Tracking System Support

Version 1.31 (09 Jul 2015)
Firmware 2.71 (09 Jul 2015)
Status Under development
Author Paul Reuvers

Contents

Document history

0.02	12 Feb 2008	Initial version for testing	1.20
1.10	05 Oct 2010	Minor changes	1.41
1.20	07 Jun 2014	Commands B and H added	2.54
1.30	07 Jul 2015	Reorganised, command C and R added	2.70
1.31	09 Jul 2015	Minor changes to parameters. Corrections. Remote control.	2.71

1. Introduction

W*eed*-IT is a system for selective weed control, using patented methods for the recognition and detection of weed, based on the fluorescent properties of the chlorophyl that is present in the leaves of living plants. It is a stand-alone system, that detects weed and sprays a herbicide solution exactly on top of the selected plant, resulting in a very low environmental strain.

Some customers have expressed the wish to monitor the activity of their W*eed*-IT machines. Although the W*eed*-IT itself maintains many parameters about the processed area and the amount of herbicide used, this does not put the data in an historical and geographical context. W*eed*-IT therefore supports the connection of an external (optional) tracking system that allows actual data to be sent to a remote database, e.g. via a GSM link. The data can then be processed further at the remote end, and specific user views can be obtained from this.

In order to allow an external tracking system to be connected to the W*eed*-IT, the data-exchange between both systems has to follow a certain protocol. The protocol supported by the W*eed*-IT is described in *this* document.

1.1 Serial Port

The W*eed*-IT Console is equipped with a serial RS232 port especially for diagnostics and expansion purposes. From version 1.14 onwards, support for a tracking system has been added to the firmware. The serial port is available on an internal connector inside the console and can be brought out when desired. It is a fully buffered RS232 port, with proper RS232 levels and handshaking, although the latter is disabled in the firmware.

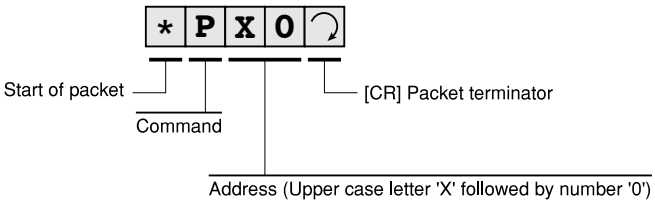
For the connection to the external tracking system, only three lines are needed: TX, RX and Ground. The configuration of the RS232 port is as follows:

38400 baud, 8N1, no handshake (8 data-bits, no parity, 1 stop-bit)

1.2 Protocol

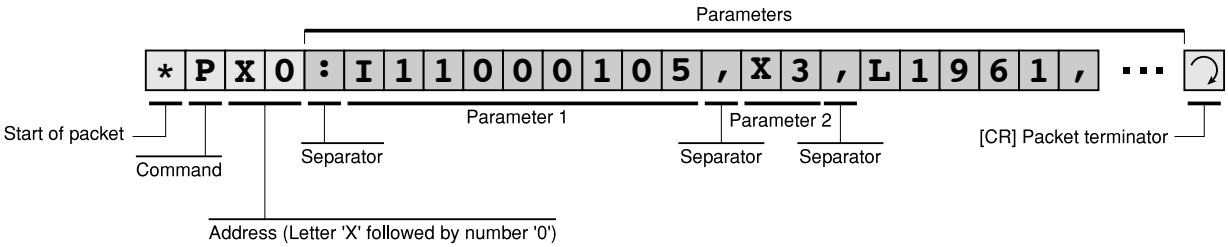
In order for two systems to communicate with each other, they both have to obey to a given protocol. The protocol supported by the Weed-IT is specified in this document. It works on the basis of polling, which means that the tracking system has to place a polling request with the Weed-IT. The Weed-IT will then reply by returning a status message. The tracking system should not issue another Poll request before the reply to the previous request has been received, or a timeout has occured.

All requests, commands and replies, consist of printable characters (i.e. no control characters and no top-bit set characters) in a human readable string. The string always starts with an asterisk (*) and is terminated by a *Carriage Return* **<cr>**, also known as **ASCII 13** or **0x0D**. The poll request has the following format:



The first character of the poll request string should always be an asterisk (*). The second position contains the command; in this case 'P' for Poll. The next two positions are fixed and should always contain 'X0' (the letter 'X' followed by the number '0'). The string should be terminated by a **<cr>**.

