensation

As noted in the introduction, the large DCCTs (600A or more) and the FGC include temperature sensors that can be used to compensate for some of the temperature related errors in the measurement chain. The same model has been applied in both cases and is based on a parabolic form.

For the small 120A DCCTs, which lack a Dallas temperature sensor, the FGC inlet temperature is used as the measurement for the ADCs and DCCTs.

In all cases, the temperature used in the calibration calculations is derived from the measured temperatures by using a first order filter with user definable time constant. This models the delay between changes in the air temperature and changes in the temperature of the analogue components. Initial measurements indicate that the time constants are in the order of 100s.

Table 2: ADC related calibration properties

Calibration property (* = A or B)	**Global config?**	**Contents**
CAL.*.ADC.INTERNAL.GAIN	Yes	Nominal gain G_{Adc16} for the internal ADC in raw units per 10V (V_{Nom}).
CAL.*.ADC.INTERNAL.ERR[6]	Yes	Internal ADC errors: $E_{AdcOffset}$, $E_{AdcGain+}$, $E_{AdcGain-}$ in ppm of G_{Adc16} all at 23°C. ERR[3,4,5] contain the time and temperature at the time of the last calibration
CAL.*.ADC.INTERNAL.TC[3]	Yes	First order temperature coefficients for the internal ADC errors $E_{AdcOffset}$, $E_{AdcGain+}$ and $E_{AdcGain-}$ in ppm/°C
CAL.*.ADC.INTERNAL.DTC[3]	Yes	Second order correction to the temperature coefficients for the internal ADC errors $E_{AdcOffset}$, $E_{AdcGain+}$, $E_{AdcGain-}$ in ppm at 28°C
CAL.*.ADC.INTERNAL.VRAW[7]	No	Raw ADC averages from the ADC16 automatic calibration
CAL.*.ADC.EXTERNAL.GAIN	Yes	Nominal gain G_{Adc22} for the external ADC in raw units per 10V (V_{Nom}).
CAL.*.ADC.EXTERNAL.ERR[3]	Yes	External ADC errors: $E_{AdcOffset}$, $E_{AdcGain+}$, $E_{AdcGain-}$ in ppm of G_{Adc22}
CAL.VREF.ERR[6]	Yes	Errors in ppm of ±10V for the internal ADC voltage references ([0] is not used, [1] for +10V, [2] for –10V) at 23°C. ERR[3,4,5] contain the time and temperature at the time of the last calibration.
CAL.VREF.TC[3]	Yes	Linear temperature coefficients in ppm of ±10V per °C for internal ADC voltage references ([0] is not used, [1] for +10V, [1] for –10V)
CAL.EXT_REF_ERR	No	Error of the external reference for calibration of the external 22-bit ADCs or DCCTs.
ADC.INTERNAL.TAU_TEMP	Yes	Time constant (s) of first order filter used to derive T_{Adc} from TEMP.FGC.IN.

Table 3: Temperature measurement properties (FGC2 hardware only)

Temperature property (`* = A or B`)	**Contents**
`ADC.INTERNAL.TEMPERATURE`	Internal ADC temperature T_{Adc}
`DCCT.*.TEMPERATURE`	DCCT temperatures T_{Dcct}
`TEMP.FGC.IN`	FGC inlet temperature (°C). This measurement is near the internal ADC components and is used to derive T_{Adc} and T_{Dcct} for the 120A DCCTs which lack a Dallas device.
`TEMP.FGC.OUT`	FGC outlet temperature (°C). This measurement is near the CPU board radiator. The value is not used for calibration.
`TEMP.*.DCCT.HEAD`	DCCT head temperature (°C). This measurement is not used for the calibration as the head is not significantly affected by the temperature.
`TEMP.*.DCCT.ELEC`	DCCT electronics temperature (°C). When available, this measurement is used to derive T_{Dcct}.
`TEMP.*.ADC22.MOD`	External ADC modulator temperature (°C). The 22-bit ADC includes temperature control so compensation is not required.
`TEMP.*.ADC22.PSU`	External ADC PSU temperature (°C). This sensor will provide an estimate of the environment temperature for the ADC.

The temperatures are measured at 10 second intervals. The filters run at 1 Hz and the calibration coefficients are also recalculated at 1 Hz.

It is assumed that there can be a different error as a function of temperature for the offset and positive and negative gains. In all cases the calibration errors specified in the properties `CAL.*.DCCT.ERR` and `CAL.*.ADC.INTERNAL.ERR` apply when the measured temperature is T_0, which is defined to be 23°C.

The effect of temperature changes is represented by an additional error ξ, which is zero by definition at T_0 and varies as a function of the measured temperature. This additional error has the same units as the offset or gain error that it modifies.

ξ has a linear first order component and a parabolic second order component. The linear component is defined by a simple T_c value in ppm/°C:

$$\xi_1(T) = T_c(T - T_0) \qquad [1]$$

The parabolic component is defined by a deviation from the linear component at T_1, which is defined to be 28°C. This parabolic component is zero by definition at T_0 and T_2, which is defined to be 33°C:

$$\xi_2(T) = \Delta T_c \frac{(T - T_0)\cdot(T - T_2)}{(T_1 - T_0)\cdot(T_1 - T_2)} \qquad [2]$$

Thus the combined temperature error is:

$$\xi(T) = (T - T_0)\cdot\left(T_c + \Delta T_c \frac{(T - T_2)}{(T_1 - T_0)\cdot(T_1 - T_2)}\right) \qquad [3]$$

These equations are illustrated in figures 2, 3 and 4. The same temperature error equation [3] is used to modify the offset and positive and negative gain calibration errors for the internal ADCs and large DCCTs alike. However, the way it is used is different for offset and gains. This is explained in more detail later.

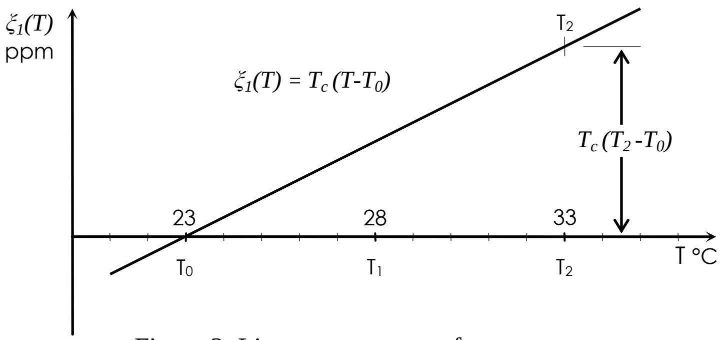

Figure 2: Linear component of temperature error

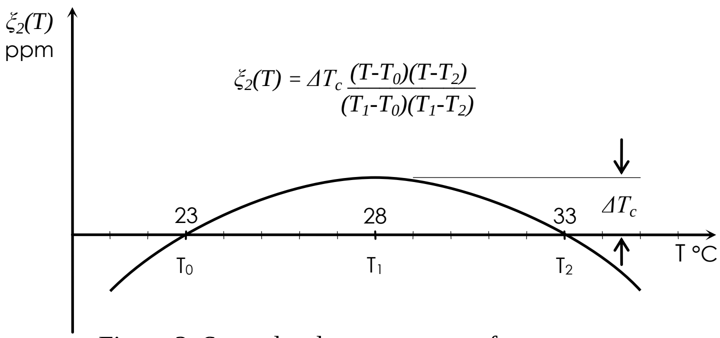

Figure 3: Second order component of temperature error

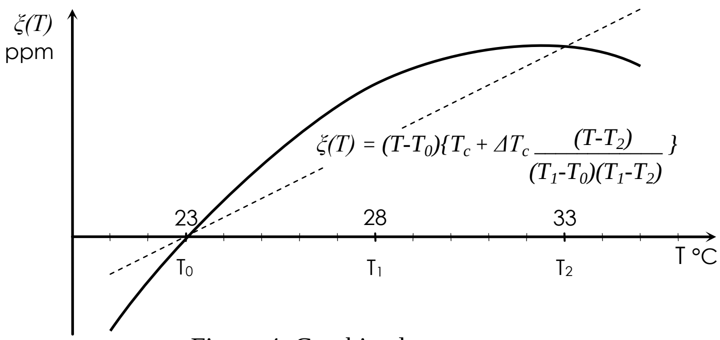

Figure 4: Combined temperature error

2. DCCT

2.1 DCCT Head

The DCCT head has a nominal gain G_{Dcct} defined in Amps/Volt/Turn such that:

