# Interfacing a Vaisala WXT520 Weather Transmitter to a *dataTaker* DT80 Series Data Logger

The Vaisala WXT520 is a lightweight weather transmitter which has the capability to measure six weather properties including wind speed and direction, temperature, relative humidity, atmospheric pressure and precipitation. The weather transmitter provides the option of outputting data using RS-232, RS-485 or SDI-12 protocols. This application note explores the connection of the WXT520 to the *dataTaker* DT80 Series 'Serial Sensor' port and compares the programs and sample speeds of the different methods.

The methods used in this application note assume that the user has the *dataTaker* DeTransfer program and the Vaisala Configuration Tool installed on their PC.

## 1 Prerequisites

- Nil

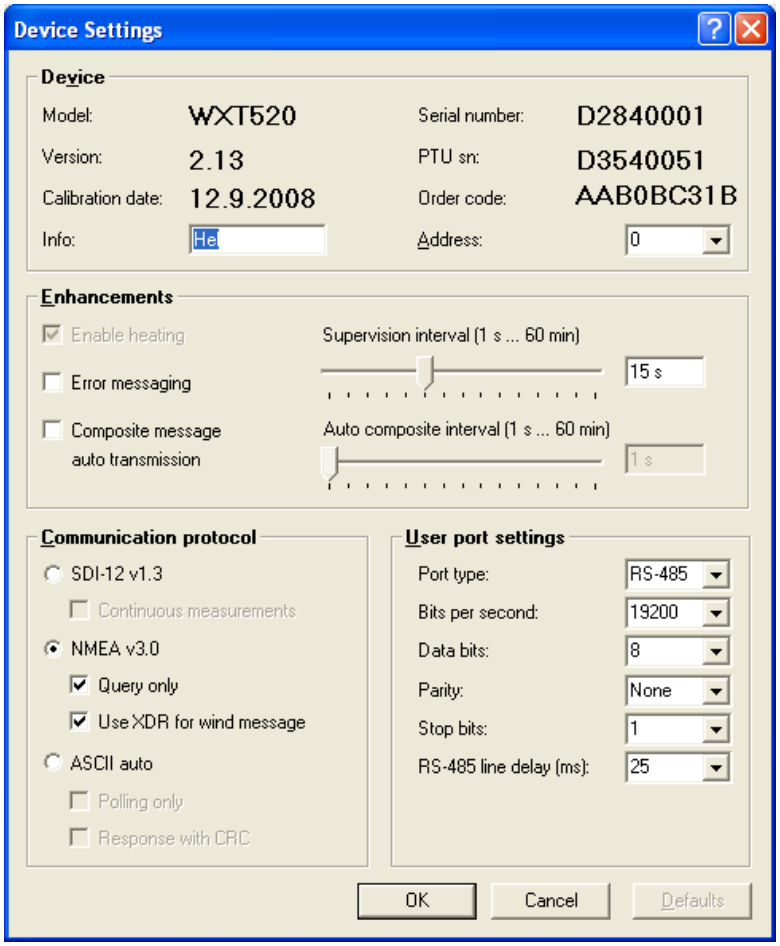
## 2 Required Equipment

- *dataTaker* DT80 series data logger
- Vaisala WXT520 Weather Transmitter
- Connection cables for either RS-232 RS-485 or SDI-12
- Vaisala USB configuration cable
- PC with free USB port
- Vaisala Instrument Tool software
- DeTransfer software

