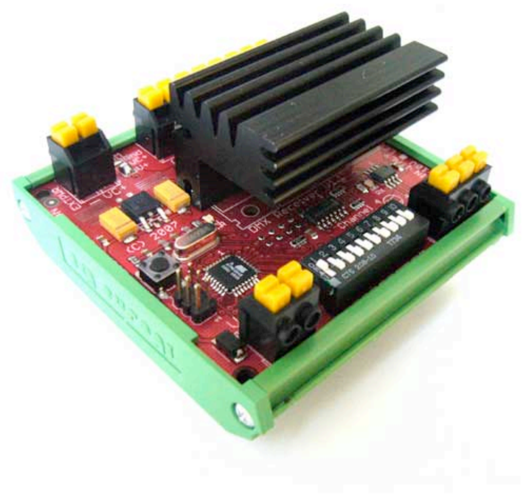
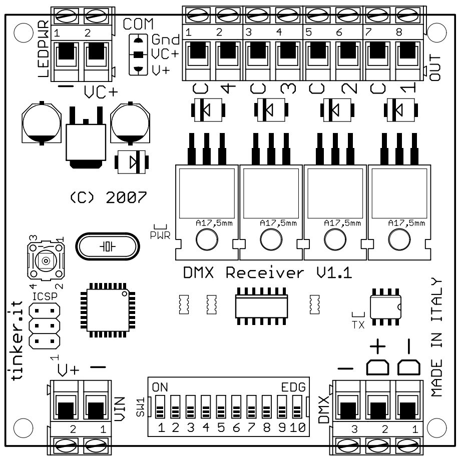
DMX to PWM converter

tinker.it! technology & design





Technical Data

Nominal Input Voltage (DC)	9 to 24 Vdc
Input Current	10 A Max
Output Current	2.5A Max/Ch
Number of DMX Channels	4
Addressing DMX Range	1 to 509
Dimensions	75 by 72 mm

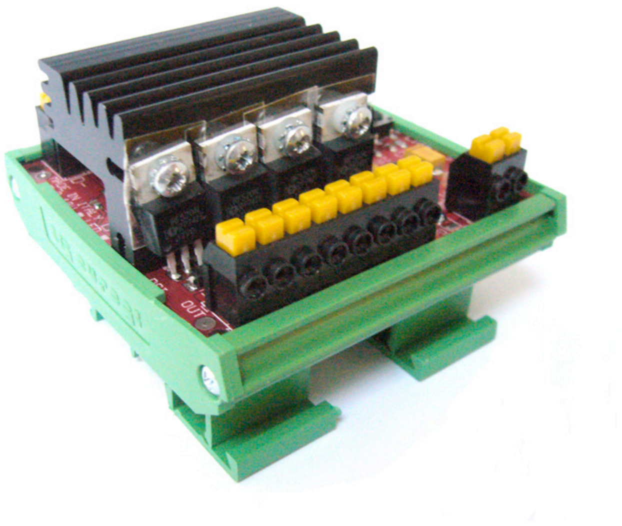
The DMX-PWM converter is used for brightness control of 4 low-voltage LED, incandescent lamps and devices that use variable voltage via the DMX protocol.

DMX messages are converted into PWM signals driving 4 power MOSFETs

DMX address is set via the DIP switches

Table of Contents

Introduction



Andrea Piccolo
16 giu, 12:29
Eliminato: Spazio

Andrea Piccolo
16 giu, 12:30
Testo aggiunto

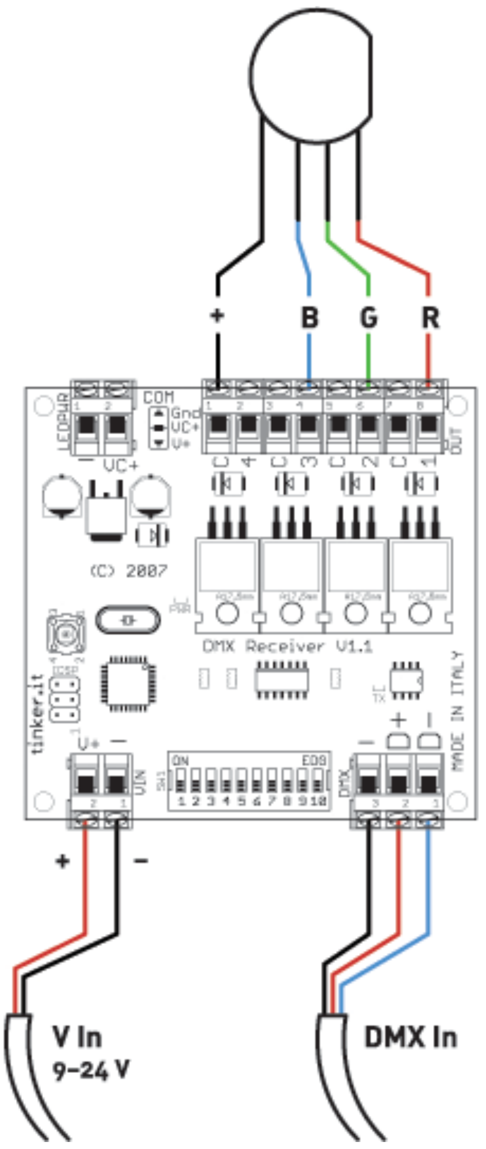
Andrea Piccolo
16 giu, 12:36
Testo aggiunto

The Tinker.it DMX to PWM Converter is a DMX receiver with 4 channels of PWM driven outputs for controlling low-voltage Incandescent and LED lamps and also any device that uses variable voltage, like DC motors. Each output is connected to a power MOSFET transistor that can be configured to use either a single power supply for both the circuit and the LEDs or two separate supplies.