$$I = \frac{G_{Dcct} \cdot V_{Dcct}}{N_{turns}} \qquad [4]$$

Or putting it in the more natural order of *output = f(input)*:

$$V_{Dcct} = \frac{N_{turns}}{G_{Dcct}} \cdot I \qquad [5]$$

The DCCT gain property `DCCT.*.GAIN` is part of the configuration database record associated with the type of DCCT being used, as identified by the DCCT barcodes.

The N_{turns} property `DCCT.PRIMARY_TURNS` is part of the configuration database record associated with the type of power converter.

2.1.1 DCCT Head Function

For the large DCCT heads, a head ratio error is measured, leading to a revised function:

$$V_{Dcct} = \frac{N_{turns} \cdot \left(1 + E_{DcctHead} \cdot 10^{-6}\right)}{G_{Dcct}} \cdot I \qquad [6]$$

Where:

N_{turns} ← `DCCT.PRIMARY_TURNS`
G_{Dcct} ← `DCCT.*.GAIN`
$E_{DcctHead}$ ← `CAL.*.DCCT.HEADERR`

The head error property `CAL.*.DCCT.HEADERR` is part of the configuration database record for the specific DCCT head being used, as identified by its Dallas ID.

2.2 DCCT Electronics

2.2.1 DCCT Offset Error

For the DCCTs, the offset error used at temperature T_{Dcct} will be:

$$E_{DcctOffset}(T_{Dcct}) = E_{DcctOffset} + \xi_{DcctOffset}\left(T_{Dcct}\right) \qquad [7]$$

Where (as described above):

$$\xi(T) = \left(T - T_0\right) \cdot \left(T_c + \Delta T_c \frac{\left(T - T_2\right)}{\left(T_1 - T_0\right) \cdot \left(T_1 - T_2\right)}\right) \qquad [3]$$

T_0 = 23°C
T_1 = 28°C
T_2 = 33°C

and:

$E_{DcctOffset}$ ← `CAL.*.DCCT.ERR[0]`
T_c ← `CAL.*.DCCT.TC[0]`
ΔT_c ← `CAL.*.DCCT.DTC[0]`

The units of $E_{DcctOffset}$ (T_{dcct}) are ppm of V_{Nom}, i.e. 10 µV if V_{Nom} is 10V.

2.2.2 DCCT Positive Gain Error

The positive gain error used at temperature T_{Dcct} will be:

$$E_{DcctGain+}(T_{Dcct}) = E_{DcctGain+} + \xi_{DcctGain+}\left(T_{Dcct}\right) \quad [8]$$

Where:

$E_{DcctGain+}$ ← `CAL.*.DCCT.ERR[1]`
T_c ← `CAL.*.DCCT.TC[1]`
ΔT_c ← `CAL.*.DCCT.DTC[1]`

The units of $E_{DcctGain+}$ (T_{Dcct}) are ppm of nominal gain (1).

2.2.3 DCCT Negative Gain Error

The negative gain error used at temperature T_{Dcct} will be:

$$E_{DcctGain-}(T_{Dcct}) = E_{DcctGain-} + \xi_{DcctGain-}\left(T_{Dcct}\right) \quad [9]$$

Where:

$E_{DcctGain-}$ ← `CAL.*.DCCT.ERR[2]`
T_c ← `CAL.*.DCCT.TC[2]`
ΔT_c ← `CAL.*.DCCT.DTC[2]`

The units of $E_{DcctGain-}$ (T_{Dcct}) are ppm of nominal gain (1).

2.2.4 DCCT Electronics Functions

Combining the offset and gain errors we can derive the functions for the DCCT electronics:

$$V_{Adc} = E_{DcctOffset}(T_{Dcct}) \cdot 10^{-6} \cdot V_{Nom} + \left(1 + E_{DcctGain+}(T_{Dcct}) \cdot 10^{-6}\right) \cdot V_{Dcct} \quad \mathit{if} \quad V_{Dcct} \geq 0 \quad [10]$$

$$V_{Adc} = E_{DcctOffset}(T_{Dcct}) \cdot 10^{-6} \cdot V_{Nom} + \left(1 + E_{DcctGain-}(T_{Dcct}) \cdot 10^{-6}\right) \cdot V_{Dcct} \quad \mathit{if} \quad V_{Dcct} < 0 \quad [11]$$

3. ADC

3.1 Nominal ADC Gain

ADC calibrations are errors are in ppm of a nominal gain factor. This makes it easier to follow changes in the ADC calibrations and to apply temperature compensation measures.

The nominal gains of the internal ADCs G_{Adc16}, will be calculated on request as part of an automatic calibration process if user issues the command:

```
S CAL ADCS,NOMINAL_GAIN
```

If the NOMINAL_GAIN option is missing then the existing values of G_{Adc16} for each channel will be used for the calibration process. This is normally the case. It is only necessary to recalculate the nominal gain if the ADC has drifted so much that the gain errors are now out of the tolerance limits (typically 300 ppm), or if the filter has been changed. When the software calculates the NOMINAL_GAIN it is set such that the positive gain error is initially zero.

The nominal gain of the external ADCs, G_{Adc22} will be measured in the lab as part of their manufacturing tests and is never calculated automatically.

3.2 ADC Offset Error

3.2.1 External ADCs

For the external 22-bit ADCs, the offset error is not dependent upon the measured temperature and is simply:

$$E_{AdcOffset} \qquad [12]$$

Where: $E_{AdcOffset}$ ← `CAL.*.ADC.EXTERNAL.ERR[0]`

The units of $E_{AdcOffset}$ are ppm of the nominal ADC gain G_{Adc22} for the specific ADC.

3.2.2 Internal ADCs

For the internal ADCs, the offset error at temperature T_{Adc} is derived using:

$$\boxed{E_{AdcOffset}(T_{Adc}) = E_{AdcOffset} + \xi_{AdcOffset}(T_{Adc})} \qquad [13]$$

Where (as described above):

$$\xi(T) = (T - T_0)\left(T_c + \Delta T_c \frac{(T - T_2)}{(T_1 - T_0)(T_1 - T_2)}\right) \qquad [3]$$

and:

$E_{AdcOffset}$ ← `CAL.*.ADC.INTERNAL.ERR[0]`
T_c ← `CAL.*.ADC.INTERNAL.TC[0]`
ΔT_c ← `CAL.*.ADC.INTERNAL.DTC[0]`

The units of $E_{AdcOffset}$ (T_{Adc}) are ppm of the nominal ADC gain G_{Adc16} for the specific ADC.

