theremino system

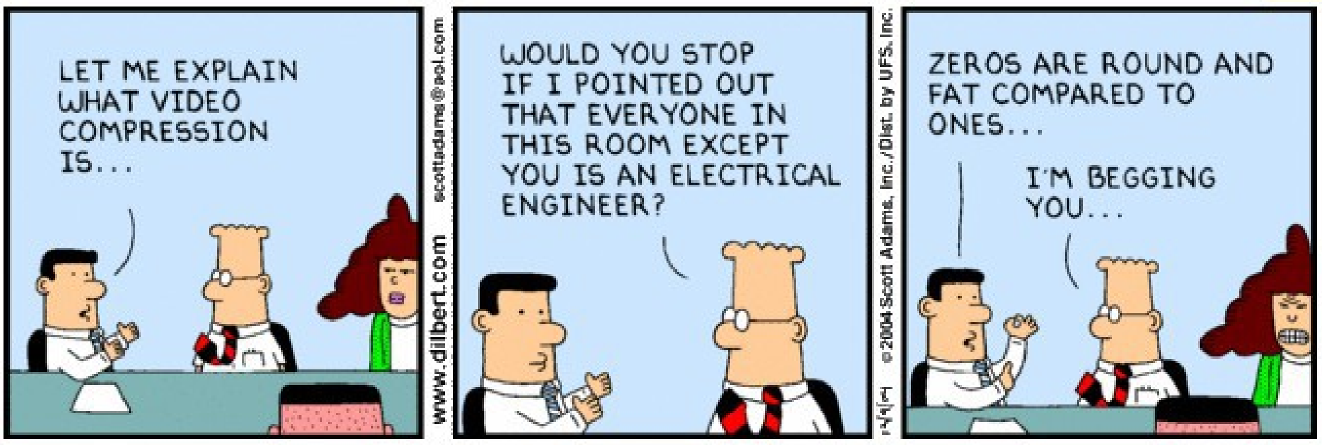
DPM 3.0

Dilbertian Protocol Modified

Serial communication protocol DPM
(Dilbertian Protocol Modified)

The protocol name

The name “Dilbertian” comes from the first version of this protocol, we called “IDP Inverted Dilbertian Protocol” (2010), where zeros were represented with a cell “thin” and with a cell “large” (“0” skinny and “1” fat – the inverse of the following strip from “Dilbert” by Scott Adams –www.dilbert.com).


LET ME EXPLAIN WHAT VIDEO COMPRESSION IS...
WOULD YOU STOP IF I POINTED OUT THAT EVERYONE IN THIS ROOM EXCEPT YOU IS AN ELECTRICAL ENGINEER?
ZEROS ARE ROUND AND FAT COMPARED TO ONES...
I'M BEGGING YOU...
scottadams@aol.com
www.dilbert.com
©2004 Scott Adams, Inc./Dist. by UFS, Inc.

Why a new Protocol?

Theremino system aims to provide a simple, cheap and easy to use, Input-Output System for PC. To make it easy to use auto-numbering and the recognition of the types are needed, to simplify and minimize the connections, it is necessary to communicate on a single wire.

Because there is a protocol with the necessary features we had to write it. The current version collects the best of many years of experimentation and research.

Features

•Bidirectional serial communication on a single wire
•Auto-configuration
•Auto detection of connected devices.
•Number of devices ranging from 1 at 200
•Number of bytes transmitted and received varies depending on the type of device
•Transmission speed up to 4 Megabits/sec to transmit a lot of data with short cables
•Transmission speed up to 100 kbits/sec for long cables (up to 10 Km)
•High efficiency of communication (from 10 at 20 times greater than in CAN)

Electrical aspects

The transmission takes place on a single wire, but you need a reference mass and a supply voltage, so the wires are normally 3.

Any device that supports DPM must have an input connector (to the Master) and one output to downstream devices (Slaves)

The connectors normally are 3 Pin connectors, distant 2.54 mm.

On the communication chain should have a Master (providing power and timing) and a number of devices connected in cascade (called Slave).

The tension that the Master provides on line is 5 V, with the same tolerance and with the same maximum current of the 5 V that withdrawing from the USB connector.

The maximum current that the Master can provide limits the number and type of devices that can be connected. This current, that is normally 250 mA, could be increased up to 500 mA, going over it would exceed the capacity of the USB and it would impose unusual features to the connectors.