The Weed-IT will reply to the Poll request ***PX0<cr>** by returning a status string of variable length. This reply has a fixed start ('PX0'), a separator (':') followed by a number of comma-separated variables, and finally a **<cr>** as terminator. The returned string has the following format:



The string between the (:) and the <cr> contains a series of parameters, separated by commas. Each parameter consists of a name (single letter) followed by a numerical decimal value. For example, in the diagram above, the first parameter is '`I11000105`', which means that the serial number of the Weed-IT (`I` = ID) is `11000105`.

Please note that the length of each field (i.e. each parameter) is of variable length and is in decimal format. The values do not contain a decimal dot or comma. Also note that additional parameters may be added in a future release of the software. Furthermore, the order of the parameters may be changed in a future release as well. The tracking software should be able to handle this. The entire string has a maximum length of 128 characters, including the terminator.

2. Commands

*P	Poll	Read variable parameters
*C	Config	Read fixed parameters
*B	Bin	Read nozzle activity in binary format (no checksum)
*H	Hex	Read Nozzle activity in Hex format (no checksum)
*R	Reset	Reset specified parameter
*Z	HOLD	Put the machine on HOLD
*G	GO	Put the machine in Driving mode

2.1 Polling commands

Command P *P0<cr>

This command is used to read the continuously changing values from the WEEDit system. This command should be issued regularly by the host system in order to get a near realtime indication of the current status of the system. As the protocol is based on polling, the P-command is used for 'polling'.

Syntax

```
*PX0<cr>
```

Reply

```
*PX0:I310001,X3,E0,F00000000,L3600 ... C64<cr><lf>
```

The following parameters are currently defined:

I	Identification
X	Status [2]
E	Error status [4]
F	Warning Flags (32-bit value in Hex8 notation) [4]
L	Liquid used since last reset (herbicide solution) [ml]
H	Herbicide used (pure) since last reset [ml]
U	Current herbicide Usage [ml/Ha]
A	Area processed since last reset x 1000 [m²]
T	Time since last reset [min]
V	Velocity [mm/s]
D	Distance [m]
P	Pressure [mbar]
Y	Sunlight factor [4]
W	Width (vehicle working width) [mm] [3]
Q	Savings [0-100%]
C	Checksum [1]

Remark

1. Note that if a checksum is supplied (C), it will always be the last parameter on the line. Calculation of the checksum is explained under the heading 'Checksum' on one of the next pages.
2. During normal operation a value of '3' should be returned. When the Weed-IT is temporary put on hold, e.g. when crossing a street or a field, a value of '6' will be returned. The other values are unlikely to be encountered. Note that State 2 will only be seen when the device has an internal error, such as a memory device not working.
3. Parameter W (Width) has been removed in version 2.70 of the firmware as it is a system constant rather than a variable value. Systems constants may now be read using the 'C' command (*CX0).
4. Available from version 2.70 onwards.

A typical string may look like this:

```
*PX0:I11000105,X3,L290565,H5811,S58,U5941,A8906400,T30,V4090,D3107,P1989,C28<cr>
```

I	ID = 11000105
X	Status = 3
L	Liquid = 290565 [ml] = 290.56 [l]
H	Herbicide = 5811 [ml] = 5.81 [l]
S	Solution = 58 = 5.8%
U	Usage = 5941 [ml/Ha] = 5.94 [l/Ha]
A	Area = 8906.400 [m²]
T	Time = 30 [min]
V	Velocity 4090 [mm/s] = 4.09 [m/s]
D	Distance = 3107 [m]
P	Pressure = 1989 [mbar] = 1.98 [bar]
C	Checksum = 28

X - Status

The Status parameter (X) can take the following values:

0	Initialising
1	Idle
2	Error
3	**Running**
4	PowerDown
5	Maintenance
6	Hold
7	Sleep
8	Flushing

E - Error

The Error parameter (E) can take the following values. Note that in a remote controlled application, any value of 'E' other than '0' should stop the vehicle immediately. This type of error is unrecoverable.

0	**No error**	
1	Power	Power supply failure
2	Overheat	Console is overheated
3	Solenoid	Too many solenoids failing
4	I2C	Internal I2C bus error
5	EEPROM	Internal EEPROM error
6	Slave	Internal error reported by a slave
7	MaxSlaves	Too many slaves connected
8	SPI	Internal SPI bus error

F - Warnings

All system warnings are present as bit-flags in the flags parameter (F). Note that some of these flags are self-clearing, which means that the system can clear the flag automatically when the conditions for the warning are no longer valid. All other warnings are permanent. In a remote controlled application, it is up to that application to determine for what type of warnings the vehicle should be stopped.

