FDC1 12-42V DC
COMMUNICATION PROTOCOL v2

HARDWARE

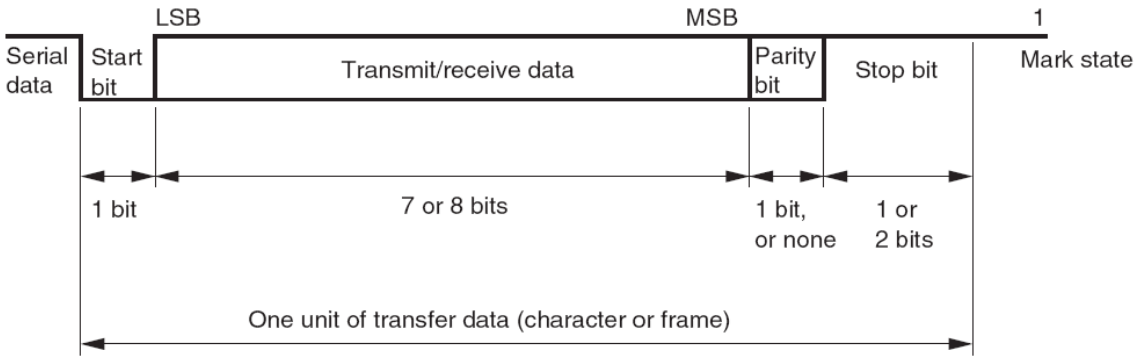
Wiring:

RJ-25 phone connector	1	RxD (FDC1 input / external controller output)
	2	IN1 input
	3	GND
	4	+5V / 25mA (Slave mode) / Led output (other modes)
	5	IN2 input
	6	TxD (FDC1 output / external controller input)

TTL levels (0-5V)

Universal Asynchronous Receiver/Transmitter (UART)

1200 baud
1 start bit (0V)
8 bits of data (0V=0 / 5V=1)
no parity
1 stop bit (5V)



OUTPUT

Every 0’5 s, one of these 6-byte sequences is sent:

If the motor is OFF		If the motor is ON
27	1	27
76	2	76
0	3	SP1 = (motor_speed) / 256
A=alarm code	4	SP2 = (motor_speed) mod 256
TR1=(time_remaining*122) / 256	5	INT1 = (measured_current*3160) / 256
TR2=(time_remaining*122) mod 256	6	INT2 = (measured_current*3160) mod 256
checksum_odd	7	checksum_odd
checksum_even	8	checksum_even

- Since the motor speed will be greater than 256 whenever it’s ON, the 3rd byte helps determine which of the two sequences is sent.

- Alarm codes:

16	Battery out of limits
32…47	Fan output overload
48…63	Motor failed to start
64…79	Motor overload
80	Overtemperature
160…175	Fan output overload (end of retries)
176…191	Motor failed to start (end of retries)
192…207	Motor overload (end of retries)
208	Overtemperature (end of retries)
240…255	Internal error

- To obtain the time in seconds remaining until the next motor start-up:
  (TR1 * 256 + TR2) / 122
- To obtain the actual motor speed in rpm:
  SP1 * 256 + SP2
- To obtain the measured consumption of the motor in amperes:
  (INT1 * 256 + INT2) / 3160

- The checksum values are obtained from the other bytes by means of a xor funcion:

checksum_odd = byte_1 XOR byte_3 XOR byte_5
checksum_even = byte_2 XOR byte_4 XOR byte_6

INPUT

Programming mode + parameters

To program the device, one of the next 22-byte sequences must be received.