DMX signal can be produced with different devices and controllers like Arduino. The Appendix C code examples are Arduino based.

Installation

Connections



Connecting the DMX-PWM converter requires providing a 9 to 24V power supply to the VIN inputs and providing a valid DMX signal to the DMX inputs. LEDs can be connected to C1 to C4 outputs. If you need to power the LEDs with a different power supply connect it to LEDPWR and remove the solder jumper between V+ and VC+ near the EXTPWR connector.

Operation

Setting Address

The DMX address is set using the first 9 switches of the DMX address switch. The address is set as a binary number (with switch 1 the LSB, Appendix A lists addresses and their corresponding switch settings). As the unit is a 4 channel receiver, it will respond to data in the set address, and the following 3 addresses in the DMX data stream. Therefore, the address should not be set to over 509.

Using With DMX Input

In order to use the Tinker.it DMX to PWM Converter with a DMX feed, simply set the desired address and connect the DMX feed to the DMX input. Remember to terminate the DMX feed at the end of the chain with a 120Ω resistor between the DMX + and - lines. When data is being correctly received, the yellow LED should blink fast.

Test Mode

The Tinker.it DMX to PWM Converter-3 has test mode which ramps each individual output up to full brightness then back off in sequence. This test mode is entered by setting the DMX address to 0 (all switches down). While in this mode, any incoming DMX data is ignored. Once the address is changed from 0, pressing the RESET button will return the device to normal operation.

Appendix A
DIP Switch Settings

Switch Up = 1

Switch Down = 0

Ad-dress	Switch								
	1	2	3	4	5	6	7	8	9
Test Mode	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0
5	1	0	1	0	0	0	0	0	0
6	0	1	1	0	0	0	0	0	0
7	1	1	1	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0
9	1	0	0	1	0	0	0	0	0
10	0	1	0	1	0	0	0	0	0
11	1	1	0	1	0	0	0	0	0
12	0	0	1	1	0	0	0	0	0
13	1	0	1	1	0	0	0	0	0
14	0	1	1	1	0	0	0	0	0
15	1	1	1	1	0	0	0	0	0
16	0	0	0	0	1	0	0	0	0
17	1	0	0	0	1	0	0	0	0
18	0	1	0	0	1	0	0	0	0
19	1	1	0	0	1	0	0	0	0
20	0	0	1	0	1	0	0	0	0
21	1	0	1	0	1	0	0	0	0
22	0	1	1	0	1	0	0	0	0
23	1	1	1	0	1	0	0	0	0
24	0	0	0	1	1	0	0	0	0
25	1	0	0	1	1	0	0	0	0
26	0	1	0	1	1	0	0	0	0
27	1	1	0	1	1	0	0	0	0
28	0	0	1	1	1	0	0	0	0
29	1	0	1	1	1	0	0	0	0
30	0	1	1	1	1	0	0	0	0
31	1	1	1	1	1	0	0	0	0
32	0	0	0	0	0	1	0	0	0
33	1	0	0	0	0	1	0	0	0
34	0	1	0	0	0	1	0	0	0
35	1	1	0	0	0	1	0	0	0
36	0	0	1	0	0	1	0	0	0
37	1	0	1	0	0	1	0	0	0
38	0	1	1	0	0	1	0	0	0
39	1	1	1	0	0	1	0	0	0
40	0	0	0	1	0	1	0	0	0
41	1	0	0	1	0	1	0	0	0
42	0	1	0	1	0	1	0	0	0
43	1	1	0	1	0	1	0	0	0
44	0	0	1	1	0	1	0	0	0
45	1	0	1	1	0	1	0	0	0
46	0	1	1	1	0	1	0	0	0
47	1	1	1	1	0	1	0	0	0
48	0	0	0	0	1	1	0	0	0
49	1	0	0	0	1	1	0	0	0
50	0	1	0	0	1	1	0	0	0
51	1	1	0	0	1	1	0	0	0

