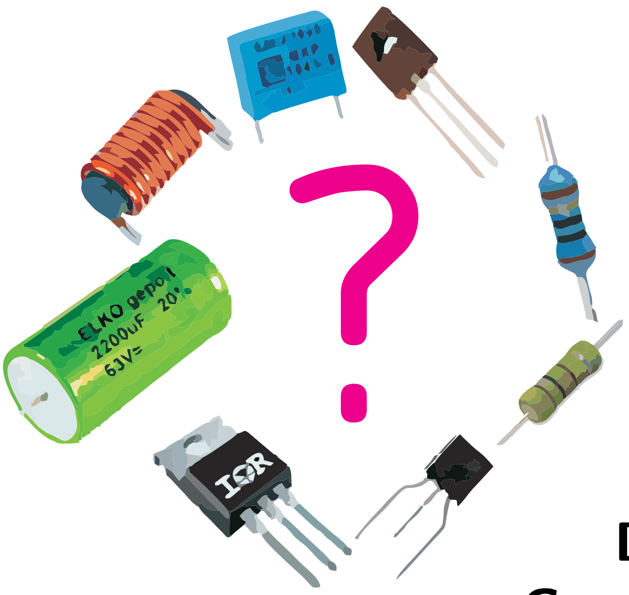
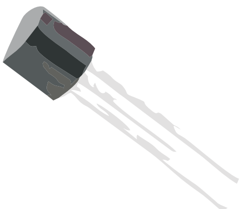


Description Component Tester A device for detecting and testing electronic components, supporting an optional computer interface

1 ⊣⊢ 3
329pF



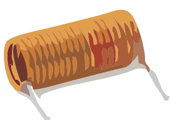
version 1.43m

BJT-PNP
hFE=226
Ie=1.6mA
Ube=673mV+

BJT-PNP
hFE=226
Ie=1.6mA
PIN 1=C 2=B 3=E

Markus Reschke

madires@theca-tabellaria.de

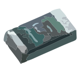


1 — 3
0.1Ω L=.05mH

Compiled
by bm-magic

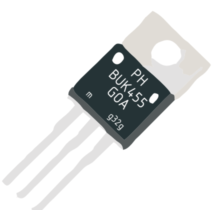
April 26, 2021

1 — 3
Uf=670mV
25pF-17pF @0-5V
Ir=.80µA



1 — 2
2205 Ω

1 — 3
10.08 kΩ



13.05.2019/MOR

N-E-MOS
Vt=3.1V
Cg=2.86nF
RDS=.2Ω +

N-E-MOS
Vt=3.1V
Cg=2.86nF
3 — 2 Uf=598mV+

N-E-MOS
Vt=3.1V
Cg=2.86nF
PIN 1=G 2=D 3=S

Contents

1 Features **7**

1.1 About . . . 7
1.2 Safety Advice . . . 7
1.3 License . . . 7
1.3.1 Additional Disclaimer . . . 7
1.4 What's different? . . . 8
1.5 Source Code . . . 8
1.6 Supported Hardware . . . 8
1.7 Following hardware options are supported . . . 9
1.7.1 user interface . . . 9
1.7.2 enhancements . . . 9
1.7.3 additional checks and measurements . . . 9
1.8 Building the firmware . . . 9

2 Hardware **10**

2.1 Hardware options . . . 10
2.2 And of course the software options . . . 10
2.3 Available UI languages . . . 11
2.4 configuration . . . 11
2.5 Busses & Interfaces . . . 11
2.5.1 I2C/SPI . . . 11
2.5.2 TTL Serial . . . 11
2.5.3 OneWire . . . 12
2.6 Displays . . . 12
2.6.1 HD44780 . . . 13
2.6.2 ILI9163 . . . 13
2.6.3 ILI9341/ILI9342 . . . 15
2.6.4 ILI9481 & ILI9486 . . . 16
2.6.5 ILI9488 . . . 16
2.6.6 PCD8544 . . . 16
2.6.7 PCF8814 . . . 17
2.6.8 SH1106 . . . 17
2.6.9 SSD1306 . . . 18
2.6.10 ST7036 (untested) . . . 18
2.6.11 ST7565R . . . 19
2.6.12 ST7735 . . . 19
2.6.13 ST7920 . . . 19
2.6.14 STE2007/HX1230 . . . 20
2.6.15 VT100 Terminal . . . 20
2.7 Test push button and other input options . . . 20
2.7.1 Test Key . . . 20
2.7.2 Rotary Encoder (hardware option) . . . 20
2.7.3 Increase/Decrease Buttons (hardware option) . . . 20