3.3 **ADC Positive Gain Error**

3.3.1 External ADCs

For the external 22-bit ADCs, the positive gain error is not dependent upon the measured temperature and is simply:

$$E_{AdcGain+} \qquad [14]$$

where: $E_{AdcGain+}$ ← `CAL.*.ADC.EXTERNAL.ERR[1]`

The units of $E_{AdcGain+}$ are ppm of the nominal ADC gain G_{Adc22} for the specific ADC.

3.3.2 Internal ADCs

For the internal ADCs, the positive gain error at temperature T_{Adc} is derived using:

$$E_{AdcGain+}(T_{Adc}) = E_{AdcGain+} + \xi_{AdcGain+}(T_{Adc}) \qquad [15]$$

where:
$E_{AdcGain+}$ ← `CAL.*.ADC.INTERNAL.ERR[1]`
T_c ← `CAL.*.ADC.INTERNAL.TC[1]`
ΔT_c ← `CAL.*.ADC.INTERNAL.DTC[1]`

The units of $E_{AdcGain+}$ (T_{Adc}) are ppm of the nominal ADC gain G_{Adc16} for the specific ADC.

3.4 **ADC Negative Gain Error**

3.4.1 External ADCs

For the external 22-bit ADCs, the negative gain error is not dependent upon the measured temperature and is simply:

$$E_{AdcGain-} \qquad [16]$$

where: $E_{AdcGain-}$ ← `CAL.*.ADC.EXTERNAL.ERR[2]`

The units of $E_{AdcGain-}$ are ppm of the nominal ADC gain G_{Adc22} for the specific ADC.

3.4.2 Internal ADCs

For the internal ADCs, the negative gain error at temperature T_{Adc} is derived using:

$$E_{AdcGain-}(T_{Adc}) = E_{AdcGain-} + \xi_{AdcGain-}(T_{Adc}) \qquad [17]$$

where:
$E_{AdcGain-}$ ← `CAL.*.ADC.INTERNAL.ERR[2]`
T_c ← `CAL.*.ADC.INTERNAL.TC[2]`
ΔT_c ← `CAL.*.ADC.INTERNAL.DTC[2]`

The units of $E_{AdcGain-}$ (T_{Adc}) are ppm of the nominal ADC gain G_{Adc16} for the specific ADC.

3.5 ADC Functions

3.5.1 External ADCs

For $V_{Adc} \geq 0$:

$$V_{Raw} = E_{AdcOffset} \cdot 10^{-6} \cdot G_{Adc22} + \left(1 + E_{AdcGain+} \cdot 10^{-6}\right) \cdot \frac{G_{Adc22}}{V_{Nom}} \cdot V_{Adc} \qquad [18]$$

For $V_{Adc} < 0$:

$$V_{Raw} = E_{AdcOffset} \cdot 10^{-6} \cdot G_{Adc22} + \left(1 + E_{AdcGain-} \cdot 10^{-6}\right) \cdot \frac{G_{Adc22}}{V_{Nom}} \cdot V_{Adc} \qquad [19]$$

where: G_{Adc22} ← `CAL.*.ADC.EXTERNAL.GAIN`

3.5.2 Internal ADCs

For $V_{Adc} \geq 0$:

$$V_{Raw} = E_{AdcOffset}(T_{Adc}) \cdot 10^{-6} \cdot G_{Adc16} + \left(1 + E_{AdcGain+}(T_{Adc}) \cdot 10^{-6}\right) \cdot \frac{G_{Adc16}}{V_{Nom}} \cdot V_{Adc} \qquad [20]$$

For $V_{Adc} < 0$:

$$V_{Raw} = E_{AdcOffset}(T_{Adc}) \cdot 10^{-6} \cdot G_{Adc16} + \left(1 + E_{AdcGain-}(T_{Adc}) \cdot 10^{-6}\right) \cdot \frac{G_{Adc16}}{V_{Nom}} \cdot V_{Adc} \qquad [21]$$

where: G_{Adc16} ← `CAL.*.ADC.INTERNAL.GAIN`

4. Using the Calibration Values

Equations 6, 10, 11 and 18–21 define the functions from:

$$I \rightarrow V_{Dcct} \rightarrow V_{Adc} \rightarrow V_{Raw}$$

In operation, the software must reverse this chain:

$$I \leftarrow V_{Dcct} \leftarrow V_{Adc} \leftarrow V_{Raw}$$

Using the approximation:

$$\frac{1}{1+\xi} \cong 1-\xi \qquad [22]$$

will result in an error of approximately ξ^2. So this is safe to use provided ξ is less than ~300 ppm (i.e. the error will be less than 0.1 ppm).

Using 22 to invert equations 18 and 19 for the external ADCs we get:

$$V_{Adc} \cong \frac{V_{Raw}}{\frac{G_{Adc22}}{V_{Nom}}} - \left\{ V_{Nom} \cdot E_{AdcOffset} + \frac{V_{Raw}}{\frac{G_{Adc22}}{V_{Nom}}} \cdot E_{AdcGain+} \right\} \cdot 10^{-6} \quad if \quad V_{Raw} \geq 0 \qquad [23]$$

$$V_{Adc} \cong \frac{V_{Raw}}{\frac{G_{Adc22}}{V_{Nom}}} - \left\{ V_{Nom} \cdot E_{AdcOffset} + \frac{V_{Raw}}{\frac{G_{Adc22}}{V_{Nom}}} \cdot E_{AdcGain-} \right\} \cdot 10^{-6} \quad if \quad V_{Raw} < 0 \qquad [24]$$

Similarly, inverting equations 20 and 21 for the internal ADCs:

$$V_{Adc} \cong \frac{V_{Raw}}{\frac{G_{Adc16}}{V_{Nom}}} - \left\{ V_{Nom} \cdot E_{AdcOffset}(T_{Adc}) + \frac{V_{Raw}}{\frac{G_{Adc16}}{V_{Nom}}} \cdot E_{AdcGain+}(T_{Adc}) \right\} \cdot 10^{-6} \quad if \quad V_{Raw} \geq 0 \quad [25]$$

$$V_{Adc} \cong \frac{V_{Raw}}{\frac{G_{Adc16}}{V_{Nom}}} - \left\{ V_{Nom} \cdot E_{AdcOffset}(T_{Adc}) + \frac{V_{Raw}}{\frac{G_{Adc16}}{V_{Nom}}} \cdot E_{AdcGain-}(T_{Adc}) \right\} \cdot 10^{-6} \quad if \quad V_{Raw} < 0 \quad [26]$$

Apply the same approximation to invert the DCCT functions described by equations 10 and 11:

$$V_{Dcct} \cong V_{Adc} - \left(V_{Nom} \cdot E_{DcctOffset}(T_{Dcct}) + V_{Adc} \cdot E_{DcctGain+}(T_{Dcct})\right) \cdot 10^{-6} \quad if \quad V_{Adc} \geq 0 \qquad [27]$$

$$V_{Dcct} \cong V_{Adc} - \left(V_{Nom} \cdot E_{DcctOffset}(T_{Dcct}) + V_{Adc} \cdot E_{DcctGain-}(T_{Dcct})\right) \cdot 10^{-6} \quad if \quad V_{Adc} < 0 \qquad [28]$$

And finally inverting equation 6:

$$I \cong \frac{V_{Dcct} \cdot G_{Dcct}\left(1 - E_{DcctHead} \cdot 10^{-6}\right)}{N_{turns}} \qquad [29]$$

5. Last Calibration Time and Temperature

Tracking the long term evolution of the calibration errors is of great importance. For this reason, all the properties containing calibration errors are six elements long, with the last three elements dedicated to recording the time and temperature at the time of the last calibration. This concerns the following properties:

- `CAL.A.DCCT.ERR[6]`
- `CAL.B.DCCT.ERR[6]`
- `CAL.A.ADC.INTERNAL.ERR[6]`
- `CAL.B.ADC.INTERNAL.ERR[6]`
- `CAL.A.ADC.INTERNAL.ERR[6]`
- `CAL.B.ADC.INTERNAL.ERR[6]`
- `CAL.VREF.ERR[6]`

In all cases the last three elements are:

- [3] = Temperature measurement (°C)
- [4] = Calibration date (days since 1 January 1970)
- [5] = Calibration time (seconds since midnight UTC)

The DCCT and ADCs can be calibrated automatically in which case these temperature and date/time stamps are made automatically. The VREF error can only be calibrated by an external PC connected to a DVM and the calibration program running in the PC is responsible for setting the temperature and date/time stamps at the same time as the VREF errors.