52	0	0	1	0	1	1	0	0	0
53	1	0	1	0	1	1	0	0	0
54	0	1	1	0	1	1	0	0	0
55	1	1	1	0	1	1	0	0	0
56	0	0	0	1	1	1	0	0	0
57	1	0	0	1	1	1	0	0	0
58	0	1	0	1	1	1	0	0	0
59	1	1	0	1	1	1	0	0	0
60	0	0	1	1	1	1	0	0	0
61	1	0	1	1	1	1	0	0	0
62	0	1	1	1	1	1	0	0	0
63	1	1	1	1	1	1	0	0	0
64	0	0	0	0	0	0	1	0	0
65	1	0	0	0	0	0	1	0	0
66	0	1	0	0	0	0	1	0	0
67	1	1	0	0	0	0	1	0	0
68	0	0	1	0	0	0	1	0	0
69	1	0	1	0	0	0	1	0	0
70	0	1	1	0	0	0	1	0	0
71	1	1	1	0	0	0	1	0	0
72	0	0	0	1	0	0	1	0	0
73	1	0	0	1	0	0	1	0	0
74	0	1	0	1	0	0	1	0	0
75	1	1	0	1	0	0	1	0	0
76	0	0	1	1	0	0	1	0	0
77	1	0	1	1	0	0	1	0	0
78	0	1	1	1	0	0	1	0	0
79	1	1	1	1	0	0	1	0	0
80	0	0	0	0	1	0	1	0	0
81	1	0	0	0	1	0	1	0	0
82	0	1	0	0	1	0	1	0	0
83	1	1	0	0	1	0	1	0	0
84	0	0	1	0	1	0	1	0	0
85	1	0	1	0	1	0	1	0	0
86	0	1	1	0	1	0	1	0	0
87	1	1	1	0	1	0	1	0	0
88	0	0	0	1	1	0	1	0	0
89	1	0	0	1	1	0	1	0	0
90	0	1	0	1	1	0	1	0	0
91	1	1	0	1	1	0	1	0	0
92	0	0	1	1	1	0	1	0	0
93	1	0	1	1	1	0	1	0	0
94	0	1	1	1	1	0	1	0	0
95	1	1	1	1	1	0	1	0	0
96	0	0	0	0	0	1	1	0	0
97	1	0	0	0	0	1	1	0	0
98	0	1	0	0	0	1	1	0	0
99	1	1	0	0	0	1	1	0	0
100	0	0	1	0	0	1	1	0	0
101	1	0	1	0	0	1	1	0	0
102	0	1	1	0	0	1	1	0	0
103	1	1	1	0	0	1	1	0	0
104	0	0	0	1	0	1	1	0	0
105	1	0	0	1	0	1	1	0	0
106	0	1	0	1	0	1	1	0	0
107	1	1	0	1	0	1	1	0	0
108	0	0	1	1	0	1	1	0	0
109	1	0	1	1	0	1	1	0	0
110	0	1	1	1	0	1	1	0	0
111	1	1	1	1	0	1	1	0	0

112	0	0	0	0	1	1	1	0	0
113	1	0	0	0	1	1	1	0	0
114	0	1	0	0	1	1	1	0	0
115	1	1	0	0	1	1	1	0	0
116	0	0	1	0	1	1	1	0	0
117	1	0	1	0	1	1	1	0	0
118	0	1	1	0	1	1	1	0	0
119	1	1	1	0	1	1	1	0	0
120	0	0	0	1	1	1	1	0	0
121	1	0	0	1	1	1	1	0	0
122	0	1	0	1	1	1	1	0	0
123	1	1	0	1	1	1	1	0	0
124	0	0	1	1	1	1	1	0	0
125	1	0	1	1	1	1	1	0	0
126	0	1	1	1	1	1	1	0	0
127	1	1	1	1	1	1	1	0	0
128	0	0	0	0	0	0	0	1	0
129	1	0	0	0	0	0	0	1	0
130	0	1	0	0	0	0	0	1	0
131	1	1	0	0	0	0	0	1	0
132	0	0	1	0	0	0	0	1	0
133	1	0	1	0	0	0	0	1	0
134	0	1	1	0	0	0	0	1	0
135	1	1	1	0	0	0	0	1	0
136	0	0	0	1	0	0	0	1	0
137	1	0	0	1	0	0	0	1	0
138	0	1	0	1	0	0	0	1	0
139	1	1	0	1	0	0	0	1	0
140	0	0	1	1	0	0	0	1	0
141	1	0	1	1	0	0	0	1	0

142	0	1	1	1	0	0	0	1	0
143	1	1	1	1	0	0	0	1	0
144	0	0	0	0	1	0	0	1	0
145	1	0	0	0	1	0	0	1	0
146	0	1	0	0	1	0	0	1	0
147	1	1	0	0	1	0	0	1	0
148	0	0	1	0	1	0	0	1	0
149	1	0	1	0	1	0	0	1	0
150	0	1	1	0	1	0	0	1	0
151	1	1	1	0	1	0	0	1	0
152	0	0	0	1	1	0	0	1	0
153	1	0	0	1	1	0	0	1	0
154	0	1	0	1	1	0	0	1	0
155	1	1	0	1	1	0	0	1	0
156	0	0	1	1	1	0	0	1	0
157	1	0	1	1	1	0	0	1	0
158	0	1	1	1	1	0	0	1	0
159	1	1	1	1	1	0	0	1	0
160	0	0	0	0	0	1	0	1	0
161	1	0	0	0	0	1	0	1	0
162	0	1	0	0	0	1	0	1	0
163	1	1	0	0	0	1	0	1	0
164	0	0	1	0	0	1	0	1	0
165	1	0	1	0	0	1	0	1	0
166	0	1	1	0	0	1	0	1	0
167	1	1	1	0	0	1	0	1	0
168	0	0	0	1	0	1	0	1	0
169	1	0	0	1	0	1	0	1	0
170	0	1	0	1	0	1	0	1	0
171	1	1	0	1	0	1	0	1	0

172	0	0	1	1	0	1	0	1	0
173	1	0	1	1	0	1	0	1	0
174	0	1	1	1	0	1	0	1	0
175	1	1	1	1	0	1	0	1	0
176	0	0	0	0	1	1	0	1	0
177	1	0	0	0	1	1	0	1	0
178	0	1	0	0	1	1	0	1	0
179	1	1	0	0	1	1	0	1	0
180	0	0	1	0	1	1	0	1	0
181	1	0	1	0	1	1	0	1	0
182	0	1	1	0	1	1	0	1	0
183	1	1	1	0	1	1	0	1	0
184	0	0	0	1	1	1	0	1	0
185	1	0	0	1	1	1	0	1	0
186	0	1	0	1	1	1	0	1	0
187	1	1	0	1	1	1	0	1	0
188	0	0	1	1	1	1	0	1	0
189	1	0	1	1	1	1	0	1	0
190	0	1	1	1	1	1	0	1	0
191	1	1	1	1	1	1	0	1	0
192	0	0	0	0	0	0	1	1	0
193	1	0	0	0	0	0	1	1	0
194	0	1	0	0	0	0	1	1	0
195	1	1	0	0	0	0	1	1	0
196	0	0	1	0	0	0	1	1	0
197	1	0	1	0	0	0	1	1	0
198	0	1	1	0	0	0	1	1	0
199	1	1	1	0	0	0	1	1	0
200	0	0	0	1	0	0	1	1	0
201	1	0	0	1	0	0	1	1	0