## 3 Process

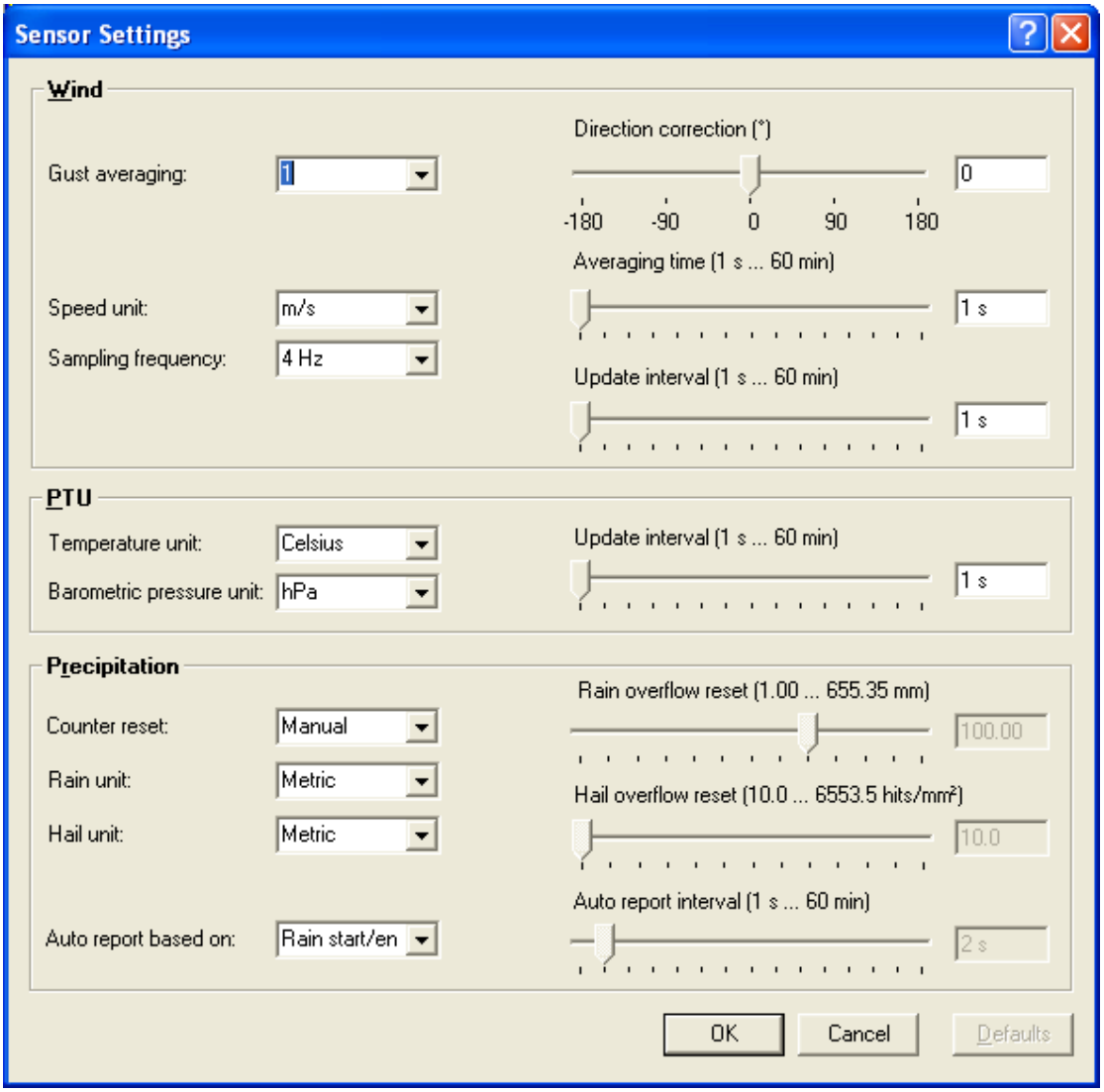
### 3.1 Set parameters on the WXT520

- Connect the WXT520 to the PC using the USB configuration cable and start the Vaisala Instrument Tool.
- Click **Settings → Device** then apply the settings as per the following screenshot (similarly with the sensor and message settings):




**NOTES:**
These settings apply if using the RS-485 protocol, if you are using the RS-232 protocol, the only setting that would be different to those above would be the 'port type', which would be 'RS-232'.