The maximum number of connectable devices is limited by the following factors:
– The maximum current that the “master” can provide
– The maximum number of bytes that the line bears in a cycle time

Both the “Master” that the “Slave” are connected to the line with a damping resistor which also acts as a low pass filter against radio disturbance and how surge protection. The value of this resistor is normally from 33 at 330 ohm

Transmission format

Using a Non-Return-to-Zero format (NRZ) standard.

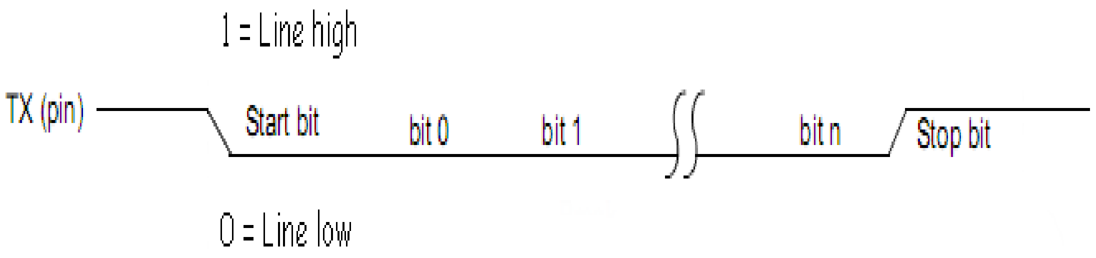
Default format = 8, N, 1
– 1 start bit
– 8 data bits
– no-parity
– 1 stop bit

Voltage levels on the data line "Com. Line"

Normally the line has a high level (3.3 Volt)

The signal "1" is denoted by 3.3 Volt

The signal "0" is denoted by zero Volt



Repeat times and throughput

Transmission speed

Using low speeds when the length, and so the capacity of the connection cables, are high. When the distances are short and high speed may be used to communicate with a large number of devices (or communicate with devices that require a large number of bytes) Are established named speeds from "1" (1 k bits/Sec) up to "12" (4 mega bits per second)

Maximum number of bytes

Depending on the baud rate the maximum number of bytes that can be transmitted is listed in the following table.

Times, distances and bytes

Speed	Cell time	bit/s	Bytes in 15 mS	Bytes in 30 mS	Max distance	Max capacity	Max slaves
1	1 mS	1K	1	3	10 Km	1 uF	3
2	500 uS	2K	3	6	5 Km	500 nF	6
3	200 uS	5K	4	8	2 Km	200 nF	15
4	100 uS	10K	15	30	1 Km	100 nF	30
5	50 uS	20K	30	60	500 m	50 nF	60
6	20 uS	50K	40	80	200 m	20 nF	150
7	10 uS	100K	150	300	100 m	10 nF	160
8	5 uS	200K	300	600	50 m	5 nF	80
9	2 uS	500K	400	800	20 m	2 nF	32
10	1 uS	1M	1500	3000	10 m	1 nF	16
11	500 nS	2M	3000	6000	5 m	500 pF	8
12	250 nS	4M	6000	12000	2.5 m	250 pF	4

All devices must implement at least the speed "7" that is considered the default speed. If you want to set a different speed from the "7" all devices in the chain must support it.

The "maximum distance" depends on the characteristics of cables, the table values are calculated for a shielded cable RG58 from 50 ohm with ability to 100pF per meter.

At low speeds the number of "slaves" is limited by the maximum number of bytes that can be transmitted in 30 milliseconds. (each slave uses at least one byte and you claim a repeat fast enough to make fluid movements)

At high speed, the number of "slaves" is limited by the maximum capacity, each "slave" Adds a capacity of approximately 40..60 PF and reduces the maximum distance of approximately 50 cm.