202	0	1	0	1	0	0	1	1	0
203	1	1	0	1	0	0	1	1	0
204	0	0	1	1	0	0	1	1	0
205	1	0	1	1	0	0	1	1	0
206	0	1	1	1	0	0	1	1	0
207	1	1	1	1	0	0	1	1	0
208	0	0	0	0	1	0	1	1	0
209	1	0	0	0	1	0	1	1	0
210	0	1	0	0	1	0	1	1	0
211	1	1	0	0	1	0	1	1	0
212	0	0	1	0	1	0	1	1	0
213	1	0	1	0	1	0	1	1	0
214	0	1	1	0	1	0	1	1	0
215	1	1	1	0	1	0	1	1	0
216	0		0	1	1	0	1	1	0
217	1	0	0	1	1	0	1	1	0
218	0	1	0	1	1	0	1	1	0
219	1	1	0	1	1	0	1	1	0
220	0	0	1	1	1	0	1	1	0
221	1	0	1	1	1	0	1	1	0
222	0	1	1	1	1	0	1	1	0
223	1	1	1	1	1	0	1	1	0
224	0	0	0	0	0	1	1	1	0
225	1	0	0	0	0	1	1	1	0
226	0	1	0	0	0	1	1	1	0
227	1	1	0	0	0	1	1	1	0
228	0	0	1	0	0	1	1	1	0
229	1	0	1	0	0	1	1	1	0
230	0	1	1	0	0	1	1	1	0
231	1	1	1	0	0	1	1	1	0

232	0	0	0	1	0	1	1	1	0
233	1	0	0	1	0	1	1	1	0
234	0	1	0	1	0	1	1	1	0
235	1	1	0	1	0	1	1	1	0
236	0	0	1	1	0	1	1	1	0
237	1	0	1	1	0	1	1	1	0
238	0	1	1	1	0	1	1	1	0
239	1	1	1	1	0	1	1	1	0
240	0	0	0	0	1	1	1	1	0
241	1	0	0	0	1	1	1	1	0
242	0	1	0	0	1	1	1	1	0
243	1	1	0	0	1	1	1	1	0
244	0	0	1	0	1	1	1	1	0
245	1	0	1	0	1	1	1	1	0
246	0	1	1	0	1	1	1	1	0
247	1	1	1	0	1	1	1	1	0
248	0	0	0	1	1	1	1	1	0
249	1	0	0	1	1	1	1	1	0
250	0	1	0	1	1	1	1	1	0
251	1	1	0	1	1	1	1	1	0
252	0	0	1	1	1	1	1	1	0
253	1	0	1	1	1	1	1	1	0
254	0	1	1	1	1	1	1	1	0
255	1	1	1	1	1	1	1	1	0
256	0	0	0	0	0	0	0	0	1
257	1	0	0	0	0	0	0	0	1
258	0	1	0	0	0	0	0	0	1
259	1	1	0	0	0	0	0	0	1
260	0	0	1	0	0	0	0	0	1
261	1	0	1	0	0	0	0	0	1
262	0	1	1	0	0	0	0	0	1
263	1	1	1	0	0	0	0	0	1
264	0	0	0	1	0	0	0	0	1
265	1	0	0	1	0	0	0	0	1
266	0	1	0	1	0	0	0	0	1
267	1	1	0	1	0	0	0	0	1
268	0	0	1	1	0	0	0	0	1
269	1	0	1	1	0	0	0	0	1
270	0	1	1	1	0	0	0	0	1
271	1	1	1	1	0	0	0	0	1
272	0	0	0	0	1	0	0	0	1
273	1	0	0	0	1	0	0	0	1
274	0	1	0	0	1	0	0	0	1
275	1	1	0	0	1	0	0	0	1
276	0	0	1	0	1	0	0	0	1
277	1	0	1	0	1	0	0	0	1
278	0	1	1	0	1	0	0	0	1
279	1	1	1	0	1	0	0	0	1
280	0	0	0	1	1	0	0	0	1
281	1	0	0	1	1	0	0	0	1
282	0	1	0	1	1	0	0	0	1
283	1	1	0	1	1	0	0	0	1
284	0	0	1	1	1	0	0	0	1
285	1	0	1	1	1	0	0	0	1
286	0	1	1	1	1	0	0	0	1
287	1	1	1	1	1	0	0	0	1
288	0	0	0	0	0	1	0	0	1
289	1	0	0	0	0	1	0	0	1
290	0	1	0	0	0	1	0	0	1
291	1	1	0	0	0	1	0	0	1