	Standard battery system	Special battery system
1	72	72
2	80	80
3	minimum_speed / 256	minimum_speed / 256
4	minimum_speed mod 256	minimum_speed mod 256
5	maximum_speed / 256	maximum_speed / 256
6	maximum_speed mod 256	maximum_speed mod 256
7	0	0
8	mode (bit 5=0)	mode (bit 5=1)
9	(1187 * cut_out_12V) / 256	(1187 * cut_out_special) / 256
10	(1187 * cut_out_12V) mod 256	(1187 * cut_out_special) mod 256
11	(1187 * cut_in_12V) / 256	(1187 * cut_in_special) / 256
12	(1187 * cut_in_12V) mod 256	(1187 * cut_in_special) mod 256
13	(1187 * cut_out_24V) / 256	255
14	(1187 * cut_out_24V) mod 256	255
15	(1187 * cut_in_24V) / 256	255
16	(1187 * cut_in_24V) mod 256	255
17	(1187 * cut_out_42V) / 256	255
18	(1187 * cut_out_42V) mod 256	255
19	(1187 * cut_in_42V) / 256	(1187 * nominal_voltage) / 256
20	(1187 * cut_in_42V) mod 256	(1187 * nominal_voltage) mod 256
21	checksum_odd	checksum_odd
22	checksum_even	checksum_even

- The whole sequence must be received within 1 second. Otherwise, the command will be ignored.
- These parameters are stored in non volatile memory.
- The speeds are expressed in rpm and should be between 1500 and 3500 rpm.
- The mode byte is the sum of the next members:

<table>
<tr><td rowspan="2">0: no thermostat delay

128: 3 minutes delay</td><td rowspan="2">+</td><td rowspan="2">0: internal battery limits

16: external battery limits (using in1 & in2)

32: special battery system (from 9 to 46V)</td><td rowspan="2">+</td><td colspan="3">2: external thermostat = SLAVE MODE</td></tr>
<tr><td>0: normal (speed selected using IN1 & IN2)
4: Smart Speed (speed selected using IN1 & IN2)
8: Sleep mode (speed selected using IN2)
12: Energy Saving Sleep mode (speed selected using IN2)</td><td>+</td><td>0: normal

3: speed proportional to battery voltage</td></tr>
</table>

- The cut-out and cut-in levels for each of the battery systems (12V, 24V, 42V or special) are expressed in volts and must fulfill the next requirements:

  STANDARD:
  cut_out < cut-in for each pair of values
  cut_out_12V > 9V
  cut_in_12V < 17V
  cut_out_24V > 17V
  cut_in_24V < 33V
  cut_out_42V > 33V
  cut_in_42V < 46V

  SPECIAL:
  cut_out_special < cut_in_special < nominal_voltage
  cut_out_special > 9V
  nominal_voltage < 46V

  Otherwise, operation is not guaranteed.

- The checksum values are obtained from the other bytes by means of a xor funcion:

checksum_odd = byte_1 XOR byte_3 XOR byte_5 XOR … XOR byte_19
checksum_even = byte_2 XOR byte_4 XOR byte_6 XOR … XOR byte_20

- If the checksum values are not correct, the parameters will be ignored by the device.

- When the parameters are acknowledged and saved to the memory, the device will return the same sequence of bytes to the programmer, except for the first two bytes (27 and 80).

Programming speed

To change the programmed speed, the next 8-byte sequence must be received.

1	72
2	115
3	minimum_speed / 256
4	minimum_speed mod 256
5	maximum_speed / 256
6	maximum_speed mod 256
7	85
8	checksum

- The checksum value is obtained from the other bytes by means of a xor funcion:

checksum = byte_1 XOR byte_2 XOR byte_3 XOR … XOR byte_7

- The speeds are expressed in rpm and should be within 1500 and 3500 rpm.
- The speeds programmed this way are not stored in non volatile memory, so these values will be reset when the power supply is cut off.
- SLAVE MODE: In this mode, the maximum speed is irrelevant, because the minimum speed will become the actual speed.
- SLAVE MODE: In order to stop the motor, a minimum speed equal to 0 should be programmed. The thermostat input (T+) should remain connected to T-.
- SLAVE MODE: The speed must be refreshed to check that the communication is OK, so the motor will stop if no speed is received within 60s.