App. Note Code: 2RA-K

APPLICATION NOTE

EKO MS-710 and MS-712 Broadband Infrared Spectroradiometers (WISER Series) with CR3000 Datalogger

4/14

Table of Contents

PDF viewers: These page numbers refer to the printed version of this document. Use the PDF reader bookmarks tab for links to specific sections.

Figure

EKO MS-710 and MS-712 Broadband Infrared Spectroradiometers (WISER Series) with CR3000 Datalogger

1. Introduction

The EKO Instruments MS-710 and MS-712 Spectroradiometers (WISER) measure visible (VIS) and near infrared (NIR) total sky spectral irradiance. The combination of the EKO Instruments MS-710 and MS-712 Spectroradiometers (WISER) is known as the WISER series. The WISER measures spectral flux density of solar radiation. Each sensor is one of the few spectroradiometers that have been designed to be installed outdoors in all weather conditions. Each sensor has NIST traceability and is made with a high quality hermetically sealed dome and diffuser that couple lambertian incident light to an optical fiber before emitting the light onto a diffraction grating. The grating projects photons of varying wavelengths across a detector array. The MS-710 uses a silicon detector (Si) while the MS-712 uses an indium gallium arsenide (InGaAs) detector. The magnitude of the signal on each pixel is converted to spectral irradiance values. The optics of both sensors are kept at stabile temperatures using a peltier element. A drawing of the sensor head configuration is provided below in FIGURE 1-1.

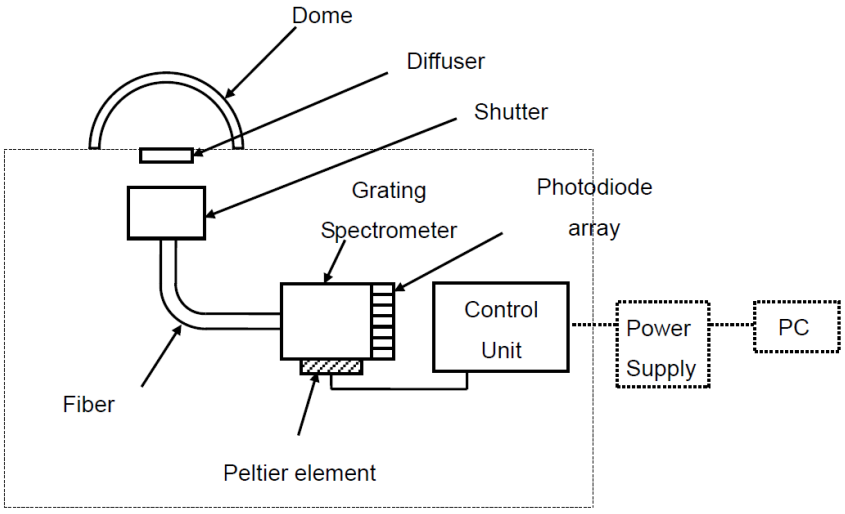

FIGURE 1-1. MS-710/712 sensor head and configuration with external power supply and PC as shown in EKO documentation

The WISER is designed to measure spectral flux density of solar radiation. Both sensors can be deployed permanently outdoors. Common applications include providing data for solar radiation studies by renewable energy companies or atmospheric science researchers. EKO Instruments supplies the sensor with a standard software package for operating the sensor with a PC. This application note presents the Campbell Scientific CR3000 datalogger as an alternative method of sensor control and data management. Benefits of

using a Campbell Scientific system would include but it is not limited to LoggerNet data handling, robust electronics, and excellent customer support.

Use of this application note assumes prior experience with Campbell Scientific dataloggers.