292	0	0	1	0	0	1	0	0	1
293	1	0	1	0	0	1	0	0	1
294	0	1	1	0	0	1	0	0	1
295	1	1	1	0	0	1	0	0	1
296	0	0	0	1	0	1	0	0	1
297	1	0	0	1	0	1	0	0	1
298	0	1	0	1	0	1	0	0	1
299	1	1	0	1	0	1	0	0	1
300	0	0	1	1	0	1	0	0	1
301	1	0	1	1	0	1	0	0	1
302	0	1	1	1	0	1	0	0	1
303	1	1	1	1	0	1	0	0	1
304	0	0	0	0	1	1	0	0	1
305	1	0	0	0	1	1	0	0	1
306	0	1	0	0	1	1	0	0	1
307	1	1	0	0	1	1	0	0	1
308	0	0	1	0	1	1	0	0	1
309	1	0	1	0	1	1	0	0	1
310	0	1	1	0	1	1	0	0	1
311	1	1	1	0	1	1	0	0	1
312	0	0	0	1	1	1	0	0	1
313	1	0	0	1	1	1	0	0	1
314	0	1	0	1	1	1	0	0	1
315	1	1	0	1	1	1	0	0	1
316	0	0	1	1	1	1	0	0	1
317	1	0	1	1	1	1	0	0	1
318	0	1	1	1	1	1	0	0	1
319	1	1	1	1	1	1	0	0	1
320	0	0	0	0	0	0	1	0	1
321	1	0	0	0	0	0	1	0	1

322	0	1	0	0	0	0	1	0	1
323	1	1	0	0	0	0	1	0	1
324	0	0	1	0	0	0	1	0	1
325	1	0	1	0	0	0	1	0	1
326	0	1	1	0	0	0	1	0	1
327	1	1	1	0	0	0	1	0	1
328	0	0	0	1	0	0	1	0	1
329	1	0	0	1	0	0	1	0	1
330	0	1	0	1	0	0	1	0	1
331	1	1	0	1	0	0	1	0	1
332	0	0	1	1	0	0	1	0	1
333	1	0	1	1	0	0	1	0	1
334	0	1	1	1	0	0	1	0	1
335	1	1	1	1	0	0	1	0	1
336	0	0	0	0	1	0	1	0	1
337	1	0	0	0	1	0	1	0	1
338	0	1	0	0	1	0	1	0	1
339	1	1	0	0	1	0	1	0	1
340	0	0	1	0	1	0	1	0	1
341	1	0	1	0	1	0	1	0	1
342	0	1	1	0	1	0	1	0	1
343	1	1	1	0	1	0	1	0	1
344	0	0	0	1	1	0	1	0	1
345	1	0	0	1	1	0	1	0	1
346	0	1	0	1	1	0	1	0	1
347	1	1	0	1	1	0	1	0	1
348	0	0	1	1	1	0	1	0	1
349	1	0	1	1	1	0	1	0	1
350	0	1	1	1	1	0	1	0	1
351	1	1	1	1	1	0	1	0	1

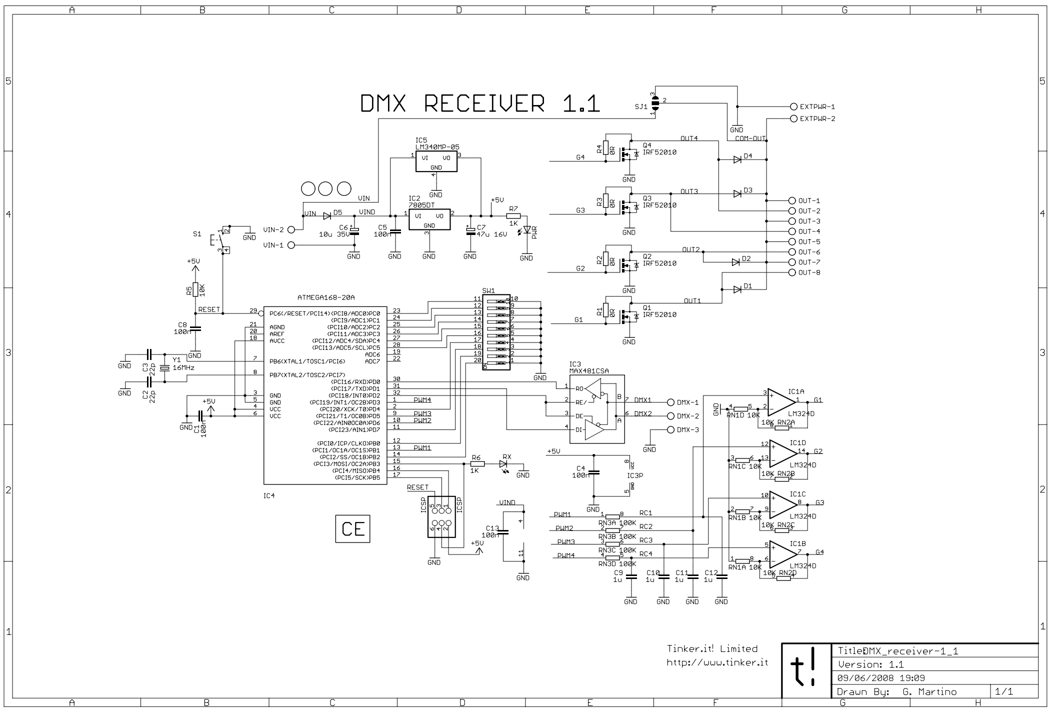
352	0	0	0	0	0	1	1	0	1
353	1	0	0	0	0	1	1	0	1
354	0	1	0	0	0	1	1	0	1
355	1	1	0	0	0	1	1	0	1
356	0	0	1	0	0	1	1	0	1
357	1	0	1	0	0	1	1	0	1
358	0	1	1	0	0	1	1	0	1
359	1	1	1	0	0	1	1	0	1
360	0	0	0	1	0	1	1	0	1
361	1	0	0	1	0	1	1	0	1
362	0	1	0	1	0	1	1	0	1
363	1	1	0	1	0	1	1	0	1
364	0	0	1	1	0	1	1	0	1
365	1	0	1	1	0	1	1	0	1
366	0	1	1	1	0	1	1	0	1
367	1	1	1	1	0	1	1	0	1
368	0	0	0	0	1	1	1	0	1
369	1	0	0	0	1	1	1	0	1
370	0	1	0	0	1	1	1	0	1
371	1	1	0	0	1	1	1	0	1
372	0	0	1	0	1	1	1	0	1
373	1	0	1	0	1	1	1	0	1
374	0	1	1	0	1	1	1	0	1
375	1	1	1	0	1	1	1	0	1
376	0	0	0	1	1	1	1	0	1
377	1	0	0	1	1	1	1	0	1
378	0	1	0	1	1	1	1	0	1
379	1	1	0	1	1	1	1	0	1
380	0	0	1	1	1	1	1	0	1
381	1	0	1	1	1	1	1	0	1

