Analysis of existing circuit

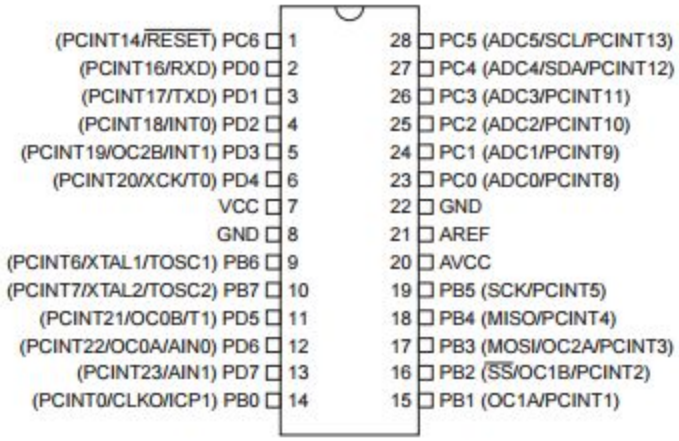
********** TODO **********

- Check what is the frequency update of each digit : use the core with resistor. Every sec, get the core to publish event with measured frequency
- Check what are the signals connected to the Atmega and how they are used
- See if it is possible to add a UART to the Atmega and replace it with Arduino chip
- Measure the voltage of one digit, see if it getting 5V or less. Can it be connected to 3.3v??
- Draw global schematic
- Write code see http://www.electroschematics.com/10512/arduino-4-digit-7-segment-led-display/
-

****

- ★ Power Input
  - 12Vdc +/-25%
  - Regulated with KA7805 on Power Supply Board 2
  - 5V reach Display Board on CON1
- ★ Display Board
  - Front
    - INPUT: information from the wheel, used to measure the KM
    - CON1: 5V+GND, see 1)
    - Buzzer is soldered
    - DP: External Push Button
    - MICRO: Switch activated by rotating the “For Hire” handle
    - ATMEL ATMEGA48PA-PU with quartz 4Mhz
    - It seems there is one I2C connector not soldered (maks SDA/SCL)
  - Back
    - The numeric displayed are soldered directly on the back. They all have a comma, to see if it is connected
    - There are:
      - One line with 5 big digits (E10561-G)
      - One line with 4 big digits (E10561-G)
      - One line with 4 small digits (E10391-G)
    - The displays are controlled by a chip 74HCT154N
    - There are 13 transistors CTBC 557B. These are NPN between the 74HCT154N and each digit http://www.farnell.com/datasheets/296678.pdf through a resitor
- ★ ATMEL ATMEGA48PA-PU chip
  - Seems to be very low ram
  - Using with Arduino http://www.thinkcreate.org/index.php/arduino-porting-to-atmega48/

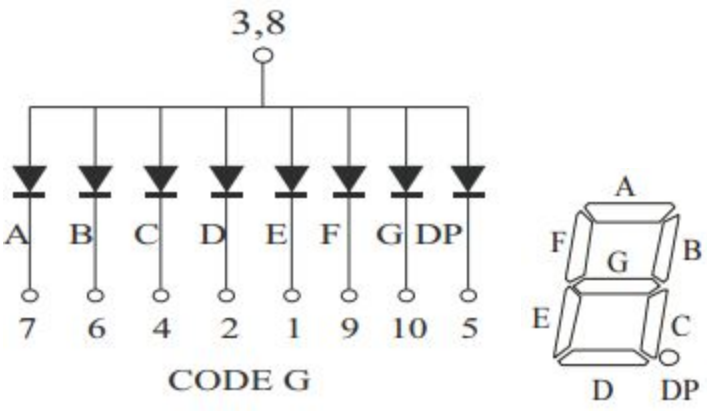
- Spec: http://www.atmel.com/images/doc8161.pdf
- Spec2: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf
- Seems to be compatible with pin to pin with Arduino Atmega168 from Arduino duemilanove. See below



R18 - Vcc	1	28	R4 - segmF
Odometer Brown	2	27	R14 - segmE
(nothing)	3	26	R11 - segmD
74HCT pin 23 (A0)	4	25	R10 - segmC
74HCT pin 22 (A1)	5	24	R8 - segmB
74HCT pin 21 (A2)	6	23	R6 - segmA
Vcc	7	22	GND - R16 - Odometer Black
GND	8	21	Vcc
Quartz	9	20	Vcc - R18 - Odometer Red
Quartz	10	19	R27 - Transistor - Buzzer
74HCT pin 20 (A3)	11	18	R26 - DP external button
U1 pin 5	12	17	(nothing)
U1 pin 6	13	16	74HCT pin 18+19 (EN)
RB25 - MICRO Handle	14	15	R3 - segmG

- ★ 74HCT154N chip
  - Spec http://www.nxp.com/documents/data_sheet/74HC_HCT154.pdf
  - This chip is a MUX 4->16
  - It is probably used to enable individually each of the 13 digits
  - Vcc min is 4.5V. Cannot work under 3.3V. However similar 74HC154N can...
- ★ E10xx1-G Digits
  - Spec http://www.py2bbs.qsl.br/projetos/freq_dl4yhf/TOYO-E10561.pdf
  - They are Code G, so common anode. So they are active at high level
  - The pin 8 is connected to the output of the PNP transistor, which is coming from the mux. If the pin 8 is down, display is off.
  - All digits of a row are using the digit selection signal. There are 7 signals commanded by resistors
    - First line of digits