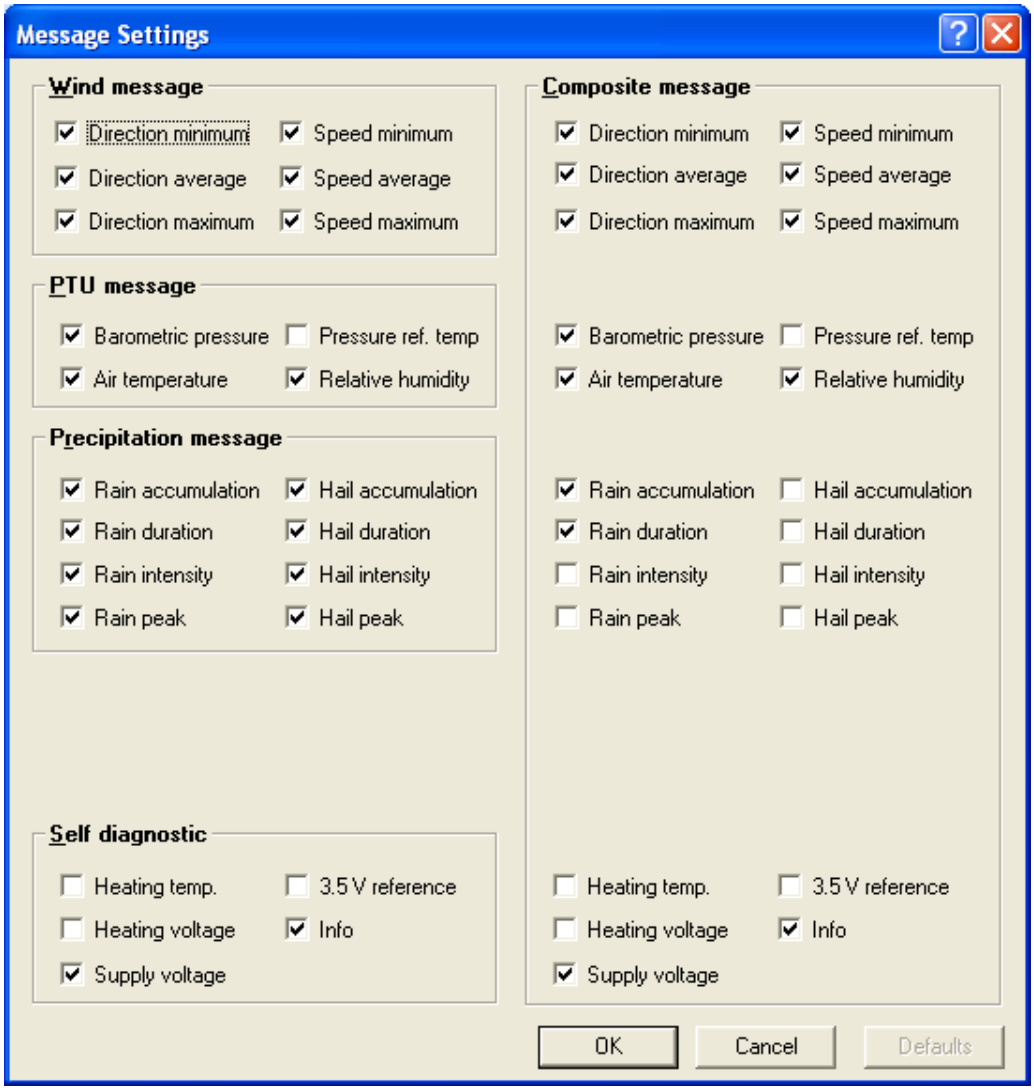
If you are using SDI-12 for your connection method, you will need to select 'SDI-12 v1.3' and 'Continuous measurements' under communication protocol and a 'port type' of 'SDI-12'.




The important settings here are:

- Metric units
- An averaging time of 1 second
- Manual counter reset

We use the low averaging time because this provides us with real time data that we can average later in the logger (so that we may record more information if required).




These settings are standard; please ensure that they match the settings in your WXT520.

## 3.2 Settings on the data logger

The DT80 can be set up at any time by issuing commands through DeTransfer, by executing a job or by assigning actions to a function button. See 'User Defined Functions' in the *dataTaker* user manual for more information about setting up function buttons.

The PROFILE commands in the logger are used to modify all settings in the logger, including the serial port. The following commands (sent to the *dataTaker* through DeTransfer) will set up all parts of the serial sense port, to which the WXT520 will be connected, in either RS232 or RS485 modes:

```
PROFILE "SERSEN_PORT" "MODE"="RS485" ' or RS232
PROFILE "SERSEN_PORT" "BPS"="19200"
PROFILE "SERSEN_PORT" "DATA_BITS"="8"
PROFILE "SERSEN_PORT" "STOP_BITS"="1"
PROFILE "SERSEN_PORT" "PARITY"="NONE"
PROFILE "SERSEN_PORT" "FLOW"="NONE"
```

## 3.3 Wiring

The following chart details the M12 connections between the WXT520 and the DT80. The colours represent the wire core-colours in the M12 cable.

| | Wire Colour and destination | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode** | **Blue** | **Grey** | **White** | **Green** | **Pink** | **Yellow** | **Brown** | **Red/Clear** |
| **RS232** | Rx | - | Tx | GND | - | - | - | GND |
| **RS485** | Tx | RTS | - | - | - | - | - | GND |
| **SDI-12** | 5D | - | 5D | GND | - | - | $V_{IN}$ | GND |

For more information on wiring, please refer to 'Wiring Using the 8-pin M12 connector' on page 46 of the WXT520 user manual.