The capacity was calculated on the basis of 30pF per "slave" plus other 20pF to a connection cable from 20 cm and other 10pF to take account of the additional resistance due to "bilateral switch" (total: 60pF)

Maximum length of the communication cable vs supply current and resistance per meter

Cable type ---> Max current (peak values)	**H1500 / H1000 / H500 / H155 / RG11** 20 milli ohm or less per meter	**RG58 / RG59U / H155 / net cables** about50 milli ohm per metro	**RG59 / RG6 / phone twisted wires / net cables** about 100 milli ohm per meter
10 mA	1 Km	400 m	200 m
20 mA	500 m	200 m	100 m
50 mA	200 m	80 m	40 m
80 mA	125 m	50 m	25 m
100 mA	100 m	40 m	20 m
200 mA	50 m	20 m	10 m
400 mA	25 m	10 m	5 m
500 mA	20 m	8 m	4 m
800 mA	12.5 m	5 m	2.5 m
1 A	10 m	4 m	2 m

In the calculation of the distance we take into account that the voltage drop on the ground, does not exceed 200mV.

The voltage drop on power cable, not causing transmission errors, It can also be much higher (the 5 V can drop down to 3.3 V without creating problems).

In case of shielded cables on ground is the screen, which usually has less resistance than declared, so the distance will be greater.

Cables electrical capacity

The "maximum length" values indicated in the table above are valid only for connecting cable with a capacity of about 100 pF per meter. The following table shows the corrections to be applied for the most commonly used cables.

Cable	External diameter (mm)	Impedance (ohm)	Capacity (pF/mt.)	Resistance (milli ohm / meter)	Max length correction
H1500	15	50	80	4	x 1.25
H1000	10.3	50	80	11	x 1.25
RG213	10.3	50	100		-
H500	9.8	50	82	15	x 1.22
H155	5.8	50	82	32	x 1.22
RG8	10	52	90		-
RG11 (TV)	10.3	75	60	21	x 1.7
RG59 (TV)	6.15	75	67	159	x 1.5
RG6 (TVsat)	6.8	75	51	100	x 2.0
RG56/U (TV)	6.9	75	53		x 2.0
RG59/U (TV)	4.5	75	53	45	x 2.0
RG58	5.2	50	100	53	-
RG142	4.95	50	96		-
RG174	2.8	50	100		-
RG178	1.85	50	95		-
RG179	2.55	75	64		x 1.5
RG187	2.7	75	65		x 1.5
RG188	2.7	50	95		-
RG196	1.9	50	93		-
RG316	2.5	50	95		-
Net cable			min 50 max 130	min 60 max 200	x 2.0 x 0.7
PC audio cable			min 120 max 300	min 500 max 3000	x 0.8 x 0.5
Microphonic cables			min 60 max 300		x 1.7 x 0.3
Phone twisted cable			50	100	x 2.0

There are also low-capacity cables (little used and hard to find):

•Rg62 – 93 ohm – 44 pF/mt
•RG71 – 93 ohm – 44 pF/mt
•Rg210 – 93 ohm – 44 pF/mt
•RG63 – 125 ohm – 33pF/mt
•RG114 – 185 ohm – 27pF/mt

Measure the ability of an unknown cable:
- Prepare the perfect skinning shielded cable and retaining insulated central strand.
- Measure between Central and outer shield with a meter or capacitance meter.
- To improve measurement accuracy, use five or ten metres of cable.
- Divide the value of Picofarads measured by the number of meters of cable.

Device types

The devices are labeled with a number from 0 at 199 identifying her "Type".

At the stage of recognition and numbering each device identifies itself with this "Type".

Currently are defined the following devices:

Device Type	Speed min	Speed max	In out Pins	Power	Name
0					Special type "custom"
1	1	12	1	12mA	Capacitive Sensor Hi Quality
2	1	12	10		InOut Servo
3	1	12	12		InOut Generic
4	1	12	12		InOut
5			6		Virtual Master Pins (first version)
8			10		Virtual Master Pins - V2
9			12		Virtual Master Pins - V4
255					UNKNOWN

Max number of devices

The maximum number of connectable devices is limited by:

- The maximum number of bytes that can be transmitted depending on the selected speed.
- The maximum current that the "master" can provide (normally 500 mA)
- The maximum number of devices supported by the Protocol is 200 (from 0 at 199)

Output - Pin Types

Pins are labeled with a number from 0 to 255 that identifies the "PinType"

Output Pin Type	Name	Master to Slave bytes	Slave to Master bytes
0	UNUSED	0	0
1	DIG_OUT	1	0
2	PWM_8	1	0
3	PWM_16	2	0
4	SERVO_8	1	0
5	SERVO_16	2	0
6	STEPPER	4	0
7	PWM_FAST	5	0