Bit	Code	Description	Mask (hex)
0	Error	Serious error	00000001
1	Wald	PSU not found	00000002
2	PwrOff	Forced power down	00000004
3	Battery	Low battery	00000008
4	HiBatt	Battery voltage too high	00000010
5	Comms	Slave lost (communications error)	00000020
6	Sync	Sync failure	00000040
7	WaldComms	Communications error when talking to PSU	00000080
8	MaxSlaves	Too many slaves connected	00000100
9	MySerial	Console serial number invalid	00000200
10	Firmware	Slave firmware version out of date	00000400
11	Serial	Slave has reported invalid serial number	00000800
12	EEPROM	EEPROM needs configuring	00001000
13	LoPress	Pressure too low	00002000
14	HiPress	Pressure too high	00004000
15	HALL	HALL speed sensor missing	00008000
16	SlaveOrder	Slave order has changed	00010000
17	SlaveNew	A new Slave has been detected	00020000
18	SlaveMissing	A Slave has been lost	00040000
19	NoSlaves	No Slaves found on startup	00080000
20	WaldTemp	PSU temperature too high	00100000
21	LoMisc	Misc sensor value too low	00200000
22	HiMisc	Misc sensor value too high	00400000
23	PowerCut	Power has been lost enexpectedly	00800000

The following bit flags are self-clearing snd should not be considered as warnings or errors:

24	Hold	System is on HOLD	01000000
25	Reverse	Vehicle is driving in reverse	02000000
26	Manual	Manual mode	04000000
27	Flush	Flushing the nozzles	08000000
28	Simulation	Speed simulation mode (fixed speed)	10000000
29	*reserved*	-	20000000
30	*reserved*	-	40000000
31	*reserved*	-	80000000

Available from version 2.70 onwards

Whilst the command *PX0 is used for polling the system regularly for continuously changing values, command *CX0 can be used to read the system constants. Typically, these constants should be read once by the host system. As some of these constants can be altered manually by the user, it may be wise to re-read the values at regular intervals.

Syntax

```
*CX0<cr>
```

Reply

```
*CX0:I310001,X3,E0, ... C32<cr>
```

Please note that the length of each field (i.e. each parameter) is of variable length and is in decimal format. The values do not contain a decimal dot or comma. Also note that additional parameters may be added in a future release of the software. Furthermore, the order of the parameters may be changed in a future release as well. The tracking software should be able to handle this. The entire string has a maximum length of 128 characters, including the terminator. The following parameters are currently defined (case sensitive):

I	ID	See *PX0 command
W	Vehicle width [mm]	
S	Solution (Dose) in [% x 10]	e.g.: 58 = 5.8%
M	PWM Mode (see below)	
R	Rate (Coverage) [l/Ha]	
B	Bias [0-50%]	
N	Nozzle type (see below)	
G	Sensitivity [1-4]	
m	Margin [mm]	Dependent on current Wind setting
w	Wind [1-4]	
C	Checksum	Last item (see chapter 'Checksum')

M - Mode

0	Normal (PWM off)
1	PWM Rate mode
2	Bias mode

N - Nozzle type

This parameter returns the full name of the nozzle as a string, preceeded by a colon (:). The name is terminated either by a field separator (i.e. a comma) or the end-of-line character (i.e. CR), e.g.:

```
N:TP30-04E
```

Available from version 2.70 onwards

This command can be used to reset certain paramaters or counters in the WEEDit system. Currently only one parameter can be reset, but further variables may be added in a future release.

Syntax

`*RX0:<par><cr>`

Example

`*RX0:1<cr>` Clear warnings

Reply

`*RX0:<par><cr><lf>`

Parameter

The number following the colon (:) determines which variable should be reset:

1 Warnings

Warnings

Resetting the Warnings will cause the system to 'forget' about any outstanding problems. If a problem is persistent though, the system will re-issue the ralated warning near intantly.

2.1 Activity monitoring

In order to give an external system an indication of the activity that takes place on each channel of the WEEDit system, two commands are available. With these commands, it can be checked whether a channel has been active since the last time it was checked. It does not give any information about the amount of activity of that channel; just whether or not there was any activity.

