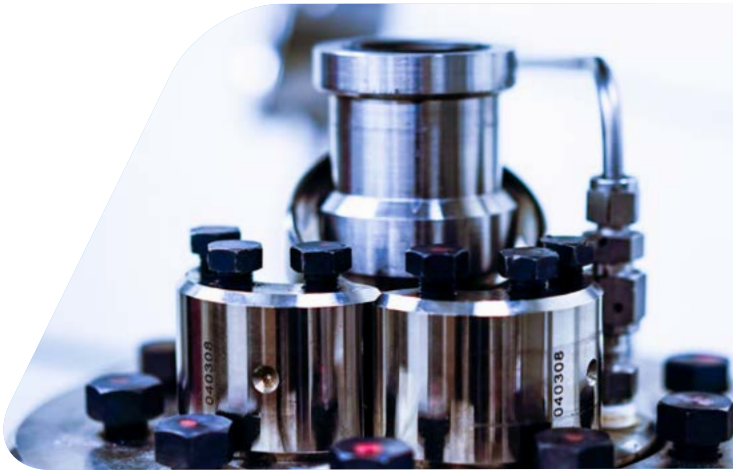
APPLICATION NOTE

CAS DATALOGGERS

Measuring & Sending Sensor Data: Analog to CANbus

Introduction

In automotive or other vehicle engineering and development projects it is sometimes useful to convert analog data and send it via CAN. For example, in automotive applications you may need to read data from one or more sensors that provide voltage output and transmit this data via analog to CANbus so that it can be read along with other data being broadcast by an ECU by a single CANbus data logging device. ADwin data acquisition systems provide an ideal platform to implement a simple solution for this.



Program Description

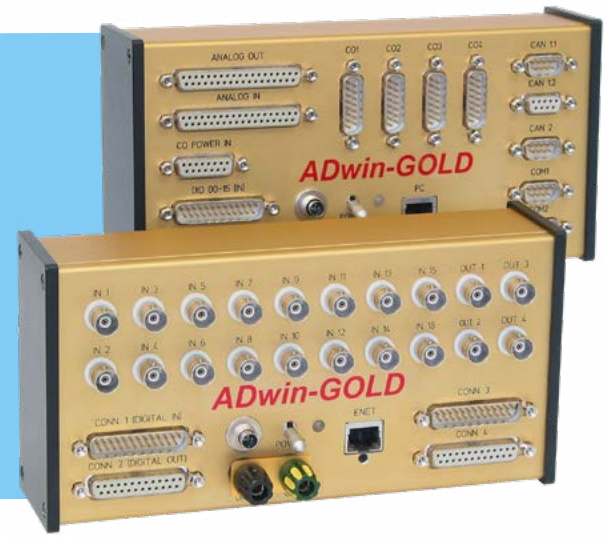
There are a few basic steps to converting sensor data from analog to CANbus data:

1. Read the analog output of the sensor at the desired sample rate.
2. Perform any required signal processing such as filtering or averaging and scale the output to the desired engineering units such as temperature, pressure, etc.
3. Convert the value from engineering units to hexadecimal with any required formatting such as and offset or scale factor.
4. Encode the hex value in a CAN message with the specified ID and byte/bit offset.
5. Either broadcast the message or load it into a register in the case of a polled message.

USAGE

ADwin systems are available in several different models with different analog input capabilities, but they are all suitable for this application when outfitted with the CANbus interface option. The flexibility of the open programming environment of ADwin systems makes it especially easy to read and scale the data. High level functions and built-in message structures allow the generation and transmission of CAN messages with just a few simple statements as shown in the following sample program.

The ADwin operating system is an event based environment which allows periodic message generation without the need for special timing routines. In the sample program, the GLOBALDELAY statement configures the event loop to read the sensor and send the data every millisecond. In this example the data is being broadcast, but it is quite easy to configure the system to use polled message transmission. In this case, the CAN interface can be configured to look for a particular message ID and generate an interrupt to trigger the event loop and send the data.



Other features of the ADwin systems include built-in high level functions and the ability to freely program calculations with very little overhead. I the example below, a single instruction can be used to return the input voltage. Easy to use math functions allow scaling calculations to be done with a single, intuitive equation. These capabilities enable more complex operations such as averaging, filtering and statistical operations. Internal routines which handle casting operations make it easy to manipulate integer, floating point and binary data transparently.

For development, the ADwin architecture provides for transparent shared data between the internal operating environment of the ADwin system and an attached computer.

This greatly simplifies debugging by monitoring the values of integer and floating point variables. The environment also provides for data arrays to enable charting and logging.

Sample Program

The following sample program is for an ADwin-Pro system with a Pro-Ain-32/16, 16 bit analog input card and a Pro-CAN-2, 2 channel high speed CAN interface card.

```
#INCLUDE ADwinPRO_ALL.inc

'* ADWIN-PRO MODULE ADDRESS DEFINITIONS
#DEFINE modAIN32_16 1       'Define address of analog input module
#DEFINE modCAN2       1     'Define address of CAN interface module
#DEFINE CAN_CH1 1           'Define CAN channel

'* DEFINITION OF PAR VALUES
#DEFINE RAW_ADC          PAR_1    'Temporary variable for raw A/D conversion

'* DEFINITION OF FPAR VALUES
#DEFINE OFFSET           FPAR_1  'Sensor zero offset for engineering units
#DEFINE GAIN             FPAR_2  'Sensor gain for engineering units
#DEFINE SCALE            FPAR_3  'Scale factor for CAN message
#DEFINE PRESSURE         FPAR_4  'Sensor value in engineering units

DIM CANDATA AS LONG              'Sensor value for CAN message

INIT:
  GLOBALDELAY =  40000                  'Update rate in ticks = 1 msec.
 INIT_CAN(modCAN2, CAN_CH1)      '* Initialize CAN Controller
  FPAR_2 = SET_CAN_BAUDRATE(modCAN2, CAN_CH1, 250000)      'For J1939=350
kBit
```

```
'PGN 65263 is for oil pressure, send as J1939, 29 bit identifier of 018FEEF
 EN_TRANSMIT(modCAN2, CAN_CH1, 1, 018FEEF00h, 1)
 GAIN = 100.0          '0-10V out is 0-100 PSI
 OFFSET = 0.0          'default offset is 0 PSI
 SCALE = 4.0           'Std. scale factor for message is 4

EVENT:
' Read ADC, convert counts to pressure and scale based on CAN std.
 RAW_ADC=ADC(modAIN32_16,2)
 PRESSURE = (((RAW_ADC - 32767)/32767)* GAIN) - OFFSET
 CANDATA = PRESSURE * SCALE
 PAR_2 = CANDATA     'Debug

' Build CAN Message - pressure is in byte 4 so pad leading bytes with FF
 CAN_MSG[1] = 0FFh
 CAN_MSG[2] = 0FFh
 CAN_MSG[3] = 0FFh
 CAN_MSG[4]= CANDATA    ' BYTE 4 = ENGINE OIL PRESSURE
 CAN_MSG[5] = 0FFh   ' Add trailing pad byte
 CAN_MSG[9] = 6      ' message length is 6 bytes

'Send message
 TRANSMIT(modCAN2, CAN_CH1, 1)

END:
```

For further information on ADwin systems, analog to CANbus, or to find the ideal solution for your application-specific needs, contact a CAS Data Logger Application Specialist at **(800) 956-4437** or www.DataLoggerInc.com.