2.7.4 Touch Screen (hardware option) . . . 21
2.7.5 User Interface . . . 21
2.7.6 Communication with PC . . . 22
2.7.7 Serial Output . . . 22
2.7.8 Automation . . . 22
2.7.9 VT100 Output . . . 22

3 Operation **23**
3.0.1 Power-On . . . 23
3.0.2 Probing . . . 23
3.0.3 Battery Monitoring . . . 23
3.0.4 Power Off . . . 23
3.0.5 Menu . . . 23
3.1 Menu . . . 24
3.1.1 PWM Tool . . . 24
3.1.2 Simple PWM . . . 24
3.1.3 Fancy PWM . . . 24
3.1.4 Square Wave Signal Generator . . . 24
3.1.5 Zener Tool (hardware option) . . . 25
3.1.6 ESR Tool . . . 25
3.1.7 Capacitor Leakage Check . . . 25
3.1.8 R/C/L Monitor . . . 25
3.1.9 LC Meter . . . 25
3.1.10 Frequency Counter (hardware option) . . . 26
3.1.11 Basic Counter . . . 26
3.1.12 Extended Counter . . . 26
3.1.13 Event counter (hardware option) . . . 26
3.1.14 Rotary Encoder . . . 27
3.1.15 Contrast . . . 27
3.1.16 IR RC Detector/Decoder . . . 27
3.1.17 IR RC Transmitter . . . 29
3.1.18 Opto Coupler Tool . . . 30
3.1.19 Servo Check . . . 30
3.1.20 OneWire Scan . . . 31
3.1.21 DS18B20 Temperature Sensor . . . 31
3.1.22 DHTxx Sensors . . . 31
3.1.23 Self Test . . . 32
3.1.24 Self Adjustment . . . 32
3.1.25 Save/Load . . . 33
3.1.26 Show Values . . . 33
3.1.27 Font . . . 33
3.1.28 Power Off . . . 33
3.1.29 Exit . . . 33

4 Measuring **34**
4.1 Resistors . . . 34
4.2 Capacitors . . . 34
4.3 Inductors . . . 35
4.4 Discharging Components . . . 35
4.5 ADC Oversampling . . . 35
4.6 Displaying Results . . . 35
4.7 Additional Hints . . . 36
4.8 Helpful Links . . . 37
4.9 Change Log . . . 37

5 Remote Control 38
5.1 Remote Commands . . . 38
5.1.1 ERR . . . 38
5.1.2 OK . . . 38
5.1.3 N/A . . . 38
5.2 Basic Commands . . . 38
5.2.1 VER . . . 38
5.2.2 OFF . . . 38
5.3 Probing Commands . . . 38
5.3.1 PROBE . . . 38
5.3.2 COMP . . . 38
5.3.3 MSG . . . 38
5.3.4 QTY . . . 38
5.3.5 NEXT . . . 39
5.3.6 TYPE . . . 39
5.3.7 HINT . . . 39
5.3.8 MHINT . . . 39
5.3.9 PIN . . . 39
5.3.10 R . . . 40
5.3.11 C . . . 40
5.3.12 L . . . 40
5.3.13 ESR . . . 40
5.3.14 I_l . . . 40
5.3.15 V_F . . . 40
5.3.16 V_F2 . . . 40
5.3.17 C_D . . . 40
5.3.18 I_R . . . 40
5.3.19 R_BE . . . 40
5.3.20 h_FE . . . 40
5.3.21 h_FE_r . . . 40
5.3.22 I_C . . . 41
5.3.23 I_E . . . 41
5.3.24 V_BE . . . 41
5.3.25 I_CEO . . . 41
5.3.26 V_th . . . 41
5.3.27 C_GS . . . 41
5.3.28 R_DS . . . 41
5.3.29 V_GS_off . . . 41
5.3.30 I_DSS . . . 41
5.3.31 C_GE . . . 41
5.3.32 V_GT . . . 41
5.3.33 V_T . . . 42
5.3.34 R_BB . . . 42
5.3.35 V_Z . . . 42