## 3.4 Diagnostics

### *3.4.1 Snooping*

Snooping is used to monitor the activity on data lines such as serial connections. This can be useful when diagnosing communication issues between RS-232 devices.

The *logger* has the ability to communicate directly with serial sensors using the SSDIRECT command in DeTransfer. This gives us the ability to snoop the communications lines between the two devices.

**Example**
Whilst the logger is connected to the PC through the USB or Ethernet interface, and connected to the WXT520 through the RS-232 interface, entering the following lines in DeTransfer returns the ID and wind data from the WXT520:

```
SSDIRECT 1 "^M^J" 'direct serial mode
?                 'request ID from the WXT520
0R1               'request weather data from channel 0
ENDSSDIRECT       'end direct serial mode
```

The Vaisala Online Monitor can also be used to snoop the data lines. This requires the USB service cable to be plugged into the WXT520.

For further information on diagnostics, and commands, please consult the DT80 user manual under the section "Sensors & Channels" and "Generic Serial Channel". Also consult the "Vaisala Weather Transmitter Users Guide".

## 3.5 Sample Programs

The below code samples are intended to demonstrate only the basic capabilities of the WXT520 and DT80 data logger combination

### *3.5.1 RS-232 and RS-485 Protocols*

The programs for RS-232 and RS-485 protocols will be the same, except that the serial sensor (SERSEN) port mode will be different.

### 3.5.1.1 Sample Program 1 – Simple RS-485 Demonstration Program

```
BEGIN "WS_SIM" 'SIMPLE RS-485 DEMONSTRATION PROGRAM
    LOGON 'Enable logging
'=====================================================
'     Schedule A
'   - Logs up to 1MB of data records, overwrites when full
'   - Runs every 10 seconds
'=====================================================
RA(DATA:OV:1MB)10S
    'Wind Data (clear buffer using \\e first)
    1SERIAL("\\e{0R1\\013\\010}",0.3,W)  'send the command 0R1
    1SERIAL("\\m[D,0,A,]%d[11CV]",0.3,W) 'read in wind direction
    1SERIAL("\\m[M,0,S,]%f[10CV]",0.3,W) 'read in wind speed

    'Temperature, Humidity, Pressure (THP)
    1SERIAL("\\e{0R2\\013\\010}",0.3,W)     'send the command 0R2
    1SERIAL("\\m[$WIXDR,C,]%f[20CV]",0.3,W) 'read in temperature
    1SERIAL("\\m[,H,]%f[21CV]",0.2,W)       'read in humidity
    1SERIAL("\\m[,0,P,]%f[22CV]\\e",0.3,W)  'read in pressure

    'Precipitation
    1SERIAL("\\e{0R3\\013\\010}",0.3,W)     'send the command 0R3
    1SERIAL("\\m[$WIXDR,V,]%f[30CV]",0.3,W) 'read in rain accumul.
    1SERIAL("\\m[Z,]%d[31CV]\\e",0.3,W)     'read in rain duration

    'add name and units to the data and log
    10CV("Wind Speed~m/s")
    11CV("Wind Dir.~Deg")
    20CV("Temperature ~Deg C")
    21CV("Humidity ~%RH")
    22CV("Pressure ~hPa")
    30CV("Rain Accumulation ~mm")
    31CV("Rain Duration ~s")
END
```