382	0	1	1	1	1	1	1	0	1
383	1	1	1	1	1	1	1	0	1
384	0	0	0	0	0	0	0	1	1
385	1	0	0	0	0	0	0	1	1
386	0	1	0	0	0	0	0	1	1
387	1	1	0	0	0	0	0	1	1
388	0	0	1	0	0	0	0	1	1
389	1	0	1	0	0	0	0	1	1
390	0	1	1	0	0	0	0	1	1
391	1	1	1	0	0	0	0	1	1
392	0	0	0	1	0	0	0	1	1
393	1	0	0	1	0	0	0	1	1
394	0	1	0	1	0	0	0	1	1
395	1	1	0	1	0	0	0	1	1
396	0	0	1	1	0	0	0	1	1
397	1	0	1	1	0	0	0	1	1
398	0	1	1	1	0	0	0	1	1
399	1	1	1	1	0	0	0	1	1
400	0	0	0	0	1	0	0	1	1
401	1	0	0	0	1	0	0	1	1
402	0	1	0	0	1	0	0	1	1
403	1	1	0	0	1	0	0	1	1
404	0	0	1	0	1	0	0	1	1
405	1	0	1	0	1	0	0	1	1
406	0	1	1	0	1	0	0	1	1
407	1	1	1	0	1	0	0	1	1
408	0	0	0	1	1	0	0	1	1
409	1	0	0	1	1	0	0	1	1
410	0	1	0	1	1	0	0	1	1
411	1	1	0	1	1	0	0	1	1

412	0	0	1	1	1	0	0	1	1
413	1	0	1	1	1	0	0	1	1
414	0	1	1	1	1	0	0	1	1
415	1	1	1	1	1	0	0	1	1
416	0	0	0	0	0	1	0	1	1
417	1	0	0	0	0	1	0	1	1
418	0	1	0	0	0	1	0	1	1
419	1	1	0	0	0	1	0	1	1
420	0	0	1	0	0	1	0	1	1
421	1	0	1	0	0	1	0	1	1
422	0	1	1	0	0	1	0	1	1
423	1	1	1	0	0	1	0	1	1
424	0	0	0	1	0	1	0	1	1
425	1	0	0	1	0	1	0	1	1
426	0	1	0	1	0	1	0	1	1
427	1	1	0	1	0	1	0	1	1
428	0	0	1	1	0	1	0	1	1
429	1	0	1	1	0	1	0	1	1
430	0	1	1	1	0	1	0	1	1
431	1	1	1	1	0	1	0	1	1
432	0	0	0	0	1	1	0	1	1
433	1	0	0	0	1	1	0	1	1
434	0	1	0	0	1	1	0	1	1
435	1	1	0	0	1	1	0	1	1
436	0	0	1	0	1	1	0	1	1
437	1	0	1	0	1	1	0	1	1
438	0	1	1	0	1	1	0	1	1
439	1	1	1	0	1	1	0	1	1
440	0	0	0	1	1	1	0	1	1
441	1	0	0	1	1	1	0	1	1
442	0	1	0	1	1	1	0	1	1
443	1	1	0	1	1	1	0	1	1
444	0	0	1	1	1	1	0	1	1
445	1	0	1	1	1	1	0	1	1
446	0	1	1	1	1	1	0	1	1
447	1	1	1	1	1	1	0	1	1
448	0	0	0	0	0	0	1	1	1
449	1	0	0	0	0	0	1	1	1
450	0	1	0	0	0	0	1	1	1
451	1	1	0	0	0	0	1	1	1
452	0	0	1	0	0	0	1	1	1
453	1	0	1	0	0	0	1	1	1
454	0	1	1	0	0	0	1	1	1
455	1	1	1	0	0	0	1	1	1
456	0	0	0	1	0	0	1	1	1
457	1	0	0	1	0	0	1	1	1
458	0	1	0	1	0	0	1	1	1
459	1	1	0	1	0	0	1	1	1
460	0	0	1	1	0	0	1	1	1
461	1	0	1	1	0	0	1	1	1
462	0	1	1	1	0	0	1	1	1
463	1	1	1	1	0	0	1	1	1
464	0	0	0	0	1	0	1	1	1
465	1	0	0	0	1	0	1	1	1
466	0	1	0	0	1	0	1	1	1
467	1	1	0	0	1	0	1	1	1
468	0	0	1	0	1	0	1	1	1
469	1	0	1	0	1	0	1	1	1
470	0	1	1	0	1	0	1	1	1
471	1	1	1	0	1	0	1	1	1