Input - Pin Types

Pins are labeled with a number from 0 to 255 that identifies the "PinType"

Input Pin Type	Name	Master to Slave bytes	Slave to Master bytes
129	DIG_IN	0	1
130	DIG_IN_PU	0	1
131	ADC_8	0	1
132	ADC_16	0	2
133	CAP_8	0	1
134	CAP_16	0	2
135	RES_8	0	1
136	RES_16	0	2
140	COUNTER	0	2
141	COUNTER_PU	0	2
142	FAST_COUNTER	0	2
143	FAST_COUNTER_PU	0	2
144	PERIOD	0	4
145	PERIOD_PU	0	4
146	SLOW_PERIOD	0	4
147	SLOW_PERIOD_PU	0	4
150	USOUND_SENSOR	0	2
160	CAP_SENSOR	0	3
165	STEPPER_DIR	0	4
180	ENCODER_A	0	2
181	ENCODER_A_PU	0	2
182	ENCODER_B	0	0
183	ENCODER_B_PU	0	0
175	ADC_24	0	1
176	ADC_24_DIN	0	0
177	ADC_24_DOUT	0	0

Master to Slaves communications (serial line)

Start byte	Transmission type	Transmission	Reception
255 (*4)	Special extended command (for future expansions)	1 byte (extension) >> see the extended commands table	
254 (*1)	**RecogStart** Rcognizing and numbering start	1 byte (number of data bytes = 0) 1 byte (CRC - Cmd / 0)	
253 (*2)	**Recog** Send progressive number and type request	1 byte (number of data bytes = 1) 1 byte (da 0 a 199) 1 byte (CRC - Cmd / Nbytes / Type)	1 byte (tipo) 1 byte (CRC)
251 (*3)	**FastDataExchange** Fast exchange on a single USB cycle.	1 byte (number of data bytes = 0) 1 byte (CRC - Cmd / 0) From 0 to 60 data bytes	From 0 to 63 data bytes
249 (*4)	**SetupSlavePins** Send configurations for the "slave" pins	1 byte (slave index) 1 byte (number of data bytes) nn byte (PinTypes: 1 byte each pin) 1 byte (CRC - Cmd/SlaveId/ etc....)	1 byte (slave index) *1 byte (CRC) on the preceding byte*
248 (*4)	**SetMasterName** Send the name for the "master"	nn byte (name chars ending with zero)	
247 (*4)	**GetMasterName** Get the name of the "master"		nn byte (name chars ending with zero)
246 (*4)	**SendValuesToSlave** Send "n" bytes to slave "m" (max 56 byte)	1 byte (slave index) 1 byte (number of bytes) byte 1 . . . byte n 1 byte (CRC -Cmd/SlaveId/nBytes/n)	1 byte (slave index) 1 byte (CRC) *on the preceding byte*
245 (*4)	**GetValuesFromSlave** Request of "n" bytes from slave "m" (max 56 byte)	1 byte (slave index) 1 byte (number of bytes) 1 byte (CRC - Cmd/SlaveId/nBytes)	byte 1 . . . byte n 1 byte (slave index) 1 byte (CRC) on n + 1 preceding bytes
244 (*4)	**SendBytesToSlave** Send "n" bytes to slave "m" (max 56 byte)	1 byte (slave index) 1 byte (number of bytes) byte 1 . . . byte n 1 byte (CRC -Cmd/SlaveId/nBytes/n)	1 byte (slave index) 1 byte (CRC) *on the preceding byte*
243 (*4)	**GetBytesFromSlave** Request of "n" bytes from slave "m" (max 56 byte)	1 byte (slave index) 1 byte (number of bytes) 1 byte (CRC - Cmd/SlaveId/nBytes)	byte 1 . . . byte n 1 byte (slave index) 1 byte (CRC) on n + 1 preceding bytes
199 (*5)	**SetSpeed**	1 byte (CommSpeed) 1 byte (CRC - Cmd/ComSpd)	
0	No action		

(*1) Service commands.
(*2) The command Recog is used from master and slaves during recognizing only.
(*3) Fast communication – The master exchanges values with all the slaves , using a single USB exchange.
(*4) Secure communication commands to the single slave
(*5) Special commands