### 3.5.1.2 Sample Program 2 - Advanced RS-485 Demonstration Program

```
BEGIN "WS_RS485" 'ADVANCED DEMONSTRATION PROGRAM
'=====================================================
'Initial Set Up
    1SSPWR=1 'turn on the power to the weather transmitter
    FUNCTION1="Reset Counters"{XD} 'set up function button
    3..5CV(W)=0 110..140CV(W)=0 'clear variables
    1SERIAL("\\e{0RU,Z=M\\013\\010}",0.3,W)'set manual reset
    1SERIAL("\\e{0WU,I=1,A=5\\013\\010}",0.3,W) 'update interval=1
second and averaging = 5 seconds
    LOGON 'turn on logging for all schedules
'=====================================================

'=====================================================
'     Schedule A
'   - Logs up to 5MB of data records, overwrites when full
'   - Runs every 5 seconds
'=====================================================
RA(DATA:OV:5MB)5S
    'Wind Data
    1SERIAL("\\e{0R1\\013\\010}",0.3,W)  'send the command 0R1
    1SERIAL("\\m[D,0,A,]%d[11CV]",0.3,W) 'read in wind direction
    1SERIAL("\\m[M,0,S,]%f[10CV]",0.3,W) 'read in wind speed

    'Temperature, Humidity, Pressure (THP)
    1SERIAL("\\e{0R2\\013\\010}",0.3,W)      'send the command 0R2
    1SERIAL("\\m[$WIXDR,C,]%f[20CV]",0.3,W) 'read in temperature
    1SERIAL("\\m[,H,]%f[21CV]",0.2,W)        'read in humidity
    1SERIAL("\\m[,0,P,]%f[22CV]\\e",0.3,W)   'read in pressure

    'Precipitation
    1SERIAL("\\e{0R3\\013\\010}",0.3,W)      'send the command 0R3
    1SERIAL("\\m[$WIXDR,V,]%f[30CV]",0.3,W) 'read in rain accumul.
    1SERIAL("\\m[Z,]%d[31CV]\\e",0.3,W)      'read in rain duration

    'add name and units to the data and log
    10CV("5sec_Wind Speed~m/s",NL)
    11CV("5sec_Wind Dir.~Deg",NL)
    20CV("5sec_Temperature ~Deg C",NL)
    21CV("5sec_Humidity ~%RH",NL)
    22CV("5sec_Pressure ~hPa",NL)

    'code to allow calculation of average wind direction as a vector
    12CV(W)=12CV+10CV*COS(D2R(11CV)) 'Sum x components
    13CV(W)=13CV+10CV*SIN(D2R(11CV)) 'Sum y components
    120CV(W)=120CV+20CV 'Sum of temperature
    121CV(W)=121CV+21CV 'Sum of humidity
    122CV(W)=122CV+22CV 'Sum of pressure
    3CV(W)=3CV+1 'Increment number of scans

'=====================================================
'     Schedule B
'   - Logs up to 5MB of data records, overwrites when full
'   - Runs every 1 minute
'=====================================================
RB(DATA:OV:5MB)1M
    'Calculate mean wind magnitude:
```

```
    110CV(W)=SQRT((12CV*12CV)+(13CV*13CV))/3CV
    'Calculate wind direction
    111CV(W)=R2D(ATAN(13CV/12CV))
    111CV(W)=111CV+((12CV>0)AND(13CV<0))*360
    111CV(W)=111CV+(12CV<0)*180
    'If wind speed is zero, return -1.0:
    111CV(W)=111CV-(110CV=0)*(111CV+1)

    'display values and store in schedule B
    110CV("1min_Mean Wind~m/s",FF0)
    111CV("1min_Mean Wind Direction",FF0)
    220CV("1min_Temperature ~Deg C",FF1)=120CV/3CV
    221CV("1min_Humidity ~%RH",FF0)=121CV/3CV
    222CV("1min_Pressure ~hPa",FF0)=122CV/3CV
    30CV("Rain Accumulation ~mm")
    31CV("Rain Duration ~s")

    111CV(W)=D2R(111CV) 'convert degrees to radians
    112CV(W)=112CV+110CV*COS(111CV) 'Sum x components
    113CV(W)=113CV+110CV*SIN(111CV) 'Sum y components
    320CV(W)=320CV+220CV 'sum temperature
    321CV(W)=321CV+221CV 'sum humidity
    322CV(W)=322CV+222CV 'sum pressure
    4CV(W)=4CV+1 'increment counter for calculating average
    3CV(W)=0 12..13CV(W)=0 120..122CV(W)=0 'reset sums and counter
    'reset rain counters if it is 9AM (540 minutes after midnight)
    IF(2ST(0)><540,541){XD}

'=====================================================
'     Schedule C
'   - Logs up to 5MB of data records, overwrites when full
'   - Runs every 1 hour
'=====================================================
RC(DATA:OV:5MB)1H
    'Calculate mean wind magnitude
    210CV(W)=SQRT((112CV*112CV)+(113CV*113CV))/4CV
    'Calculate wind direction
    211CV(W)=R2D(ATAN(113CV/112CV))
    211CV(W)=211CV+((112CV>0)AND(113CV<0))*360
    211CV(W)=211CV+(112CV<0)*180
    'If wind speed is zero, return -1.0
    211CV(W)=211CV-(210CV=0)*(211CV+1)

    'display values and store in schedule C
    210CV("1hr_Mean Wind~m/s",FF0)
    211CV("1hr_Mean Wind Direction",FF0)
    420CV("1hr_Temperature ~Deg C",FF1)=320CV/4CV
    421CV("1hr_Humidity ~%RH",FF0)=321CV/4CV
    422CV("1hr_Pressure ~hPa",FF0)=322CV/4CV

    4CV(W)=0 112..113CV(W)=0 320..322CV(W)=0

RDX 'schedule which resets the rain counters (manually executed)
    30CV("Max Rain ~mm")
    1SERIAL("\\e{0RU,Z=Y\\013\\010}",0.2,W) 'send reset to WXT520
    LOGON
END
```