6 Code 43
6.1 Makefile . . . 43
6.1.1 MCU model . . . 43
6.1.2 MCU freqency . . . 43
6.1.3 oscillator type . . . 43
6.1.4 Avrdude MCU . . . 44
6.1.5 Avrdude ISP Programmer . . . 44
6.2 Config.h . . . 45
6.2.1 Hardware options . . . 45
6.2.2 software options . . . 47

6.2.3 user interface 50
6.2.4 power management 52
6.2.5 measurement settings and offsets 53
6.2.6 R & D - meant for firmware developers 54
6.2.7 Busses 55
6.3 Config_328.h 56
6.3.1 LCD module 56
6.3.2 port and pin assignments 56
6.3.3 Busse 58
6.4 Config_644.h 59
6.4.1 LCD module 59
6.4.2 port and pin assignments 59
6.4.3 Busse 61
6.5 Config_1280 62
6.5.1 LCD module 62
6.5.2 port and pin assignments 62
6.5.3 Busse 64

7 Collection of settings **65**
7.1 Arduino Nano, Uno or Mega 2560 65
7.2 DIY Kit "'AY-AT"' or GM328A 65
7.3 BSide ESR02 67
7.4 Fish8840 TFT 68
7.5 GM328 !NOT GM328A! 68
7.6 Hiland M644 69
7.7 M12864 DIY Transistor Tester 70
7.8 MK-328 71
7.9 T3/T4 71
7.10 Multifunktionstester TC-1 and family (T7) 73
7.11 Arduino MEGA pinout 75

8 Programming the Component Tester **76**
8.1 Configure the Component Tester 76
8.2 Programming the microcontroller 76
8.3 Operation System Linux 76
8.4 Use under Linux 77
8.5 Install program packages 77
8.6 Download the sources 77
8.7 Using the interfaces 77
8.8 Group membership 78
8.9 working environment 78
8.10 Building the firmware 79

9 Changelog **80**
9.1 v1.43m 2021-03 80
9.2 v1.42m 2020-12 80
9.3 v1.41m 2020-09 81
9.4 v1.40m 2020-06 82
9.5 v1.39m 2020-03 82
9.6 v1.38m 2019-12 82
9.7 v1.37m 2019-09 83
9.8 v1.36m 2019-05 83
9.9 v1.35m 2019-02 83
9.10 v1.34m 2018-10 84
9.11 v1.33m 2018-05 84

9.12 v1.32m 2018-02 84
9.13 v1.31m 2017-12 85
9.14 v1.30m 2017-10 85
9.15 v1.29m 2017-07 85
9.16 v1.28m 2017-04 85
9.17 v1.27m 2017-02 85
9.18 v1.26m 2016-12 86
9.19 v1.25m 2016-09 86
9.20 v1.24m 2016-08 86
9.21 v1.23m 2016-07 86
9.22 v1.22m 2016-03 87
9.23 v1.21m 2016-01 87
9.24 v1.20m 2015-12 87
9.25 v1.19m 2015-11 87
9.26 v1.18m 2015-07 87
9.27 v1.17m 2015-02 88
9.28 v1.16m 2014-09 88
9.29 v1.15m 2014-09 88
9.30 v1.14m 2014-08 88
9.31 v1.13m 2014-07 88
9.32 v1.12m 2014-03 88
9.33 v1.11m 2014-03 88
9.34 v1.10m 2013-10 89
9.35 v1.09m 2013-07 89
9.36 v1.08m 2013-07 89
9.37 v1.07m 2013-06 89
9.38 v1.06m 2013-03 89
9.39 v1.05m 2012-11 90
9.40 v1.04m 2012-11 90
9.41 v1.03m 2012-11 90
9.42 v1.02m 2012-11 90
9.43 v1.01m 2012-10 90
9.44 v1.00m 2012-09 90
9.45 v0.99m 2012-09 90

Chapter 1 Features

1.1. About

The Component Tester is based on the project of Markus Frejek [1] & [2] and the successor of Karl-Heinz Kuebbeler [3] & [4]. It's an alternative firmware for Karl-Heinz' current Transistortester circuit and comes with several changes in the user interface and the methods used for probing and measuring. It also offers a few additional features. While Karl-Heinz provides an official release supporting also older ATmega MCUs, this firmware does require an ATmega with 32kB flash at least.