472	0	0	0	1	1	0	1	1	1
473	1	0	0	1	1	0	1	1	1
474	0	1	0	1	1	0	1	1	1
475	1	1	0	1	1	0	1	1	1
476	0	0	1	1	1	0	1	1	1
477	1	0	1	1	1	0	1	1	1
478	0	1	1	1	1	0	1	11	
479	1	1	1	1	1	0	1	1	1
480	0	0	0	0	0	1	1	1	1
481	1	0	0	0	0	1	1	1	1
482	0	1	0	0	0	1	1	1	1
483	1	1	0	0	0	1	1	1	1
484	0	0	1	0	0	1	1	1	1
485	1	0	1	0	0	1	1	1	1
486	0	1	1	0	0	1	1	1	1
487	1	1	1	0	0	1	1	1	1
488	0	0	0	1	0	1	1	1	1
489	1	0	0	1	0	1	1	1	1
490	0	1	0	1	0	1	1	1	1
491	1	1	0	1	0	1	1	1	1
492	0	0	1	1	0	1	1	1	1
493	1	0	1	1	0	1	1	1	1
494	0	1	1	1	0	1	1	1	1
495	1	1	1	1	0	1	1	1	1
496	0	0	0	0	1	1	1	1	1
497	1	0	0	0	1	1	1	1	1
498	0	1	0	0	1	1	1	1	1
499	1	1	0	0	1	1	1	1	1
500	0	0	1	0	1	1	1	1	1
501	1	0	1	0	1	1	1	1	1
502	0	1	1	0	1	1	1	1	1
503	1	1	1	0	1	1	1	1	1
504	0	0	0	1	1	1	1	1	1
505	1	0	0	1	1	1	1	1	1
506	0	1	0	1	1	1	1	1	1
507	1	1	0	1	1	1	1	1	1
508	0	0	1	1	1	1	1	1	1
509	1	0	1	1	1	1	1	1	1

Appendix B: Schematic Diagram



Andrea Piccolo
16 giu, 12:53
Testo aggiunto

Appendix C: Code Examples

In these examples, Arduino is used to generate DMX signals. The DMX signals are then decoded by the DMX interface.

Example 1: Heart Beat

Here is a simple example of the code needed to generate a fading 'hearth beat' on channels 1,2 and 3

```
/*
 * DMX fade for arduino 008
 * based on the code of Tomek Ness and D. Cuartielles
 *
 * adapted to arduino 008 by Peter Szakal and Gabor Papp
 * http://nextlab.hu
 */

#include "pins_arduino.h"

int sig = 3; // signal

int value = 0;
int valueadd = 3;

/* Sends a DMX byte out on a pin.  Assumes a 16 MHz clock.
 * Disables interrupts, which will disrupt the millis() function if used
 * too frequently. */
void shiftDmxOut(int pin, int theByte)
{
```

```
int port_to_output[] = {
  NOT_A_PORT,
  NOT_A_PORT,
  _SFR_IO_ADDR(PORTB),
  _SFR_IO_ADDR(PORTC),
  _SFR_IO_ADDR(PORTD)
  };

  int portNumber = port_to_output[digitalPinToPort(pin)];
int pinMask = digitalPinToBitMask(pin);

// the first thing we do is to write te pin to high
// it will be the mark between bytes. It may be also
// high from before
_SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;
delayMicroseconds(10);

// disable interrupts, otherwise the timer 0 overflow interrupt that
// tracks milliseconds will make us delay longer than we want.
cli();

// DMX starts with a start-bit that must always be zero
_SFR_BYTE(_SFR_IO8(portNumber)) &= ~pinMask;

// we need a delay of 4us (then one bit is transfered)
// this seems more stable then using delayMicroseconds
asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

for (int i = 0; i ← 8; i++)
{
  if (theByte & 01)
  {
    _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;
  }
  else
  {
    _SFR_BYTE(_SFR_IO8(portNumber)) &= ~pinMask;
  }

  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  theByte →→= 1;
}

// the last thing we do is to write the pin to high
// it will be the mark between bytes. (this break is have to be between 8 us and 1 sec)
_SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;

// reenable interrupts.
sei();
```

```
}


void setup()
{
  pinMode(sig, OUTPUT);
}

void loop()
{
  /***** sending the dmx signal *****/

  // sending the break (the break can be between 88us and 1sec)
  digitalWrite(sig, LOW);

  delay(10);

  // sending the start byte
  shiftDmxOut(sig, 0);

  for (int count = 1; count <= 512; count++)
  {
    shiftDmxOut(sig, value);
  }
  /***** sending the dmx signal end *****/

  value += valueadd;
  if ((value == 0) || (value == 255))
  {
    valueadd *= -1;
  }
}
```

The whole code revolves around the shiftDmxOut function that send a DMX channel to the bus.