Channels

A WEEDit system consists of a number of sensor units mounted on one or two booms. Each sensor has 5 channels and scans lane with a width of 20 cm, or 1 m per sensor. For example: a 36 metre wide WEEDit installation consists of 36 sensors (18 on the left boom and 18 on the right boom), with a total of 180 channels.

Command B *BX0

Available from version 2.54 onwards

This command returns the activity of all channels in ASCII binary format. Each sensor is respresented by 5 characters; one for each channel. The characters can be '0' (no activity) or '1' (activity). The command is "BX0" followed by a cariage return (CR).

The command replies with the same header, followed by a colon (:), followed by a string of '0' and '1' characters, representing the channels from left to right (as seen from the driver's seat). The length of the string depends on the number of sensors connected to the system. The string is terminated with a carriage return <cr> and a linefeed <lf>.

Syntax

```
*BX0<cr>
```

Reply

```
*BX0:101100000011111....01101<cr><lf>
```

Command H *HX0

Available from version 2.54 onwards

This command is similar to the one above (B), but returns the channels activity of each sensor in a single byte. The byte is presented in the string in ASCII Hex format, with the least significant bit (i.e. bit 0) representing theleftmost channel of the sensor. As each sensor has 5 channels, only the lowest 5 bits of each byte are used. This means that the returned value will always be between 00 and 1F.

Syntax

```
*HX0<cr>
```

Reply

```
*HX0:1F00001F0011...00<cr><lf>
*HX0:1F<cr><lf>
```

2.2 Remote control commands

A few commands have been added to allow limited remote control. At present, this allows the unit to be put on HOLD and subsequently release it from HOLD. Note that if the unit has been put on HOLD manually (i.e. from the console) it has to be released manually first. This is done for safety reasons.

Command Z `*HX0:<action>`

Available from version 2.71 onwards

This command can be used to remotely put the system on hold. It is similar to pressing the HOLD button on the WEEDit console. The system replies with a *ZX0 packet:

Syntax

`*ZX0:<action><cr>`

Reply

`*ZX0:<state><cr><lf>`

Examples

*ZX0:1	Command issued successfully
*ZX0:0	Command ignored (vehicle already on HOLD)

Action

1	HOLD	Put vehicle on HOLD
999	ShutDown	Turn OFF the system completely (not not be turned on again remotely)

Command G `*GX0:1`

Available from version 2.71 onwards

This command can be used to put the vehicle in driving mode (GO) after an earlier HOLD command (Z). Please note that the GO command will only be accepted if the vehicle was put on HOLD by the remote host itself. If the vehicle was put on hold manually (i.e. by pressing the HOLD button on the console), is has to be released manually first. This is done for safety reasons.

Also note that this command requires the parameter '1' as part of the command. The host application should check the returned state value (see below) to see if the command was executed successfully. If the returned state is '0', the command was ignored for safety reasons.

Syntax

`*GX0:1<cr>`

Reply

`*GX0:<state><cr><lf>`

Examples

`*GX:1<cr><lf>`	Command accepted, vehicle now in running mode (GO)
`*GX:0<cr><lf>`	Command ignored, vehicle still on HOLD

3. Timeout

Under normal circumstances (i.e. when the unit is in operation) the W*eed*-IT will reply immediately to a poll request from the tracking system. When the unit is turned off however, it will **not** reply to a poll request. The tracking system should therefore be aware of this situation. If the Weed-IT has not replied to a request within one second (1 sec), it should be considered *inactive*.

4. Checksum

The last parameter of the string contains the checksum that can be used to check the string for transmission errors. There are many ways in which a checksum can be calculated. For this application we've chosen a simple Exclusive-OR scheme (EOR). It works as follows:

Each character of the string is EOR-ed with the next one, up to (but not including) the comma preceeding the 'C'-parameter. The actual checksum parameter is of course not included in the calculation, nor is the 'C' and the preceeding comma. The prefix (**`*PX0:`**) **is** included in the string. Take for example this string:

```
*PX0:I11000105,X3,L290565,H5811,U5941,A8906400,T30,V4090,D3107,P1989,C28<cr>
```

The part used for the EOR calculation is then:

```
*PX0:I11000105,X3,L290565,H5811,U5941,A8906400,T30,V4090,D3107,P1989
```

A simple example in the C Programming Language would be something like this:

```
int get_checksum (char *s) {
  int checksum = 0;
  while (*s) {
    checksum ^= *s++;
  }
  return(checksum);
}
```

For the string in the above example, this would yield a value of 28.