2. Specifications

Specifications ***(Typical)***	**MS-710**	**MS-712**
Response time (Exposure time)	10 ms to 5 s	
Non-stability (change/year)	±1%	
Non-linearity (at 1000W/m^2)		
Directional response	<7%	
Temp. response (–20°C to 50°C)	<±5%	
Control temperature	25ºC	5ºC
Operating temperature range (°C)	–10 to +40ºC	
Cable length	10 m	
Wavelength range	350 to 950 nm	950 to 1700 nm
Spectral resolution	5 nm	10 nm
Wavelength accuracy	<1.5 nm	<1.5 nm
Dimensions	ø 220 x 175 (H)	ø 310 x 270 (H)
Weight	4.5 kg	7.5 kg
Software	Applicable to MS-Windows 95/98 SPDac-for measurement, SPMan-for data analysis Automatic calculation of spectral match accuracy (IEC904-9)	
Unit of output	W/m^{2}µm	
Power supply	External control and power supply 100 Vac ~240 Vac / 12 Vdc (50 W) / 5 Vdc (5 W) / operating conditions 0 to +40ºC	
Serial interface	RS-232, RS-422	

3. Overview

The WISER outputs an RS-422 serial signal to the EKO power supply boxes. The power supply then converts the RS-422 to RS-232 to be retrieved by a PC or data acquisition system. From the EKO power boxes, the WISER is intended to be connected to a PC and operated using the software package provided by EKO. Data provided by this method will be provided in a .CSV format and stored on the PC hard drive.

When using a Campbell Scientific system, the data will be provided in comma delimited TOB3 ASCII files. Each sensor requires a baud rate of 38400 bps with a bit structure of 8 data bits, 1 stop bit and no parity. All collected data on the datalogger should be redundantly stored on an external CompactFlash (CF) card if one is available. A CFM100 or NL115 connected to the CR3000 is used to write to the data to a CF card. With the datalogger having roughly 4 MB of memory, the system will quickly begin to overwrite older data stored exclusively on the datalogger.

There are several reasons why a customer might want to interface the WISER with a CR3000 Micrologger. Often they are added to an existing suite of instruments on a data acquisition system that uses a CR3000. Of the Campbell Scientific PakBus dataloggers, the CR3000 provides the best combination of sensor input terminals and processing power for measuring the WISER in a solar monitoring system. The CR3000 also uses CRBasic, which includes a powerful checksum routine that ensures accurate measurements and data collection, although it increases processing time. Alternatively, smaller PakBus dataloggers may be used with an omission of the checksum routine.

The sensor data will be measurements of spectral power density values in units of $Wm^{-2}\mu m^{-1}$ at approximately 3 nm bandwidths with an accuracy of <0.3 nm. The total measurement bandwidth spans from 350 nm to 1700 nm providing 1536 irradiance values per measurement. There is a limitation in the sampling frequency when using the Campbell Scientific system. First, the detector requires 10 ms to 5 s of exposure for a signal to be measured. Second, the processing time to output a measurement after having a checksum calculated and verified takes some time and limits the system to approximately 1 minute measurements scans. A PC system could make faster measurements but would need a user to manually make the measurements or specific code written to automate the measurements on the PC.

4. Installation and Operation

4.1 Wiring

The WISER requires the EKO power supply for powering the device and for conversion between RS-422 to RS-232. For wiring of the MS-710/712 to the power supply, please consult the EKO MS-710/712 manual. The RS-232 cable can be connected to a PC, but for this application it will be connected to the CS I/O port of the CR3000 datalogger.

An SC105 interface is used to connect the straight through serial cable from the power supplies to the datalogger. The sensors are connected to the CS I/O port on the datalogger with an SC12 cable. Device configuration must be used to configure each SC105 for different SDC addresses. Please contact a Campbell Scientific AE for support.

4.2 Data Analysis

Campbell Scientific's *Real Time Monitoring and Control* (*RTMC* and *RTMC Pro*) software can create displays of the results. However, for displays of time series of irradiance values, *RTMC* is limited to 15 values. Newer versions of *RTMC Pro* are not limited. *Matlab*® or another statistical analysis software or environment, such as *R*, is better suited for analyzing the data and producing visualizations.