6. External ADC voltage references

The external 22-bit ADCs can be calibrated semi-automatically using an external portable voltage reference standard (PBC) that provides a stabilised voltage with a known small error with respect to +10V. It also provides a –10V reference by being connected with inverted polarity, however, it should be noted that all power converters that use external high precision 22-bit ADCs are unipolar in current, so the –10V calibration is not operationally significant and is only done to fully monitor the evolution of the ADC.

Before using the PBC reference to calibrate the external ADCs connected to an FGC, the PBC error must be entered into the property CAL.EXT_REF_ERR using the command:

S CAL.EXT_REF_ERR *pbc_error_in_ppm*

The PBC reference is temperature compensated and has a negligible residual temperature coefficient, thus the software does not need to apply temperature compensation.

The DIV+PAM calibrations use a stable high voltage reference that is monitored by a HP3458 DVM. The nominal value of the high voltage reference V_{HVref} may be changed and must be set in an FGC property CAL.DIVPAM.HVREF. This property is a vector with one value per DIV+PAM. If V_{HVref} is changed then the nominal gain factor for the DIV+PAM must be recalculated.

6.1 +10V PBC Reference

The voltage of the +10V PBC reference is estimated using:

$$V_{PBC+} = 10 \cdot \left(1 + E_{PBC} \cdot 10^{-6}\right) \quad [30]$$

Where: $E_{PBC} \leftarrow$ CAL.EXT_REF_ERR

6.2 –10V PBC Reference

As mentioned, the PBC is simply inverted to provide a –10V reference:

$$V_{PBC-} = -10 \cdot \left(1 + E_{PBC} \cdot 10^{-6}\right) \quad [31]$$

Where: $E_{PBC} \leftarrow$ CAL.EXT_REF_ERR

7. Internal ADC voltage references

The FGC's internal ADCs can be automatically calibrated using on-board ±10V references which may be selected using analogue multiplexers. These references have an error that is dependent upon the temperature. The references can be measured via the FGC's front panel test connector, so it is possible (though time-consuming) to measure their temperature coefficients.

When the references are used during the auto-calibration process, there is an additional error due to layout reasons, however, this can be estimated during the calibration and is compensated as explained in the next section.

7.1 +10V Internal Reference

The voltage of the +10V internal reference at temperature $T_{Adc} = T_{CalAdc}$ is estimated using:

$$V_{ref+}(T_{CalAdc}) = 10 \cdot \left(1 + E_{ref+}\left(T_{CalAdc}\right) \cdot 10^{-6}\right) \qquad [32]$$

Where the temperature dependent error in ppm of +10V is defined as:

$$E_{ref+}(T_{CalAdc}) = E_{ref+} + \xi_{ref+}\left(T_{CalAdc}\right) \qquad [33]$$

Where: $$\xi(T) = T_c\left(T - T_0\right) \qquad [34]$$

and: E_{ref+} ← `CAL.VREF.ERR[1]`
T_c ← `CAL.VREF.TC[1]`

7.2 –10V Internal Reference

Similarly, the voltage of the –10V reference is estimated using:

$$V_{ref-}(T_{CalAdc}) = -10 \cdot \left(1 + E_{ref-}\left(T_{CalAdc}\right) \cdot 10^{-6}\right) \qquad [35]$$

Where the temperature dependent error in ppm of –10V is defined as:

$$E_{ref-}(T_{CalAdc}) = E_{ref-} + \xi_{ref-}\left(T_{CalAdc}\right) \qquad [36]$$

Where: $$\xi(T) = T_c\left(T - T_0\right) \qquad [37]$$

and: E_{ref-} ← `CAL.VREF.ERR[2]`
T_c ← `CAL.VREF.TC[2]`

8. Calibrating the External 22-bit ADCs

The calibration of the external ADCs is semi-automatic. The user must physically connect the reference voltage to the input of the ADC and then request the FGC perform the calibration action. This should always start with the setting of the nominal gain, if this was not yet done. This is followed by the calibration of the offset error $E_{AdcOffset}$, since this value is needed for the calibration of the ADC gain errors $E_{AdcGain+}$ and $E_{AdcGain-}$. The nominal gain will be measured during the commissioning of the 22-bit ADCs and will not normally need to be set after that.

8.1 Deriving the external ADC offset error: $E_{AdcOffset}$

Starting from either equation 18 or 19 with the ADC input short circuited ($V_{Adc} = 0$):

$$E_{AdcOffset} = \frac{\overline{V_{Raw0}}}{G_{Adc22}} \cdot 10^6 \qquad [38]$$

Where:

	$\overline{V_{Raw0}}$	=	*Average raw ADC value with short circuit*
	G_{Adc22}	←	`CAL.*.ADC.EXTERNAL.GAIN`
	$E_{AdcOffset}$	→	`CAL.*.ADC.EXTERNAL.ERR[0]`

The FGC command to perform this offset error calibration is:

`S CAL ADC22A,CALZERO` *for channel A*
`S CAL ADC22B,CALZERO` *for channel B*

This will average the raw ADC value over a suitable period (~10s) before using equation 38 to calculate the new value of the ADC offset error.

8.2 Deriving the external ADC positive gain error: $E_{AdcGain+}$

By connecting the PBC reference, $V_{Adc} = V_{PBC+}$, and then combining equations 18 and 30 we can derive:

$$E_{AdcGain+} = \frac{\left(\overline{V_{Raw+}} - G_{Adc22}\right) \cdot 10^6}{G_{Adc22}} - E_{PBC} - E_{AdcOffset} \qquad [39]$$

Where:

	$\overline{V_{Raw+}}$	=	*Average raw ADC value with positive PBC*
	G_{Adc22}	←	`CAL.*.ADC.EXTERNAL.GAIN`
	$E_{AdcGain+}$	→	`CAL.*.ADC.EXTERNAL.ERR[1]`

The FGC command to perform this positive gain error calibration is:

`S CAL ADC22A,CALPOS` *for channel A*
`S CAL ADC22B,CALPOS` *for channel B*

This will average the raw ADC value over a suitable period (~10s) before using equation 39 to calculate the new value of the ADC positive gain error.

8.3 Deriving the external ADC negative gain error: $E_{AdcGain-}$

By connecting the PBC reference with inverted polarity, $V_{Adc} = V_{PBC-}$, and then combining equations 19 and 31 we can derive:

$$E_{AdcGain-} = \frac{\left(-\overline{V_{Raw-}} - G_{Adc22}\right)\cdot 10^{6}}{G_{Adc22}} - E_{PBC} + E_{AdcOffset} \qquad [40]$$

Where: $\overline{V_{Raw-}}$ = *Average raw ADC value with negative PBC*
G_{Adc22} ← `CAL.*.ADC.EXTERNAL.GAIN`
$E_{AdcGain-}$ → `CAL.*.ADC.EXTERNAL.ERR[2]`

The FGC command to perform this positive gain error calibration is:

`S CAL ADC22A,CALNEG` *for channel A*
`S CAL ADC22B,CALNEG` *for channel B*

This will average the raw ADC value over a suitable period (~10s) before using equation 40 to calculate the new value of the ADC negative gain error.