All commands have codes from 200 at 255, to prevent, in case of errors, IDs and slave types (from 0 at 199) could be interpreted as a command. (Set-Speed doesn't count because it is never sent over the serial line but only by HAL, to the Master, via USB)

Master Slave communications (Send and Get commands)

SendValuesToSlave

Sends the values to the Output Pin of a slave (a virtual slave on the Master or an hardware physical Slave)

GetValuesFromSlave

Reads values from the Input Pin of a slave (a virtual slave on the Master or an hardware physical Slave)

SendBytesToSlave

Send generic bytes (a configuration for example), to a slave (a virtual slave on the Master or an hardware physical Slave)

GetBytesFromSlave

Reads generic bytes (for example the state), from a slave (a virtual slave on the Master or an hardware physical Slave)

Host to Master communication (USB)

Command from "Host" to "Master"

Command name	ID USB_TxData[0]	PARAMETERS USB_TxData[1 to n]
RecogStart	CommandID,	Nbytes
FastDataExchange	CommandID,	0 to 60 data bytes
SetupSlavePins	CommandID,	SlaveId, Nbytes
SetMasterName	CommandID,	MasterName (zero terminated)
GetMasterName	CommandID	
SendValuesToSlave	CommandID,	SlaveId, Nbytes, Byte1....ByteN
GetValuesFromSlave	CommandID,	SlaveId, Nbytes
SendBytesToSlave	CommandID,	SlaveId, Nbytes, Byte1....ByteN
GetBytesFromSlave	CommandID,	SlaveId, Nbytes
SetSpeed	CommandID,	Comm Speed

Responses from "Master" to "Host"

Command name	RESPONSE USB_RxData[0]	RETURN VALUES USB_RxData[1 to n]
RecogStart	0 = OK	Nslaves, SlaveType1....SlaveTypeN
FastDataExchange	0 = OK	0 to 63 data bytes
SetupSlavePins	0 = OK	- - -
SetMasterName	0 = OK	- - -
GetMasterName	0 = OK	MasterName (zero terminated)
SendValuesToSlave	0 = OK	- - -
GetValuesFromSlave	0 = OK	Byte1....ByteN
SendBytesToSlave	0 = OK	- - -
GetBytesFromSlave	0 = OK	Byte1....ByteN
SetSpeed	0 = OK	- - -

The zero position of the USB Buffer indicates whether the command was executed from "Master" successfully.

Calculation of the CRC

All CRC used are calculated over a certain number of consecutive bytes and the CRC result is a byte.

CRC calculation uses an algorithm based on "Longitudinal redundancy check".

Longitudinal redundancy check

```
-----------------------------------------------------------------
Dim CRC as Byte
CRC = 0
For each byte b
     CRC = CRC Xor b
Next
```

To avoid "collisions" between simple sequences (for example, 0000 = 1111 or 123 = 321) and simple sequences that produce valid CRC (for example, 0000 with CRC = 0) the above method is modified with a permutation.

The resulting algorithm is efficient and extremely simple.

Calculation of the CRC used in this Protocol

```
-----------------------------------------------------------------
Dim CRC as Byte
CRC = 0
For each byte b
    CRC = CRC Xor b
    CRC = CRC + 1
Next
```