4.3 Programming Example

```
'CR3000 Series Datalogger
'Sample code for integrating an EKO MS-710 and MS-712 Spectroradiometer with a Campbell
‘Scientific CR3000 Datalogger

PreserveVariables

Public BattVolt,DeviceTemp

Const pixels_count_MS710 = 1024
Const input_buffer_len_MS710 = pixels_count_MS710 * 15 + 100
Const wavelength_coefs_count_MS710 =6
Const serial_port_MS710 = COMSDC7

Const pixels_count_MS712 = 512
Const input_buffer_len_MS712 = pixels_count_MS712 * 15 + 100
Const wavelength_coefs_count_MS712 =6
Const serial_port_MS712 = COMSDC8
Const Standard = 0
Const Deluxe = 1

Public wavelength_coefs_MS710(wavelength_coefs_count_MS710)
Public irradiance_MS710(pixels_count_MS710)
Units irradiance_MS710 = "W/m^2/um"
Public last_error_MS710 As String * 128
Public exposure_time_MS710
Units exposure_time_MS710 = "ms"

Public wavelength_coefs_MS712(wavelength_coefs_count_MS712)
Public irradiance_MS712(pixels_count_MS712)
Units irradiance_MS712 = "W/m^2/um"
Public last_error_MS712 As String * 128
Public exposure_time_MS712
Units exposure_time_MS712 = "ms"


DataTable(MS710, true, -1)
  DataInterval(0, 5, min, 10)
  CardOut(0,-1)
  Sample(wavelength_coefs_count_MS710, wavelength_coefs_MS710, IEEE4)
  FieldNames("c0,c1,c2,c3,c4,c5")
  Sample(pixels_count_MS710, irradiance_MS710, IEEE4)
EndTable

DataTable(MS712, true, -1)
  DataInterval(0, 5, min, 10)
  CardOut(0,-1)
  Sample(wavelength_coefs_count_MS712, wavelength_coefs_MS712, IEEE4)
  FieldNames("c0,c1,c2,c3,c4,c5")
  Sample(pixels_count_MS712, irradiance_MS712, IEEE4)
EndTable

DataTable(ms710_debug, true, 1000)
  Sample(1, last_error_MS710, String)
EndTable
```

```
DataTable(ms712_debug, true, 1000)
  Sample(1, last_error_MS712, String)
EndTable


BeginProg

  'The MS-710 and MS-712 require 38400 baud, 8 data bits, and 1 stop bit
  SerialOpen(serial_port_MS710, 38400, 3, 0, input_buffer_len_MS710)
  SerialOpen(serial_port_MS712, 38400, 3, 0, input_buffer_len_MS710)

  Scan (3,Sec,3,0)
    PanelTemp (DeviceTemp,_60Hz)
    Battery (BattVolt)
  NextScan

  Const xsum_buffer_len_MS710 = 25
  Dim input_buffer_MS710 As String * input_buffer_len_MS710
  Dim xsum_buffer_MS710 As String * xsum_buffer_len_MS710
  Dim ms710_vals_ok As Boolean
  Dim ms710_retry_count As Long
  Dim i As Long
  Dim calc_xsum_MS710 As Long
  Dim sent_xsum_MS710 As Long

  Const xsum_buffer_len_MS712 = 25
  Dim input_buffer_MS712 As String * input_buffer_len_MS712
  Dim xsum_buffer_MS712 As String * xsum_buffer_len_MS712
  Dim ms712_vals_ok As Boolean
  Dim ms712_retry_count As Long
  Dim calc_xsum_MS712 As Long
  Dim sent_xsum_MS712 As Long

  SlowSequence
  Scan (5,Min,0,300)
    'Initialise the variables to NaN

     For i = 1 To pixels_count_MS710
      irradiance_MS710(i) = NaN
    Next
    For i = 1 To wavelength_coefs_count_MS710
      wavelength_coefs_MS710(i) = NaN
    Next
    last_error_MS710 = ""
    ms710_retry_count = 0

    'Read the wavelength coefficients from the sensor detector
    For i = 1 To wavelength_coefs_count_MS710
      SerialFlush(serial_port_MS710)
      SerialOut(serial_port_MS710, "C" + (i - 1) + CHR(13), "", 0, 0)
      SerialIn(input_buffer_MS710, serial_port_MS710, 2 / 0.01, 13, input_buffer_len_MS710)
      If InStr(1, input_buffer_MS710, ",OK", 2) > 0 Then
        wavelength_coefs_MS710(i) = input_buffer_MS710
      Else
        last_error_MS710 = "read C" + (i - 1) + " failed: '" + input_buffer_MS710 + "'"
        CallTable(ms710_debug)
        ExitFor
      EndIf
    Next

    If Len(last_error_MS710) = 0 Then
      'Initiate an "auto" measurement
      SerialFlush(serial_port_MS710)
      SerialOut(serial_port_MS710, "AM,0" + CHR(13), "", 0, 0)
      SerialIn(input_buffer_MS710, serial_port_MS710, 15 / 0.01, 13, input_buffer_len_MS710)
