RS232 Communication protocol for AAG_CloudWatcher

Communication is always established by requesting information from device.

Request information

The request commands consist of 1 character followed by ! (exclamation mark) which is interpreted by the AAG_CloudWatcher device as end-of-command. Note that the only exception is the command that sets the Pulse Width modulation value.

Command	Description	
A!	Get internal name	
B!	Get firmware version	
C!	Get values	
D!	Get internal errors	
E!	Get rain frequency	
F!	Get switch status	
G!	Set switch – open	
H!	Set switch – close	
Pxxxx!	Set PWM Value xxxx = value of PWM (always 4 characters)	Exception in command length
Q!	Get PWM value	
S!	Get Sky IR Temp	
T!	Get Sensor Temp	
z!	Reset RS232 buffer pointers	

Received Information

The length of the information structure returned by the AAG_CloudWatcher device is always a multiple of 15 characters string as follows:

!XXyyyyyyyyyyyy

Characters	Description	Length
!	Begin of information	1 byte
XX	Information Nature	2 bytes
yyyyyyyyyyyy	Information content related to the Nature	12 bytes

NB: The quotes shown in 1st column of the table below are not present in the information returned by AAG_CloudWatcher. They are shown to delimit the 2 characters and show that the 2nd character is in most cases a space.

Meaning of XX	Description of yyyyyyyyyyy
"1 "	Infra red temperature in hundredth of degree Celsius
"2 "	Infra red sensor temperature in hundredth of degree Celsius
"3 "	Analog0 output 0-1023 => 0 to full voltage (Ambient Temp NTC)
"4 "	Analog2 output 0-1023 => 0 to full voltage (LDR ambient light)
"5 "	Analog3 output 0-1023 => 0 to full voltage (Rain Sensor Temp NTC)
"6 "	Analog3 output 0-1023 => 0 to full voltage (Zener Voltage reference)
"E1"	Number of internal errors reading infra red sensor: 1st address byte
"E2"	Number of internal errors reading infra red sensor: command byte
"E3"	Number of internal errors reading infra red sensor: 2nd address byte
"E4"	Number of internal errors reading infra red sensor: PEC byte NB: the error counters are reset after being read.
"N "	Internal Name
"V "	Firmware Version number
"Q "	PWM duty cycle
"R "	Rain frequency counter
"X "	Switch Opened
"Y "	Switch Closed
chr(XON) +chr(32)	Handshaking XON (= 0x11) (= 17)

Detailed Description of Received Information

The AAG_CloudWatcher always returns information in blocks of 15 characters. The meaning of the 15 character blocks are described in the above table.

However different commands will return different number of blocks as described in the following table:

Sent	Received			
Command	No of Blocks	Total no. of characters	Block Content	Meaning
A!	2	30	1st block: !N CloudWatcher 2nd block: !¶ 0	1st block: Internal name 2nd block: Handshaking block
B!	2	30	1st block: !N 1.10 2nd block: !¶ 0	1st block: Firmware version 2nd block: Handshaking block
C!	5	75	1st block: !6 xxxx 2nd block: !3 xxxx 3rd block: !4 xxxx 4th block: !5 xxxx 5th block: !¶ 0	1st block: xxxx Zener voltage 2nd block: xxxx Ambient Temperature 3rd block: xxxx LDR voltage 4th block: xxxx Rain Sensor Temperature 5th block: Handshaking block
D!	5	75	1st block: !E1 nnnnn 2nd block: !E2 nnnnn 3rd block: !E3 nnnnn 4th block: !E4 nnnnn 5th block: !¶ 0	1st block: nnnnn 1st address byte errors 2nd block: nnnnn Command byte errors 3rd block: nnnnn 2nd address byte errors 4th block: nnnnn PEC byte errors 5th block: Handshaking block

E!	2	30	1st block: !R xxxx 2nd block: !¶ 0	1st block: xxxx Rain frequency 2nd block: Handshaking block
F!	2	30	1st block: !X 2nd block: !¶ 0 or 1st block: !Y 2nd block: !¶ 0	1st block: Switch Open 2nd block: Handshaking block or 1st block: Switch Close 2nd block: Handshaking block
G!	2	30	1st block: !X 2nd block: !¶ 0	1st block: Switch Open 2nd block: Handshaking block
H!	2	30	1st block: !Y 2nd block: !¶ 0	1st block: Switch Close 2nd block: Handshaking block
Ppppp!	2	30	1st block: !Q pppp 2nd block: !¶ 0	1st block: PWM duty cycle 2nd block: Handshaking block
Q!	2	30	1st block: !Q pppp 2nd block: !¶ 0	1st block: pppp PWM duty cycle 2nd block: Handshaking block
S!	2	30	1st block: !1 ttttt 2nd block: !¶ 0	1st block: ttttt IR sky temperature 2nd block: Handshaking block
T!	2	30	1st block: !2 ttttt 2nd block: !¶ 0	1st block: ttttt IR sensor temperature 2nd block: Handshaking block
z!	1	15	1st block: !¶ 0	1st block: Handshaking block