9. Calibrating the Internal FGC2 ADCs

The automatic calibration of the internal ADCs is made significantly more complex by the introduction of temperature compensation. The process must derive $E_{AdcOffset}$, $E_{AdcGain+}$, $E_{AdcGain-}$ in ppm of the nominal gain G_{Adc16} at temperature T_0 (defined to be 23°C) even if the calibration is done at temperature $T_{Adc} = T_{CalAdc}$.

The calibration process for the FGC2 uses the analogue multiplexers that enable one of eight different signals to be routed to the ADCs*:

Channel	Signal	Symbol
0	Aux	AUX
1	DCCT B	DCCTB
2	DCCT A	DCCTA
3	DAC	DAC
4	–10V reference	CALPOS
5	Ground	CALZERO
6	Ground	CALZERO
7	+10V reference	CALNEG

This is illustrated in figure 5 below. Channels 5 and 6 are equivalent and short circuit the differential ADC inputs to ground, enabling $E_{AdcOffset}$ to be derived. Channels 4 and 7 provide access to the internal ±10V references, but have an error due to the op-amp input bias current and the imbalanced filter network of the voltage references. An attempt is made by the software to compensate for part of these errors as described in the next section.

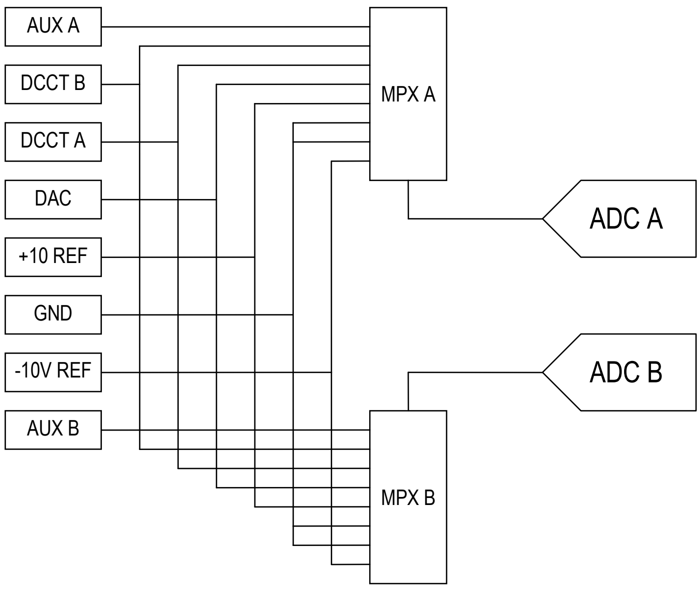

Figure 5: The internal ADC analogue multiplexers for the FGC2

9.1 The automatic calibration command for the internal ADCs

A full automatic calibration of the ADCs can be launched using the command:

```
S CAL ADCS
```

This command will only be accepted if the converter is off and the ILOOP logging has stopped (this continues for 20s after a converter switches off). Furthermore, the calibration will be launched automatically if more than 24 hours has elapsed since the last calibration, provided the property `ADC.INHIBIT_CAL` is zero and the other requirements are met (converter is off etc…). The property `ADC.INHIBIT_CAL` is a 1 Hz down counter which the user can set to any value from 0 to 1,000,000 seconds, in order to inhibit calibrations during the specified period.

If the nominal gain needs to be calculated at the same time then the command becomes:

```
S CAL ADCS,NOMINAL_GAIN
```

9.2 Voltage reference measurement error compensation

If a voltage reference is connected to an ADC via the multiplexer, an input bias current of up to 10nA will flow from the reference through an unbalanced RC filter network. This results in a voltage shift of up to –3 ppm, dependent upon the ADC card, channel and temperature.

However, if the reference is measured by one channel, and at the same time the reference is switched in and out for the other channel, it is possible to measure the effect of the bias current for the other channel and in this way, to estimate the voltage shift that the other channel will see when it measures the reference on its own. This can be done systematically for both channels by making the following series of measurements:

Measurement	Channel A	Channel B
0	CALZERO	CALZERO
1	CALPOS	CALNEG
2	CALPOS	CALPOS
3	CALNEG	CALPOS
4	CALNEG	CALNEG

For each measurement, the multiplexers will be set as specified in the table and then, after a settling time (~400ms), the raw ADC values will be averaged over a suitable period (8s). Assuming that the results are stored in the integer arrays *VrawA[7]* and *VrawB[7]*. The *Vraw* arrays have 7 elements so that the last two can be used to store the compensation differences:

$$VrawA[5] = VrawB[3] - VrawB[2] \qquad [41]$$

$$VrawA[6] = VrawB[1] - VrawB[4] \qquad [42]$$

$$VrawB[5] = VrawA[1] - VrawA[2] \quad [43]$$

$$VrawB[6] = VrawA[3] - VrawA[4] \quad [44]$$

The following calculations will provide the raw values needed for the subsequent auto-calibrations of channel A:

$$\overline{V_{Raw0}} = VrawA[0] \quad [45]$$

$$\overline{V_{Raw+}} = VrawA[1] + VrawA[5] \quad [46]$$

$$\overline{V_{Raw-}} = VrawA[3] + VrawA[6] \quad [47]$$

Similarly, the following calculations will provide the raw values needed for the auto-calibrations of channel B:

$$\overline{V_{Raw0}} = VrawB[0] \quad [48]$$

$$\overline{V_{Raw+}} = VrawB[3] + VrawB[5] \quad [49]$$

$$\overline{V_{Raw-}} = VrawB[1] + VrawB[6] \quad [50]$$

9.3 Partial automatic calibration of one internal ADC

It might be necessary to calibrate the internal ADCs without stopping operation. This is possible, though not recommended. Only one ADC channel can be calibrated at a time and the regulation must be switch to the other channel using the `MEAS.SELECT` property.

The commands to calibrate an individual error of an individual channel are for channel A:

```
S CAL ADC16A,CALZERO
S CAL ADC16A,CALPOS
S CAL ADC16A,CALNEG
```

And for channel B:

```
S CAL ADC16B,CALZERO
S CAL ADC16B,CALPOS
S CAL ADC16B,CALNEG
```

These individual operations will use the Vraw values calculated during the previous global calibration (provided there was one since the last reset).

9.4 Deriving the temperature normalised ADC offset error: $E_{AdcOffset}$

Starting from either equation 20 or 21 with $V_{Adc} = 0$ and the temperature $T_{Adc} = T_{CalAdc}$:

$$E_{AdcOffset}(T_{CalAdc}) = \frac{\overline{V_{Raw0}}}{G_{Adc16}} \cdot 10^6 \quad [51]$$

Thus, using equation 13, the temperature normalised ADC offset error can be estimated using:

$$E_{AdcOffset} = \frac{\overline{V_{Raw0}}}{G_{Adc16}} \cdot 10^6 - \xi_{AdcOffset}\left(T_{CalAdc}\right) \qquad [52]$$

Where: G_{Adc16} ← `CAL.*.ADC.INTERNAL.GAIN`
$E_{AdcOffset}$ → `CAL.*.ADC.INTERNAL.ERR[0]`

9.5 Deriving the temperature normalised ADC positive gain error: $E_{AdcGain+}$

Using equations 20 and 32 with the average raw values V_{Raw0}, V_{Raw+} *and* V_{Raw-} we can estimate the positive gain error using:

$$E_{AdcGain+}(T_{CalAdc}) = \frac{\left(\overline{V_{Raw+}} - \overline{V_{Raw0}} - G_{Adc16}\right) \cdot 10^6}{G_{Adc16}} - E_{ref+}(T_{CalAdc}) \qquad [53]$$