      If InStr(1, input_buffer_MS710, ",OK", 2) > 0 Then
        'The exposure time is returned along with the OK signal
        exposure_time_MS710 = input_buffer_MS710
```

```
        'Attempt to read the measured values from the sensor
        ms710_vals_ok = false
        ms710_retry_count = 0

 Do While NOT ms710_vals_ok AND ms710_retry_count < 3
          'Issue the command to send the values and read these into the input buffer
          SerialFlush(serial_port_MS710)
          SerialOut(serial_port_MS710, "DT,0" + CHR(13), "", 0, 0)
          SerialIn(input_buffer_MS710, serial_port_MS710, 2 / 0.01, 13, input_buffer_len_MS710)

 If InStr(1, input_buffer_MS710, ",OK", 2) > 0 Then
            'Calculate the check sum of the data buffer.  This will need to include
            ' the CR that was stripped off by SerialIn()
            calc_xsum_MS710 = 0
            For i = 1 To Len(input_buffer_MS710)
              calc_xsum_MS710 = (calc_xsum_MS710 XOR ASCII(input_buffer_MS710(1, 1, i))) AND
&hFF
            Next i
            calc_xsum_MS710 = (calc_xsum_MS710 XOR 13) AND &hff

            'Verify the check sum for the data values
            SerialFlush(serial_port_MS710)
            SerialOut(serial_port_MS710, "CS" + CHR(13), "", 0, 0)
            SerialIn(xsum_buffer_MS710, serial_port_MS710, 2 / 0.01, 13, xsum_buffer_len_MS710)
            If InStr(1, xsum_buffer_MS710, ",OK", 2) > 0 Then
              'Compare the checksum calculated above with that returned
              ' by the CS command.  These should be equal.  If not, retry the
              ' command to collect data
              sent_xsum_MS710 = HexToDec(xsum_buffer_MS710)
              If sent_xsum_MS710 = calc_xsum_MS710 Then
                ms710_vals_ok = true
                SplitStr(irradiance_MS710, input_buffer_MS710, ",", pixels_count_MS710, 0)

                'Verify the values that were read
                For i = 1 To pixels_count_MS710
                  If irradiance_MS710(i) < 0.0 Then
                    irradiance_MS710(i) = 0.0
                  ElseIf irradiance_MS710(i) >= 1000 OR irradiance_MS710(i) = NaN Then
                    last_error_MS710 = "invalid irradiance(" + i + "): " + irradiance_MS710(i)
                    CallTable(ms710_debug)
                    ms710_vals_ok = false
                    ms710_retry_count = ms710_retry_count + 1
                    irradiance_MS710(i) = NaN
                  EndIf
                Next
              Else
                last_error_MS710 = "checksums don't match: calc=" + calc_xsum_MS710 + " sent="
+ sent_xsum_MS710
                CallTable(ms710_debug)
                ms710_retry_count = ms710_retry_count + 1
              EndIf
            Else
              last_error_MS710 = "invalid CS ack: " + xsum_buffer_MS710
              CallTable(ms710_debug)
              ms710_retry_count = 3
            EndIf
          Else
            last_error_MS710 = "invalid DT,0 ack"
            CallTable(ms710_debug)
            ms710_retry_count = ms710_retry_count + 1
          EndIf
        Wend
      Else
        last_error_MS710 = "measure failed: '" + input_buffer_MS710 + "'"
        CallTable(ms710_debug)
      EndIf
    EndIf
```

```
  CallTable(Ms710)

'Initialise the variables to NaN
  For i = 1 To pixels_count_MS712
    irradiance_MS712(i) = NaN
  Next
  For i = 1 To wavelength_coefs_count_MS712
    wavelength_coefs_MS712(i) = NaN
  Next
  last_error_MS712 = ""
  ms712_retry_count = 0

  'Read the wavelength coefficients from the sensor
  For i = 1 To wavelength_coefs_count_MS712
    SerialFlush(serial_port_MS712)
    SerialOut(serial_port_MS712, "C" + (i - 1) + CHR(13), "", 0, 0)
    SerialIn(input_buffer_MS712, serial_port_MS712, 2 / 0.01, 13, input_buffer_len_MS712)