### 3.5.2 SDI-12

The SDI-12 implementation is more straight forward and easy to read because the logger reads directly from the registers in the weather station. If you have a fast sampling program then it is recommended to use continuous mode, alternatively if you wish to lower the power consumption of the logger then use polled mode.

#### 3.5.2.1 Sample Program 3 – Basic SDI-12 Demonstration Program

```
BEGIN "WS_SDI12"
'=====================================================
'Initial Set Up
    'create reset function button
    FUNCTION1="Reset Counters"{SDI12SEND 5 "0RU,Z=Y!"}
    'activate SDI-12 continuous mode
    SDI12SEND 5 "0XU,M=R!"
    'manual reset command
    SDI12SEND 5 "0RU,Z=M!"
    LOGON
'=====================================================

'=====================================================
'     Schedule A
'   - Logs up to 3MB of data records, overwrites when full
'   - Runs every 5 seconds
'=====================================================
RA(DATA:OV:3MB)5S
    'read the registers of interest and display the values
     5SDI12(AD0,R102,CM,"Ave Wind Dir.~Deg")
     5SDI12(R105,CM,"Ave Wind Speed~m/s")
     5SDI12(R106,CM,"Max Wind Speed~m/s")
     5SDI12(R201,CM,"Air Temperature~Deg")
     5SDI12(R202,CM,"Rel Humidity~%RH")
     5SDI12(R203,CM,"Air Pressure~hPa")
     5SDI12(R301,CM,"Rain Accum.~mm")
     5SDI12(R304,CM,"Hail Accum.~hits/cm^2")
END
```

# 4 Appendices

## 4.1 Example serial communications

(→ = message from the D80, ← = response from the WXT520)

→SSDIRECT 1 “^M^J”
→0R1
←$WIXDR,A,283,D,0,A,283,D,1,A,283,D,2,S,0.0,M,0,S,0.1,M,1,S,0.1,M,2*5C
→0R2
←$WIXDR,C,23.9,C,0,H,42.5,P,0,P,1016.2,H,0*71
→0R3
←$WIXDR,V,0.00,M,0,Z,0,s,0,R,0.0,M,0,V,0.0,M,1,Z,0,s,1,R,0.0,M,1,R,0.0,M,2,R,0.0,M,3*60
→ENDSSDIRECT

## 4.2 0R1 return message format

**$WIXDR,A,283,D,0,A,283,D,1,A,283,D,2,S,0.0,M,0,S,0.1,M,1,S,0.1,M,2*5C**

**Where..**

$ -> Start of the message
WI -> Device Type (WI = weather instrument)
XDR -> Transducer measurement response identifier

A -> Transducer id 0 type (wind direction)
283 -> Transducer id 0 data (min wind direction)
D -> Transducer id 0 units (degrees, min wind direction)
0 -> Transducer id for min wind direction

A -> Transducer id 1 type (average wind direction)
283 -> Transducer id 1 data (average wind direction)
D -> Transducer id 1 units (degrees, average wind direction)
1 -> Transducer id for average wind direction