Thus, using equation 15, the temperature normalised ADC positive gain error will be:

$$E_{AdcGain+} = \frac{\left(\overline{V_{Raw+}} - \overline{V_{Raw0}} - G_{Adc16}\right) \cdot 10^6}{G_{Adc16}} - E_{ref+}(T_{CalAdc}) - \xi_{AdcGain+}(T_{CalAdc}) \qquad [54]$$

Where: G_{Adc16} ← `CAL.*.ADC.INTERNAL.GAIN`
$E_{AdcGain+}$ → `CAL.*.ADC.INTERNAL.ERR[1]`

9.6 Deriving the temperature normalised ADC negative gain error: $E_{AdcGain-}$

Using equations 21 and 35 with the average raw values V_{Raw0}, V_{Raw+} *and* V_{Raw-} we can estimate the positive gain error using:

$$E_{AdcGain-}(T_{CalAdc}) = \frac{\left(-\overline{V_{Raw-}} + \overline{V_{Raw0}} - G_{Adc16}\right) \cdot 10^6}{G_{Adc16}} - E_{ref-}(T_{CalAdc}) \qquad [55]$$

Thus, using equation 17, the temperature normalised ADC negative gain error is:

$$E_{AdcGain-} = \frac{\left(-\overline{V_{Raw-}} + \overline{V_{Raw0}} - G_{Adc16}\right) \cdot 10^6}{G_{Adc16}} - E_{ref-}(T_{CalAdc}) - \xi_{AdcGain-}(T_{CalAdc}) \qquad [56]$$

Where: G_{Adc16} ← `CAL.*.ADC.INTERNAL.GAIN`
$E_{AdcGain-}$ → `CAL.*.ADC.INTERNAL.ERR[2]`

9.7 Deriving the nominal ADC gain error: G_{Adc16}

The nominal ADC gain is calculated so that the temperature normalised ADC positive gain error will be zero. So setting equation 54 to zero we derive:

$$G_{Adc16} = \left(\overline{V_{Raw+}} - \overline{V_{Raw0}}\right)\left(1 - \left(E_{ref+}(T_{CalAdc}) - \xi_{AdcGain+}(T_{CalAdc})\right) \cdot 10^6\right) \qquad [57]$$

Where: G_{Adc16} → `CAL.*.ADC.INTERNAL.GAIN`

10. Calibrating the DCCTs

The calibration of the DCCTs will be made easier by a semi-automatic process included in the FGC software (similar to the calibration of the external 22-bit ADCs). The process must derive $E_{DcctOffset}$, $E_{DcctGain+}$, $E_{DcctGain-}$ at temperature T_0 (defined to be 23°C) even if the calibration is done at temperature $T_{Dcct} = T_{CalDcct}$.

The calibration process requires that:

- The ADCs are already well calibrated
- The user drives the DCCT electronics to produce a signal corresponding first to zero current, then positive full scale and finally negative full scale. If the current is not exactly zero, and plus or minus full scale, then provided the error is known, it can be entered into the property `CAL.EXT_REF_ERR` before each calibration command. This error is in ppm of full scale.
- For each current, the semi-automatic calibration process must be triggered using the commands:
    - `S CAL DCCTA,CALZERO` or `S CAL DCCTB,CALZERO` or `S CAL DCCTS,CALZERO`
    - `S CAL DCCTA,CALPOS` or `S CAL DCCTB,CALPOS` or `S CAL DCCTS,CALPOS`
    - `S CAL DCCTA,CALNEG` or `S CAL DCCTB,CALNEG` or `S CAL DCCTS,CALNEG`

10.1 Deriving the temperature normalised DCCT offset error: $E_{DcctOffset}$

Starting from either equation 10 or 11 with the DCCT measuring zero amps in the circuit ($I = 0$) and the temperature $T_{Dcct} = T_{Cal}$:

$$E_{DcctOffset}(T_{CalDcct}) = \frac{\overline{V_{Adc0}}}{V_{Nom}} \cdot 10^6 \qquad [58]$$

Where: $\overline{V_{Adc0}}$ = *Average ADC value for I=0*

Thus, using equation 7, the temperature normalised DCCT offset error is:

$$E_{DcctOffset} = \frac{\overline{V_{Adc0}}}{V_{Nom}} \cdot 10^6 - E_{ref} - \xi_{DcctOffset}\left(T_{CalDcct}\right) \qquad [59]$$

Where: E_{ref} ← `CAL.EXT_REF_ERR`
$E_{DcctOffset}$ → `CAL.*.DCCT.ERR[0]`

10.2 Deriving the temperature normalised DCCT positive gain error: $E_{DcctGain+}$

Starting from equation 6 with the DCCT measuring positive full scale $I = (V_{Nom} \cdot G_{Dcct}) / N_{turns}$:

$$V_{Dcct+} = V_{Nom} \cdot (1 + E_{DcctHead} \cdot 10^{-6}) \qquad [60]$$

Combining this with equations 10, 20 and 22 and with the DCCT temperature $T_{Dcct} = T_{CalDcct}$ and ADC temperature $T_{Adc} = T_{CalAdc}$:

$$E_{DcctGain+}(T_{CalDcct}) = \begin{bmatrix} \dfrac{\left(\overline{V_{Raw+}} - G_{Adc}\right) \cdot 10^{6}}{G_{Adc}} \\ - E_{AdcOffset}(T_{CalAdc}) - E_{AdcGain+}(T_{CalAdc}) \\ - E_{DcctHead} - E_{DcctOffset}(T_{CalDcct}) - E_{ref} \end{bmatrix} \qquad [61]$$

Where:

E_{ref} $\leftarrow$ `CAL.EXT_REF_ERR`
G_{Adc} $\leftarrow$ `CAL.*.ADC.INTERNAL/EXTERNAL.GAIN`
$\overline{V_{raw+}}$ = *Average raw ADC value for* $I = (V_{Nom} \cdot G_{Dcct}) / N_{turns}$

Thus, using equation 8, the temperature normalised DCCT positive gain error is:

$$E_{DcctGain+} = \begin{bmatrix} \dfrac{\left(\overline{V_{Raw+}} - G_{Adc}\right) \cdot 10^{6}}{G_{Adc}} \\ - E_{AdcOffset}(T_{CalAdc}) - E_{AdcGain+}(T_{CalAdc}) \\ - E_{DcctHead} - E_{DcctOffset}(T_{CalDcct}) - E_{ref} \\ - \xi_{DcctGain+}(T_{CalDcct}) \end{bmatrix} \qquad [62]$$

Where: $E_{DcctGain+}$ $\rightarrow$ `CAL.*.DCCT.ERR[1]`

Note that it is important to take the difference between $\overline{V_{raw+}}$ and G_{Adc} using integer maths and not 32-bit floating point, which has insufficient resolution.