    If InStr(1, input_buffer_MS712, ",OK", 2) > 0 Then
      wavelength_coefs_MS712(i) = input_buffer_MS712
    Else
      last_error_MS712 = "read C" + (i - 1) + " failed: '" + input_buffer_MS712 + "'"
      CallTable(ms712_debug)
      ExitFor
    EndIf
  Next

  If Len(last_error_MS712) = 0 Then
    'Initiate an "auto" measurement
    SerialFlush(serial_port_MS712)
    SerialOut(serial_port_MS712, "AM,0" + CHR(13), "", 0, 0)
    SerialIn(input_buffer_MS712, serial_port_MS712, 15 / 0.01, 13, input_buffer_len_MS712)
    If InStr(1, input_buffer_MS712, ",OK", 2) > 0 Then
      'The exposure time is returned along with the OK signal
      ' exposure_time_MS712 = input_buffer_MS712
      exposure_time_MS712 = 5000
      'Attempt to read the measured values from the sensor
      ms712_vals_ok = false
      ms712_retry_count = 0
      Do While NOT ms712_vals_ok AND ms712_retry_count < 3
        'Issue the command to send the values and read these into the input buffer
        SerialFlush(serial_port_MS712)
        SerialOut(serial_port_MS712, "DT,0" + CHR(13), "", 0, 0)
        SerialIn(input_buffer_MS712, serial_port_MS712, 2 / 0.01, 13, input_buffer_len_MS712)
        If InStr(1, input_buffer_MS712, ",OK", 2) > 0 Then
          'Calculate the check sum of the data buffer.  This will need to include
          ' the CR that was stripped off by SerialIn()
          calc_xsum_MS712 = 0
          For i = 1 To Len(input_buffer_MS712)
            calc_xsum_MS712 = (calc_xsum_MS712 XOR ASCII(input_buffer_MS712(1, 1, i))) AND &hFF
          Next i
          calc_xsum_MS712 = (calc_xsum_MS712 XOR 13) AND &hff

          'Verify the check sum for the data values
          SerialFlush(serial_port_MS712)
          SerialOut(serial_port_MS712, "CS" + CHR(13), "", 0, 0)
          SerialIn(xsum_buffer_MS712, serial_port_MS712, 2 / 0.01, 13, xsum_buffer_len_MS712)
          If InStr(1, xsum_buffer_MS712, ",OK", 2) > 0 Then
            'Compare the checksum calculated above with that returned
            ' by the CS command.  These should be equal.  If not, retry the
            ' command to collect data
            sent_xsum_MS712 = HexToDec(xsum_buffer_MS712)
            If sent_xsum_MS712 = calc_xsum_MS712 Then
              ms712_vals_ok = true
              SplitStr(irradiance_MS712, input_buffer_MS712, ",", pixels_count_MS712, 0)
```

```
                'Verify the values that were read
                For i = 1 To pixels_count_MS712
                  If irradiance_MS712(i) < 0.0 Then
                    irradiance_MS712(i) = 0.0
                  ElseIf irradiance_MS712(i) >= 1000 OR irradiance_MS712(i) = NaN Then
                    last_error_MS712 = "invalid irradiance(" + i + "): " + irradiance_MS712(i)
                    CallTable(ms712_debug)
                    ms712_vals_ok = false
                    ms712_retry_count = ms712_retry_count + 1
                    irradiance_MS712(i) = NaN
                  EndIf
                Next
              Else
                last_error_MS712 = "checksums don't match: calc=" + calc_xsum_MS712 + " sent="
+ sent_xsum_MS712
                CallTable(ms712_debug)
                ms712_retry_count = ms712_retry_count + 1
              EndIf
            Else
              last_error_MS712 = "invalid CS ack: " + xsum_buffer_MS712
              CallTable(ms712_debug)
              ms712_retry_count = 3
            EndIf
          Else
            last_error_MS712 = "invalid DT,0 ack"
            CallTable(ms712_debug)
            ms712_retry_count = ms712_retry_count + 1
          EndIf
        Wend
      Else
        last_error_MS712 = "measure failed: '" + input_buffer_MS712 + "'"
        CallTable(ms712_debug)
      EndIf
    EndIf
    CallTable(Ms712)

  NextScan

EndProg
```