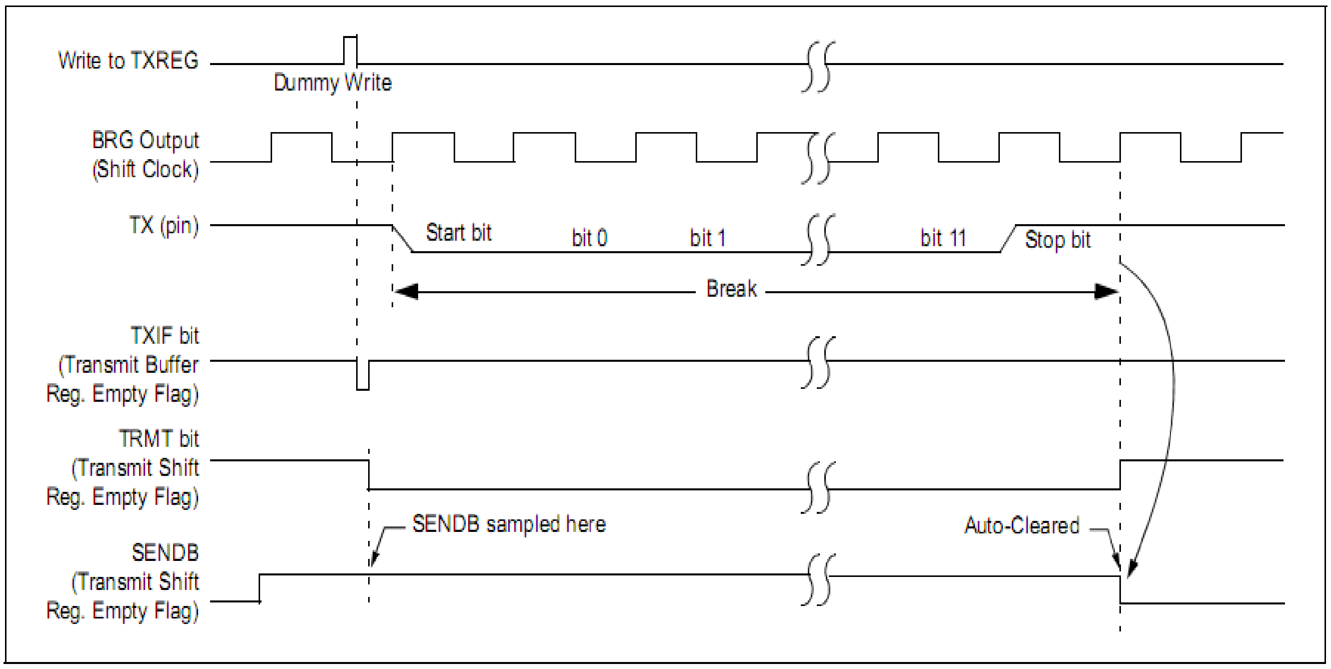
Setting the baud rate

If you decide to use a baud rate other than the default one, then the "master" should communicate to all devices in the chain the new speed.

This setting should be possible even before making a loop device recognition and must be also possible with very long transmission lines. Therefore there is a special command that is now shown.

1 – The Master maintains the high line for 50 mS
2 – All the slaves are definitely placed pending one character
3 – The Master generates a BREAK (low level line 12 bit at minimum speed
4 – The Master issues a 55 character (01010101) the desired baud rate
5 – All slaves infer the baud rate from this bytes (Auto-baud)
6 – The Master sends one byte that specifies the "Speed" (from 1 at 12)
7 – The Master sends one byte of CRC calculated on two bytes (cmd/speed)
8 – If the Slave is a mistake does not change its speed

BREAK CHARACTER SEQUENCE



Checking the baud rate

If you set a speed too high for transmission line in use some devices in the chain may not be able to support the speed setting and errors can occur when transmitting data.

If transmission errors are zero or less than the 0.1% then the speed set is valid.

Recognition and numbering

1 – The Master outputs to the Slave the whole sequence of "Speed" setting to ensure that all communicate at the same speed.

2 – The Master does not transmit commands to 50 milliseconds.

3 – At this point all slaves *should* be waiting for a command.

4 – The Master issues a code "254” (RecogStart).

5 – All the slaves they put the weak-pull-up (100-400 uA) on the input-output and open the output connection to downstream devices. They no longer respond to any commands except "253” (Recog).

6 – Il Master” emette un codice “253” (type request) and then a byte with the number “0”, the first device in the chain meets a byte with its type, Removes the pull-up, connect to the downstream Slave and not responding to any commands.

7 – Il Master emette un codice “253” (type request) and then a byte with the number “1”, the second device in the chain meets a byte with its type, Removes the pull-up, connect to the downstream Slave and not responding to any commands.

8 – Il Master emette un codice “253” (type request) and then a byte with the number "2”,

…. the “253” (type request) repeats up to 200 times

When no longer responds to a duration greater than 10 bit at current throughput, means that the chain is over. To avoid this calculation, You can use a time-out to 12 mS that always works, even at the lowest speed.

9 – The Master issues toward the slaves all setting sequence "Speed", that shows all the slaves in the normal communication mode.

10 – The Master informs the Host (PC) via USB Slave recognized number and type of each.

- - - - - - - -

Roberto Cena & Livio Cicala – 2010 to 2016