10.3 Deriving the temperature normalised DCCT negative gain error: $E_{DcctGain-}$

Starting from equation 6 with the DCCT measuring negative full scale $I = -(V_{Nom} \cdot G_{Dcct})/N_{turns}$:

$$V_{Dcct-} = -V_{Nom} \cdot (1 + E_{DcctHead} \cdot 10^{-6}) \qquad [63]$$

Combining this with equations 11, 21 and 22 and with the DCCT temperature T_{Dcct} = $T_{CalDcct}$ and ADC temperature T_{Adc} = T_{CalAdc}:

$$E_{DcctGain-}(T_{CalDcct}) = \begin{bmatrix} \dfrac{\left(-\overline{V_{Raw-}} - G_{Adc}\right) \cdot 10^6}{G_{Adc}} \\ + E_{AdcOffset}(T_{CalAdc}) - E_{AdcGain-}(T_{CalAdc}) \\ + E_{DcctOffset}(T_{CalDcct}) - E_{DcctHead} - E_{ref} \\ \end{bmatrix} \qquad [64]$$

Where:

E_{ref} ← `CAL.EXT_REF_ERR`
G_{Adc} ← `CAL.*.ADC.INTERNAL/EXTERNAL.GAIN`
$\overline{V_{raw-}}$ = *Average raw ADC value for* $I = -(V_{Nom} \cdot G_{Dcct})/N_{turns}$

Thus, using equation 9, the temperature normalised DCCT negative gain error is:

$$E_{DcctGain-} = \begin{bmatrix} \dfrac{\left(-\overline{V_{Raw-}} - G_{Adc}\right) \cdot 10^6}{G_{Adc}} \\ + E_{AdcOffset}(T_{CalAdc}) - E_{AdcGain-}(T_{CalAdc}) \\ + E_{DcctOffset}(T_{CalDcct}) - E_{DcctHead} - E_{ref} \\ - \xi_{DcctGain-}(T_{CalDcct}) \end{bmatrix} \qquad [65]$$

Where: $E_{DcctGain-}$ → `CAL.*.DCCT.ERR[2]`

Note that it is important to take the difference between $-\overline{V_{raw-}}$ and G_{Adc} using integer maths and not 32-bit floating point, which has insufficient resolution.

11. Calibration library: libcal

There is a C library that implements the equations presented in this paper called libcal. It contains:

- 8 public functions concerned with the measurement of a current by a DCCT and an ADC
- 2 public functions concerned with temperature compensation
- 2 public functions concerned with producing the voltage reference using a DAC

The library assumes that the system has the ability to automatically measure reference voltages and the DAC output with the internal ADCs (using analogue multiplexers and/or switches) and that calibration references can be connected either manually or automatically to the DCCT heads and external ADCs.

Furthermore, if temperature compensation is required then a temperature measurement must be possible for the ADCs and/or the DCCT electronics.

11.1 Temperature compensation

If the air temperature changes then there is a lag before the ADC or DCCT calibrations change due to the thermal inertia of the circuit boards and components. A function is provided to model this using a simple first order filter. To use the function the cal_temp_filter structure must be initialised using:

```
void calTempFilterInit(struct cal_temp_filter *temp_filter, float period_s, float time_constant_s)
```

Once this is done the temperature can be filtered by calling this function every period_s seconds:

```
void calTempFilter(struct cal_temp_filter *temp_filter, float temp_c)
```

Typically, temp_c might be measured every ten seconds and period_s might be 1s. The time_constant_s will be in the range 30-100s depending upon the mechanics, ventilation and design of the ADC or DCCT electronics.

If calTempFilter is called every second to recalculate the filtered temperature then the associated calibration factors should be recalculated as well.

11.2 Calibration Factors

There is a function that can translate V_{Raw} into I_{Dcct} or the reverse from I_{sim} to V_{Raw}.

```
void calCurrent(struct cal_dcct *cal_dcct, struct cal_adc *cal_adc,
                int32_t v_raw, float i_sim, unsigned sim_f, struct cal_current *meas);
```

This needs the DCCT and ADC calibration factors, cal_dcct and cal_adc to be calculated based on the measured (filtered) temperature and the temperature normalised calibration errors for the DCCT and ADC. These factors would normally be recalculated every second if temperature compensation is being used.

The functions to do this are:

```
unsigned calDcctFactors(struct cal_dcct_head *dcct_head, struct cal_event *dcct,
                        float dcct_temp_c, float dcct_temp_coeffs[3], float d_dcct_temp_coeffs[3],
                        struct cal_limits *limits, struct cal_dcct *cal_dcct);

unsigned calAdcFactors (int32_t nominal_adc_gain, struct cal_event *adc,
                        float adc_temp_c, float adc_temp_coeffs[3], float d_adc_temp_coeffs[3],
                        struct cal_limits *limits, struct cal_adc *cal_adc);
```

These require cal_event structures for the DCCT and ADC which contain the temperature normalised calibration errors. These are calculated by measuring known reference voltages (ADC) and currents (DCCT) between once a day and once a year, depending on the precision requirements.

11.3 Temperature normalised calibration errors

Between once a day and once a year the ADC and DCCT gain and offset errors must be measured. These errors are normalised to temperature T_0 (23C) using the temperature coefficients for the device (see chapter 1.2). The ADC errors are relative to a nominal gain which must be initially provided or calculated. Over time the errors will grow and eventually the nominal gain will need to be recalculated.

The ADCs must be calibrated before measuring the DCCT gain and offset errors. The ADC calibration process must always start with the offset measurement followed by the positive and negative gain measurement.

For the internal ADCs, multiplexers and/or analogue switches allow the software to select a short circuit or a positive or negative voltage reference so the calibration can be automatic. For external ADCs, normally this is a manual process that must be coordinated with the acquisition of the reference values.

11.3.1 Averaging the raw ADC acquisitions

To minimise the effects of ADC and measurement noise, a large number of consecutive samples must be averaged. The averaging period should be a multiple of 20 ms to reduce the effect of 50 Hz noise. If the ADC takes 1000 samples per second then 10,000 samples over 10 seconds is a good number. The noise is reduced by the square root of the number of samples, so by a factor of 100 in this case.

The average of 10,000 samples should be done using integer maths (i.e. on the raw ADC values) and overflow must be avoided. The calAverageVraw function is provided in the library to do this:

```
unsigned calAverageVraw(struct cal_average_v_raw *average_v_raw, unsigned num_samples,
                        int32_t v_raw);
```

The acquisition must be initialised by calling the function with num_samples set to the number of samples to average (e.g. 10000). Then the function should be called for this number of acquisitions with num_samples set to zero and v_raw set to the raw ADC value. The function will return the number of acquisitions remaining.

Once this reaches zero the average v_raw value will be available in:

```
average_v_raw->v_raw
```

This can then be used to calculate the cal_event structure.

11.3.2 Calculating the ADC calibration errors

For an internal ADC, the average of raw acquisitions for the offset and the positive and negative gains can be made automatically while for an external ADC it is normally a manual process to connect a short circuit or a positive or negative voltage reference. After each acquisition (starting with the offset measurement) the associated calibration error can be calculated using:

```
void calAdcError(enum cal_idx idx, int32_t v_raw_ave, int32_t nominal_adc_gain,
                 float adc_temp_c, float adc_temp_coeffs[3], float d_adc_temp_coeffs[3],
                 float v_ref_err_ppm, float *v_ref_temp_coeff, struct cal_event *adc);
```

The cal_idx parameter tells the function which error is being calculated: `CAL_OFFSET_V, CAL_GAIN_ERR_POS, or CAL_GAIN_ERR_NEG.`

If the error in the voltage reference is known then it can be provided in v_ref_err_ppm. This is not used for the offset measurement which is assumed to have no error. If the temperature coefficient for the voltage reference is known then it can be provided in *v_ref_temp_coeff. If the temperature coefficient is not known then a NULL pointer should be passed.

11.3.3 Calculating the nominal ADC gain

The function calAdcError needs to know the nominal ADC gain. This can be calculated using a different function:

```
int32_t calAdcNominalGain(int32_t v_offset_raw_ave, int32_t v_pos_raw_ave,
                          float adc_temp_c, float adc_temp_coeffs[3], float d_adc_temp_coeffs[3],
                          float v_ref_err_ppm, float *v_ref_temp_coeff);
```

This is normally done once at the beginning of operation for an ADC and then only when the gain errors grow too large (perhaps once a year).