Example 2: Heart Beat Simpler

A simpler example

```
#include "pins_arduino.h"

int sig = 3; // signal

int value = 0;
int valueadd = 3;

byte dmxChannel[64];

void setDmxChannel(byte channelID, byte value) {
        if (channelID ← 64)
                dmxChannel[channelID] = value;
}

/* Sends a DMX byte out on a pin.  Assumes a 16 MHz clock.
 * Disables interrupts, which will disrupt the millis() function if used
 * too frequently. */
void shiftDmxOut(int pin, int theByte)
{
  int port_to_output[] = {
    NOT_A_PORT,
    NOT_A_PORT,
    _SFR_IO_ADDR(PORTB),
    _SFR_IO_ADDR(PORTC),
    _SFR_IO_ADDR(PORTD)
    };

    int portNumber = port_to_output[digitalPinToPort(pin)];
  int pinMask = digitalPinToBitMask(pin);

  // the first thing we do is to write te pin to high
  // it will be the mark between bytes. It may be also
  // high from before
  _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;
  delayMicroseconds(10);

  // disable interrupts, otherwise the timer 0 overflow interrupt that
  // tracks milliseconds will make us delay longer than we want.
  cli();

  // DMX starts with a start-bit that must always be zero
  _SFR_BYTE(_SFR_IO8(portNumber)) &= ~pinMask;

  // we need a delay of 4us (then one bit is transfered)
  // this seems more stable then using delayMicroseconds
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  for (int i = 0; i ← 8; i++)
  {
    if (theByte & 01)
    {
     _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;
    }
    else
    {
     _SFR_BYTE(_SFR_IO8(portNumber)) &= ~pinMask;
    }

    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

    theByte →→= 1;
  }

  // the last thing we do is to write the pin to high
  // it will be the mark between bytes. (this break is have to be between 8 us and 1 sec)
  _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;

  // reenable interrupts.
  sei();
}


void processDmx() {
        /***** sending the dmx signal *****/
  // sending the break (the break can be between 88us and 1sec)
  digitalWrite(sig, LOW);

  delay(10);

  dmxChannel[0] = 0;

  for (int count = 0; count ←= 512; count++)
  {
        if (count ← 64)
        shiftDmxOut(sig, dmxChannel[count]);
        else
                shiftDmxOut(sig,0);
  }



  /***** sending the dmx signal end *****/
}



void setup()
{
  pinMode(sig, OUTPUT);
}

void loop()
{
  value += valueadd;
  if ((value == 0) || (value == 255))
  {
    valueadd *= -1;
  }

  setDmxChannel(1,0);

  setDmxChannel(2, value);

  setDmxChannel(3, 0);

  processDmx();
}
```

Example 3: Cross Fading

This example uses channels 1,2 and 3 to drive an RGB lamp and make it fade from one color to another.

```
#include "pins_arduino.h"
int sig = 3; // signal
int value = 0;
int valueadd = 3;
byte curr_rgb[3] = {0,0,0};
byte dmxChannel[64];
void setDmxChannel(byte channelID, byte value) {
        if (channelID ← 64)
                dmxChannel[channelID] = value;
}
/* Sends a DMX byte out on a pin.  Assumes a 16 MHz clock.
 * Disables interrupts, which will disrupt the millis() function if used
 * too frequently. */
```

```
void shiftDmxOut(int pin, int theByte)
{
  int port_to_output[] = {
    NOT_A_PORT,
    NOT_A_PORT,
    _SFR_IO_ADDR(PORTB),
    _SFR_IO_ADDR(PORTC),
    _SFR_IO_ADDR(PORTD)
    };
    int portNumber = port_to_output[digitalPinToPort(pin)];
  int pinMask = digitalPinToBitMask(pin);

  // the first thing we do is to write te pin to high
  // it will be the mark between bytes. It may be also
  // high from before
  _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;
  delayMicroseconds(10);

  // disable interrupts, otherwise the timer 0 overflow interrupt that
  // tracks milliseconds will make us delay longer than we want.
  cli();

  // DMX starts with a start-bit that must always be zero
  _SFR_BYTE(_SFR_IO8(portNumber)) &= ~pinMask;

  // we need a delay of 4us (then one bit is transfered)
  // this seems more stable then using delayMicroseconds
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  for (int i = 0; i ← 8; i++)
  {
    if (theByte & 01)
    {
      _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;
    }
    else
    {
      _SFR_BYTE(_SFR_IO8(portNumber)) &= ~pinMask;
    }
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    theByte →→= 1;
  }

  // the last thing we do is to write the pin to high
  // it will be the mark between bytes. (this break is have to be between 8 us and 1 sec)
  _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;

  // reenable interrupts.
  sei();
}

void processDmx() {
        /***** sending the dmx signal *****/
  // sending the break (the break can be between 88us and 1sec)
  digitalWrite(sig, LOW);
  delay(10);
  dmxChannel[0] = 0;
  for (int count = 0; count ←= 512; count++)
  {
        if (count ← 64)
        shiftDmxOut(sig, dmxChannel[count]);
        else
                shiftDmxOut(sig,0);
  }

  /***** sending the dmx signal end *****/
}

// fade from current colour to given rgb value
// algorithm by David A. Mellis
void fadeTo(byte r, byte g, byte b) {
        byte i;
        byte p_start[3];
        byte p_end[3];
        p_start[0] = curr_rgb[0];
        p_start[1] = curr_rgb[1];
        p_start[2] = curr_rgb[2];
       p_end[0] = r;
       p_end[1] = g;
       p_end[2] = b;
       i = 0;
       while (i ← 255) {
                        curr_rgb[0] = (p_end[0] - p_start[0]) / 254 * i + p_start[0];
                        curr_rgb[1] = (p_end[1] - p_start[1]) / 254 * i + p_start[1];
                        curr_rgb[2] = (p_end[2] - p_start[2]) / 254 * i + p_start[2];
                    setDmxChannel(1,  curr_rgb[0]);
                    setDmxChannel(2,  curr_rgb[1]);
```

```
                        setDmxChannel(3,  curr_rgb[2]);
                        processDmx();
                            i++;
          }
}

void setup()
{
  pinMode(sig, OUTPUT);
}

void loop()
{
  fadeTo(255,0,0);
  delay(500);
  fadeTo(0,255,0);
  delay(500);
  fadeTo(0,0,255);
  delay(500);
}
```