Hint: Run the self-adjustment for a new tester or if you've done any modifications, like a firmware update or changing probe leads.

1.2. Safety Advice

The Component Tester is no DMM!
It's a simple tester for components capable of measuring several things.
The probes aren't protected in any way and won't survive higher voltages than 5V.
Don't use the tester for live circuits!
Just use it for unsoldered electronic components!
If you test a capacitor make sure it's discharged before connecting the probes.

This isn't just the Safety Sally, your life may be at risk if you connect the probes to a live circuit or a power supply (or even mains).

1.3. License

The original author hasn't provided any information about the licence under which the firmware is distributed. He only stated that it's open source and any commercial user should contact him. Unfortunately we (Karl-Heinz and I) haven't found any way to contact him. To remedy this problem I've chosen an open source license at 2016-01-01, after giving the original author more than sufficient time to tell us his wishes regarding the license. Since the source code of this firmware version is a major rewrite with tons of new code and features, I think that this approach is justified.

Licensed under the EUPL V.1.1

1.3.1. Additional Disclaimer

Product or company names are possibly trademarks of the respective owners.

1.4. What's different?

Karl-Heinz has done a really great documentation of the tester. I recommend to read it. Therefore I'll tell you just about the major differences to the k-firmware:
- user interface
  + No worries! ;-)
  + touch screen
  + remote commands
- adaptive component discharging function
- resistance measurement
  + dedicated method for resistances < 10 Ohms (instead of using ESR check)
- capacitance measurement
  + starts at 5pF
  + additional method for caps from 4.7μF up to 47μF
  + correction/compensation method
- no SamplingADC() for very low capacitance or inductance
- diodes
  + detection logic
- BJTs
  + V_f is interpolated for a more suitable (virtual) I_b based on hFE
  + detection of Germanium BJTs with high leakage current
  + detection of Schottky-clamped BJTs
- JFETs
  + detection of JFETs with very low I_DSS
- TRIACs
  + detection of MT1 and MT2 - IR RC detector and decoder
- IR RC transmitter
- opto coupler check
- RC servo check
- OneWire (DS18B20)
- DHTxx Sensors
- Event counter
- structured source code
- some more I couldn't think of right now

There are more details in the sections below.

1.5. Source Code

The first m-firmware was based on Karl-Heinz' source code. A lot of cleaning up was done, like more remarks, renamed variables, re-structured functions, large functions splitted up into several smaller ones and what have you. After that the m-firmware moved on to become an independent version. For example, simple frameworks for displays and interface buses were added. I hope the code is easy to read and maintain.

You can download the lastest firmware from following sites:
- https://github.com/Mikrocontroller-net/transistortester/tree/master/Software/Markus [6]
- https://github.com/madires/Transistortester-Warehouse [7]

1.6. Supported Hardware

The firmware runs on all testers which are compatible with the standard circuit shown in Karl-Heinz' documentation and which use one of the following MCUs:

- ATmega 328	config_328.h
- ATmega 324/644/1284	config_644.h
- ATmega 640/1280/2560	config_1280.h

You can customize pin assigments if required. The display may be a character or graphic type (monochrome or color). Please see section 6.5.1 'Displays' on page 62 for supported controllers.

1.7. Following hardware options are supported

1.7.1. user interface

- rotary encoder
- additional push buttons (in/decrease)
- touch screen
- serial interface (TTL, RS232, USB-serial adpater)

1.7.2. enhancements

- external 2.5V voltage reference
- fixed adjustment cap
- protection relay for discharging caps

1.7.3. additional checks and measurements

- Zener check / measurement of external voltage <50V
- basic frequency counter
- extended frequency counter
with prescaler and crystal oscillators for low and high frequencies
- fixed IR RC receiver

1.8. Building the firmware

- First edit the **Makefile** list chapter 6.1 at page 43 specify your MCU model, frequency, oscillator type and programmer settings.