The nominal ADC gain is calculated by this function such that the temperature normalised positive gain error will be zero. To do this calculation the function needs to have the average raw ADC value for the offset measurement and the positive voltage reference measurement. The error in the positive voltage reference should also be given in v_ref_err_ppm.

If the temperature coefficient for the positive voltage reference is known then it can be provided in *v_ref_temp_coeff. If the temperature coefficient is not known then a NULL pointer should be passed.

11.3.4 Calculating the DCCT calibration errors

The calibration technique for a DCCT depends upon the size and make of DCCT. It may have calibration windings that allow the current through conductor to be simulated, or it may be that a calibrated current is injected into the burden resistor.

Whatever the technique, the ADC must be correctly calibrated first and then the DCCT must be provoked to produce a voltage corresponding to zero current and then positive and negative nominal current. In each case the raw ADC values must be averaged over sufficient samples, in the same way as for the ADC calibration.

The offset calibration should be done first and then the gain calibrations. The calibration errors can be calculated individually after each average acquisition has been completed:

```
void calDcctError(enum cal_idx idx, int32_t v_raw_ave, struct cal_adc *cal_adc,
                  float dcct_temp_c, float dcct_temp_coeffs[3], float d_dcct_temp_coeffs[3],
                  float i_ref_err_ppm, struct cal_dcct_head *dcct_head, struct cal_event *dcct);
```

Note that the function needs the temperature compensated ADC calibration factors cal_adc. If there is a known error in the calibration reference, it can be provided in i_ref_err_ppm.

11.3.5 Date/Time/Temperature stamp for a calibration event

The cal_event structure contains the temperature normalised offset and gain errors and also a date/time and temperature stamp. These can be set using the function:

```
void calEventStamp(struct cal_event *event, uint32_t unix_time, float temp_c);
```

The complete event is normally recorded in a database so the date/time/temperature stamp can help in tracking the evolution of an ADC or DCCT.

Looking inside the cal_event structure:

```
struct cal_event                          // Calibration event for ADC or DCCT
{                                         // PPM values are normalised to temp T0
    float  offset_ppm;                    // Voltage offset in PPM of CAL_V_NOMINAL
    float  gain_err_pos_ppm;              // Gain error in ppm of nominal gain for +ve values
    float  gain_err_neg_ppm;              // Gain error in ppm of nominal gain for -ve values
    float  temp_c;                        // Temperature when calibrated
    float  date_days;                     // Calibration date in days since 1970
    float  time_s;                        // Calibration time in seconds since midnight
};
```

We see that all the fields in the cal_event structure are floats. This is the case so that they can be represented as a single float array property which is not ideal but simplifies the storage of the calibration data in the database. It means that unix_time must be broken into two floats since one float has insufficient resolution to hold a 32 bit integer. The format chosen is to have the number days since the start of 1970 in one float and the number of seconds within the day in a second float.

11.4 **DAC calibration**

The library has DAC calibration support. To use it the DAC output must be measured using the ADC (appropriately calibrated). Three measurements should be made with the raw DAC value 0, +DAC_RAW, -DAC_RAW, where DAC_RAW is chosen to create a voltage of about 70% full scale. For example, for a 20 bit DAC in the FGC2 DAC_RAW is 367000 which results in about 8V on the DAC output.

The raw ADC averages from the three measurements should be passed in v_raw_ave[3] to the function:

```
void calDacInit(struct cal_adc *cal_adc, int32_t v_raw_ave[3], struct cal_dac *cal_dac,
                unsigned resolution, int32_t dac_raw);
```

This will prepare cal_dac structure so that it can be used subsequently with the function:

```
int32_t calDacSet(struct cal_dac *cal_dac, float v_dac);
```

This will take the required DAC voltage v_dac and return the raw integer value that must be written to the DAC in order to achieve this voltage.

The function calDacInit also prepares min and max raw and voltage values in the cal_dac structure:

```
struct cal_dac                                  // DAC calibration
{
    float               v_offset;               // Measured DAC voltage for zero calibration
    float               gain_pos;               // Gain (raw/V) for positive values
    float               gain_neg;               // Gain (raw/V) for negative values
    int32_t             max_dac_raw;            // Max raw value = 2^(resolution-1) - 1
    int32_t             min_dac_raw;            // Min raw value = -2^(resolution-1)
    float               max_v_dac;              // Maximum voltage that can be generated
    float               min_v_dac;              // Minimum voltage that can be generated
};
```

The raw min/max values are used by calDacSet to clip the DAC value to the working range defined by the DAC's resolution.

The min/max voltage limits are available to be used by limits functions in the liblim library.

12. Conclusions

The introduction of temperature compensation promises to significantly reduce measurement errors, provided the DCCT and ADC in the acquisition chain have been calibrated correctly. However, there is a price to be paid in terms of complexity, both at the level of the list of configuration properties, and at the level of the software.

In some cases, using a single calibration value may provide some benefit for lots of systems, based on a statistical analysis of a sample of them. However, the spread in temperature coefficients seen so far shows that we cannot expect miracles from this global approach. For the best results, individual FGCs will need to be calibrated at different temperatures.

13. Changes in this document

13.1 V1.14 July 29th 2010

- PDF version created using a Win7 portable instead of WinXP desktop, which for some reason corrupts the equations

13.2 V1.13 July 23rd 2010

- File converted to Word 2007 format
- Nominal voltage V_{nom} (10V) introduced into the calibration equations
- Chapter on libcal library added
- Entered into EDMS with number 1084984

13.3 V1.12 March 1st 2008

- Error properties expanded to 6 elements to include time and temperature
- Property CAL.VREF.ERR and CAL.VREF.TC reorganized to have [0] unused, [1] applies to the +10V and [2] applies to -10V.
- Automatic calibration now triggered only via the CAL top level property

13.4 V1.11 November 1st 2006

- Property DCCT.PRIMARY_TURNS added to hold N_{turns}
- Property CAL.*.DCCT.GAIN renamed to be DCCT.*.GAIN
- Property CAL.TAU_TDCCT renamed to be DCCT.TAU_TEMP
- Property DCCT.*.TEMPERATURE added
- Property ADC.TEMPERTURE added
- Property CAL.TAU_TADC16 renamed to be ADC.TAU_TEMP
- Property CAL.*.ADC.INTERNAL.VRAW added to report calibration raw values

13.5 V1.10 August 31st 2006

- p11: Nominal gain section changed to clarify that G_{Adc16} will be measured by an external calibration system
- References to G_{Adc16} added to section 0 to improve readability
- References to G_{Adc22} added to chapter 8 to improve readability
- References to G_{Adc16} added to chapter 9 to improve readability
- References to G_{Adc} added to chapter 9.7 to improve readability

13.6 V1.9 August 30th 2006

- This section on changes added
- Reference to FGC1 removed from tables 1 and 2

- ADC16 nominal gains are now only set by the user and are not calculated automatically
- ADC16 nominal gains and errors are now included in the **global** configuration. This will ensure that the errors are logged in the database after every automatic calibration.

13.7 **V1.8 August 1st 2006**

- P5: Property CAL.PBCERR replaced by CAL.EXT_REF_ERR, as the value is also used for DCCT calibrations
- p24: Description of DCCT calibration process expanded to allow an error in the external refernce (CAL.EXT.REF_ERR)
- Eq 56, 58, 59, 61, 62: Formula takes into account the external reference error