Resistor	R3	R4	R6	R8	R10	R11	R14
Pin	10	9	7	6	4	2	1
Segment	G	F	A	B	C	D	E

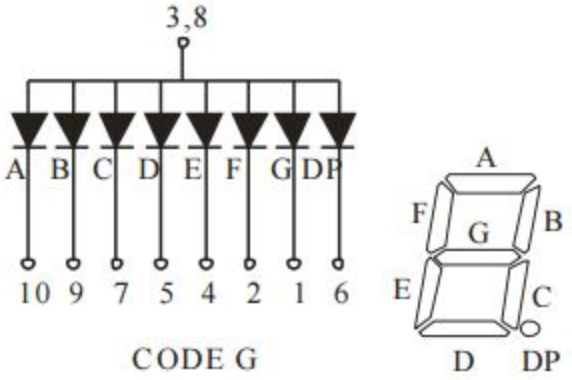


Note: pin 5 is not connected, this is the dot (DP). Also pin 3 is not connected

- Second line of digits
  - It seems the pins are connected to the same resistors as first line

- Third line of digits

Resistor	R3	R4	R6	R8	R10	R11	R14
Pin	1	2	10	9	7	5	4
Segment	G	F	A	B	C	D	E



Note: pin 6 is not connected, this is the dot (DP). Also pin 3 is connected to pin 8

★ Digit activation

- The digits are activated through PNP which are controlled by the 74HCT154

Output	Pin HCT	Resistor	Transistor		Output	Pin HCT	Resistor	Transistor
Y0	1	R13	Q5		Y8	9	R17	Q9
Y1	2	R5	Q4		Y9	10	R21	Q13
Y2	3	R7	Q3		Y10	11	R22	Q12
Y3	4	R9	Q2		Y11	13	R23	Q11
Y4	5	R12	Q1		Y12	14	R21	Q10
Y5	6	R15	Q6		Y13	15		
Y6	7	R20	Q7		Y14	16		
Y7	8	R19	Q8		Y15	17		

- The digits are installed like following:

Q1 Y4	Q2 Y3	Q3 Y2	Q4 Y1	Q5 Y0

Q7 Y8	Q8 Y7	Q9 Y6	Q6 Y5

Q10 Y12	Q11 Y11	Q12 Y10	Q13 Y9

★ Using Arduino

- Switch to 4Mhz quartz
  http://iot-playground.com/2-uncategorised/9-arduino-low-power-sensor
- Compatibility table:

PIN	ATMEGA48PA-PU PIN	ATMEGA 168	Arduino	Compatible?
1	R18 - Vcc	RESET		OK
2	Odometer Brown	PD0	0	OK
3	(nothing)	PD1	1	OK
4	74HCT pin 23 (A0)	PD2	2	OK
5	74HCT pin 22 (A1)	PD3	3	OK
6	74HCT pin 21 (A2)	PD4	4	OK
7	Vcc	Vcc		OK
8	GND	GND		OK
9	Quartz	Quartz		OK
10	Quartz	Quartz		OK
11	74HCT pin 20 (A3)	PD5	5	OK
12	U1 pin 5	PD6	6	OK
13	U1 pin 6	PD7	7	OK
14	RB25 - MICRO Handle	PB0	8	OK
15	R3 - segmG	PB1	9	OK
16	74HCT pin 18+19 (EN)	PB2	10	OK
17	(nothing)	PB3	11	OK
18	R26 - DP external button	PB4	12	OK
19	R27 - Transistor - Buzzer	PB5	13	OK
20	Vcc - R18 - Odometer Red	Vcc		OK
21	Vcc	Vcc		OK

22	GND - R16 - Odometer Black	GND		OK
23	R6 - segmA	PC0	A0(14)*	OK
24	R8 - segmB	PC1	A1(15)*	OK
25	R10 - segmC	PC2	A2(16)*	OK
26	R11 - segmD	PC3	A3(17)*	OK
27	R14 - segmE	PC4	A4(18)*	OK
28	R4 - segmF	PC5	A5(19)*	OK

*PortC is not named on the board, however it is named Ax in pins_arduino.h

★ Measuring 7segments refresh rate
- Method: by using Photon, measure the time between consecutive updates of one digit
- This can be done by measuring either:
  - when the signal EN to 74HCT is active. Assumption is that EN is toggled for every segment, which is not sure
  - when one particular 7segment is active: when it's transistor is switched ON, and for how long

★ Buzzer connection
- The buzzer is controlled by Atmega pin 19 (PB5) through a PNP transistor 5578 (BC 557). The transistor base is connected to PB5 through R27 (2.2K), the base is also connected to Vcc through R28 (100K).
- When PB5 is 5V, buzzer is OFF
- When PB5 is 0V, buzzer is ON
- When Arduino is starting, PB5 is 0V, so buzzer is ON.
- => Arduino shall be reprogrammed so that the default status is 5V