5. Attributions

Microsoft® and *Excel*® are registered trademarks of Microsoft Corporation.

Matlab® is a registered trademark of MathWorks, Inc.

R is a product of r-project.org

Campbell Scientific Companies

Campbell Scientific, Inc. (CSI)
815 West 1800 North
Logan, Utah 84321
UNITED STATES
www.campbellsci.com • info@campbellsci.com

Campbell Scientific Africa Pty. Ltd. (CSAf)
PO Box 2450
Somerset West 7129
SOUTH AFRICA
www.csafrica.co.za • cleroux@csafrica.co.za

Campbell Scientific Australia Pty. Ltd. (CSA)
PO Box 8108
Garbutt Post Shop QLD 4814
AUSTRALIA
www.campbellsci.com.au • info@campbellsci.com.au

Campbell Scientific (Beijing) Co., Ltd.
8B16, Floor 8 Tower B, Hanwei Plaza
7 Guanghua Road
Chaoyang, Beijing 100004
P.R. CHINA
www.campbellsci.com • info@campbellsci.com.cn

Campbell Scientific do Brasil Ltda. (CSB)
Rua Apinagés, nbr. 2018 ─ Perdizes
CEP: 01258-00 ─ São Paulo ─ SP
BRASIL
www.campbellsci.com.br • vendas@campbellsci.com.br

Campbell Scientific Canada Corp. (CSC)
14532 – 131 Avenue NW
Edmonton AB T5L 4X4
CANADA
www.campbellsci.ca • dataloggers@campbellsci.ca

Campbell Scientific Centro Caribe S.A. (CSCC)
300 N Cementerio, Edificio Breller
Santo Domingo, Heredia 40305
COSTA RICA
www.campbellsci.cc • info@campbellsci.cc

Campbell Scientific Ltd. (CSL)
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
UNITED KINGDOM
www.campbellsci.co.uk • sales@campbellsci.co.uk

Campbell Scientific Ltd. (CSL France)
3 Avenue de la Division Leclerc
92160 ANTONY
FRANCE
www.campbellsci.fr • info@campbellsci.fr

Campbell Scientific Ltd. (CSL Germany)
Fahrenheitstraße 13
28359 Bremen
GERMANY
www.campbellsci.de • info@campbellsci.de

Campbell Scientific Spain, S. L. (CSL Spain)
Avda. Pompeu Fabra 7-9, local 1
08024 Barcelona
SPAIN
www.campbellsci.es • info@campbellsci.es

Please visit www.campbellsci.com to obtain contact information for your local US or international representative.