A -> Transducer id 2 type (wind direction)
283 -> Transducer id 2 data (max wind direction)
D -> Transducer id 2 units (degrees, max wind direction)
2 -> Transducer id for max wind direction

S -> Transducer id 0 type (wind speed)
0.0 -> Transducer id 0 data (min wind speed)
M -> Transducer id 0 units (m/s, min wind speed)
0 -> Transducer id for min wind speed

S -> Transducer id 1 type (wind speed)
0.1 -> Transducer id 1 data (average wind speed)
M -> Transducer id 1 units (m/s, average wind speed)
1 -> Transducer id for average wind speed

S -> Transducer id 2 type (wind speed)
0.1 -> Transducer id 2 data (max wind speed)
M -> Transducer id 2 units (m/s, max wind speed)
2 -> Transducer id for max wind speed

* -> Checksum delimiter
5C -> Two-character checksum for the response
<cr><lf> -> Response Terminator

## 4.3 0R2 return message format

**$WIXDR,C,23.9,C,0,H,42.5,P,0,P,1016.2,H,0*71**

**Where..**

$ -> Start of the message
WI -> Device Type (WI = weather instrument)
XDR -> Transducer measurement response identifier

C -> Transducer id 0 type (Temperature)
23.9 -> Transducer id 0 data (Temperature)
C -> Transducer id 0 units (C, Temperature)
0 -> Transducer id for Temperature

H -> Transducer id 0 type (Humidity)
42.5 -> Transducer id 0 data (Humidity)
P -> Transducer id 0 units (%, Humidity)
0 -> Transducer id for Humidity

P -> Transducer id 0 type (Pressure)
1016.2 -> Transducer id 0 data (Pressure)
H -> Transducer id 0 units (hPa, Pressure)
0 -> Transducer id for Pressure

* -> Checksum delimiter
71 -> Two-character checksum for the response
<cr><lf> -> Response Terminator

## 4.4 0R3 return message format

**$WIXDR,V,0.00,M,0,Z,0,s,0,R,0.0,M,0,V,0.0,M,1,Z,0,s,1,R,0.0,M,1,R,0.0,M,2,R,0.0,M,3*60**

**Where..**

$ -> Start of the message
WI -> Device Type (WI = weather instrument)
XDR -> Transducer measurement response identifier

V -> Transducer id 0 type (Accumulated rainfall)
0.00 -> Transducer id 0 data (Accumulated rainfall)
M -> Transducer id 0 units (in, Accumulated rainfall)
0 -> Transducer id for Accumulated rainfall

Z -> Transducer id 0 type (Rain duration)
0 -> Transducer id 0 data (Rain duration)
s -> Transducer id 0 units (s, Rain duration)
0 -> Transducer id for Rain duration

R -> Transducer id 0 type (Rain intensity)
0.0 -> Transducer id 0 data (Rain intensity)
M -> Transducer id 0 units (mm/h, Rain intensity)
0 -> Transducer id for Rain intensity

V -> Transducer id 1 type (Hail accumulation)
0.0 -> Transducer id 1 data (Hail accumulation)
M -> Transducer id 1 units (hits/cm2, Hail accumulation)
1 -> Transducer id for Hail accumulation

Z -> Transducer id 1 type (Hail duration)

0 -> Transducer id 1 data (Hail duration)
s -> Transducer id 1 units (s, Hail duration)
1 -> Transducer id for Hail duration

R -> Transducer id 1 type (Hail intensity)
0.0 -> Transducer id 1 data (Hail intensity)
M -> Transducer id 1 units (hits/cm2h, Hail intensity)
1 -> Transducer id for Hail intensity

R -> Transducer id 1 type (Rain peak intensity)
0.0 -> Transducer id 1 data (Rain peak intensity)
M -> Transducer id 1 units (mm/h, Rain peak intensity)
2 -> Transducer id for Rain peak intensity

R -> Transducer id 1 type (Hail peak intensity)
0.0 -> Transducer id 1 data (Hail peak intensity)
M -> Transducer id 1 units (hits/cm2h, Hail peak intensity)
3 -> Transducer id for Hail peak intensity

* -> Checksum delimiter
60 -> Two-character checksum for the response


©2010 Thermo Fisher Scientific Australia Pty Ltd. All rights reserved. A.B.N. 52 058 390 917