NOTE: xxxx represents a value between 0 and 1023 which relates to the 10 bit microprocessor D/A converter;
NOTE: nnnnn represents a positive integer;
NOTE: ttttt represents a positive integer which represents the temperature in hundredth of degree Celsius, e.g. ttttt = 2456 => 24.56 ºC;
NOTE: pppp represents a value between 0 and 1023 which relates to the 10 bit microprocessor PWM modulator;

NOTE: The handshaking block represented in the above table as "!¶ 0" consists of 15 characters:

Pos	Content
1	Exclamation mark = char 0x21 (hexadecimal) or 33 (decimal)
2	XON = char 0x11 (hexadecimal) or 17 (decimal)
3	Space = char 0x20 (hexadecimal) or 32 (decimal)
4	Space = char 0x20 (hexadecimal) or 32 (decimal)
5	Space = char 0x20 (hexadecimal) or 32 (decimal)
6	Space = char 0x20 (hexadecimal) or 32 (decimal)
7	Space = char 0x20 (hexadecimal) or 32 (decimal)
8	Space = char 0x20 (hexadecimal) or 32 (decimal)
9	Space = char 0x20 (hexadecimal) or 32 (decimal)
10	Space = char 0x20 (hexadecimal) or 32 (decimal)
11	Space = char 0x20 (hexadecimal) or 32 (decimal)
12	Space = char 0x20 (hexadecimal) or 32 (decimal)
13	Space = char 0x20 (hexadecimal) or 32 (decimal)
14	Space = char 0x20 (hexadecimal) or 32 (decimal)
15	ZERO = char 0x30 (hexadecimal) or 48 (decimal)

Communication operational recommendations

The device firmware was developed using MikroBasic. The RS232 communications make use of microprocessor interrupts. On the other hand the rain frequency is measured using an internal counter and an interrupt microprocessor line.

To prevent a mix up in the microprocessor interrupts, I strongly suggest the following:

1. When communicating with the device send one command at a time and wait for the respective reply, checking that the correct number of characters has been received;
2. Perform more than one single reading (say, 5) and apply a statistical analysis to the values to exclude any outlier. I am using 5 readings and calculate the average value (AVG) and standard deviation (STD). Any values that are outside the range ***AVG-STD*** and ***AVG+STD*** are excluded. The final value is the average of the values which were not excluded;
3. The rain frequency measurement is the one that takes more time - 280 ms approximately;
4. The following reading cycle takes just less than 3 seconds to perform;

Perform 5 times	
get IR temperature	command “S!”
get Ambient temperature	command “T!”
get Values	command “C!”
get Rain Frequency	command “E!”
loop	
get PWM value	command “Q!”
get IR errors	command “D!”
get SWITCH Status	command “F!”

5. The Visual Basic 6 main program makes use of the RS232 control event to handle the device replies, thus avoiding the program to wait for the end of the above cycle.
6. The algorithm that controls the heating cycles of the rain sensor is also programmed in the Visual Basic 6 main program and not in the device microprocessor.

Converting values sent by the device to meaningful units

1. The infrared temperature is converted to ºC by dividing it by 100;
2. The temperature of the infrared sensor is converted to ºC by dividing it by 100;
3. Pulse Width values are converted to % by

   PWM = 100 * xxxx / 1023

   where
   xxxx = value sent by the device
   PWM = pulse width as %

4. The internal supply voltage value is calculated from the Zener voltage value converted to volts by:

   Vs = 1023 * ZenerConstant / xxxx

   where
   xxxx = value sent by the device
   Vs = internal supply voltage in V
   ZenerConstant = 3

5. The ambient temperature is calculated in ºC by

   ```
   If xxxx > 1022 Then xxxx = 1022
   If xxxx < 1 Then xxxx = 1
   r = AmbPullUpResistance / ((1023 / xxxx) - 1)      'resistance K ohms
   r = Log(r / AmbResAt25)
   TAmb = 1 / (r / AmbBeta + 1 / (ABSZERO + 25)) - ABSZERO
   ```

   where
   Log = The log function is the natural log (usually Ln)
   xxxx = value sent by the device
   TAmb = ambient temperature in ºC
   AmbPullUpResistance = 9.9
   AmbResAt25 = 10
   AmbBeta = 3811
   ABSZERO = 273.15
   xxxx corresponds to the value sent by the device

6. LDR value is calculated in K ohms by

   ```
   If xxxx > 1022 Then xxxx = 1022
   If xxxx < 1 Then xxxx = 1
   LDR = LDRPullupResistance / ((1023 / xxxx) - 1)      'resistance K ohms
   ```

   where
   xxxx = value sent by the device
   LDR = LDR resistance in K ohms
   LDRPullupResistance = 56

7. Rain sensor temperature is calculated in ºC by

```
If xxxx > 1022 Then xxxx = 1022
If xxxx < 1 Then xxxx = 1
r = RainPullUpResistance / ((1023 / xxxx) - 1)     'resistance K ohms
r = Log(r / RainResAt25)
TRain = 1 / (r / RainBeta + 1 / (ABSZERO + 25)) – ABSZERO
```

where
Log = The log function is the natural log (usually Ln)
xxxx = value sent by the device
TRain = rain sensor temperature in ºC
RainPullUpResistance = 1
RainResAt25 = 1
RainBeta = 3450
ABSZERO = 273.15

8. Rain frequency is equal to the value sent by the device.