- Operation and menu options are set in **config.h** list chapter 6.2 at page 45.
Here choose hardware and software options, the language for for the UI, and change any default values if required.

- Finally in the **config_<MCU>.h** the global configuration, such as display and assignment of the pins, which differ depending on the MCU used. Details 6.5 at page 62.
the config<MCU>.h set in the files:

- ATmega 328	config_328.h
- ATmega 324/644/1284	config_644.h
- ATmega 640/1280/2560	config_1280.h

The chapter 7 'Clones' at page 65 lists settings for various tester versions/clones.
If you have a tester not listed, please email the settings to the author to help other users.

In config.h please choose hardware and software options, the language for the UI, and change any default values if required.

All settings and values are explained in the file, so I won't discuss them here in depth.

You can see **how to translate the firmware** in the section 8.10 on page 79.

Chapter 2 Hardware

2.1. Hardware options

- additional keys
  - rotary encoder
  - increase/decrease push buttons
  - touch screen
- 2.5V voltage reference
- relay based cap discharger
- Zener voltage measurement
- frequency counter (basic and extendend version)
- event counter with Trigger Output
- LC Meter
- IR detector/decoder for remote controls
  (fixed IR receiver displaye)
- fixed cap for self-adjustment of voltage offsets
- SPI bus (bit-bang and hardware)
- I2C bus (bit-bang and hardware)
- TTL Serial (bit-bang and hardware)
- OneWire bus (Bit-Bang and hardware)

The external 2.5V voltage reference should be only enabled if it's at least 10 times more precise than the voltage regulator. Otherwise it would make the results worse. If you're using a MCP1702 with a typical tolerance of 0.4% as voltage regulator you really don't need a 2.5V voltage reference.

2.2. And of course the software options

- PWM generator (2 variants)
- inductance measurement
- ESR measurement and in-circuit ESR measurement
- R/L/C Monitor
- check for rotary encoders
- squarewave signal generator (requires additional keys)
- IR detector/decoder for remote controls (IR receiver displaye connected to probes)
- IR RC transmitter (IR LED with driver trabsistor)
- check for opto couplers
- servo check (requires additional keys, display with > 2 lines)
- OneWire Scan
- detection of UJTs
- cap leakage check
- DS18B20 temperature sensor
- DHTxx temperature and humidity sensors
- color coding for probes (requires color graphics display)
- output of components found also via TTL serial, e.g. to a PC (requires TTL serial)
- remote commands for automation via TTL serial
- output of reverse hFE for BJTs
- ...

Please choose the options carefully to match your needs and the MCU's ressources, i.e. RAM, EEPROM and flash memory. If the firmware exceeds the MCU's flash size, try to disable some options you don't need.

2.3. Available UI languages

- Czech	- provided by Kapa (font based on ISO 8859-1)
- Czech_2	- provided by Bohu (font based on ISO 8859-2)
- Danish	- provided by glenndk@mikrocontroller.net (needs minor changes in the font)
- English	- (default)
- German	
- Italian	- privided by Gino_09@EEVblog
- Polish	- provided by Szpila (font based on ISO 8859-1)
- Polish_2	- provided by Jacon (font based on ISO 8859-2)
- Romanian	- provided by Dumidan@EEVblog
- Russian	-provided by indman@EEVblog -font with cyrillic characters based on Windows-1251
- Russian_2	-provided by hapless@@EEVblog -font with cyrillic characters based on Windows-1251
- Spanish	- provided by pepe10000@EEVblog

For number values a decimal fraction is indicated by a dot, but you can change that to a comma if you like by enabling the corresponding setting.

2.4. configuration

Your MCU is set in the Makefile file. . . Chapter 6.1 from page 43.
Your desired options in the file config.h.. . . Chapter 6.2 from page 45.
For the MCU-specific settings, such as pin assignments and display, you make the choice MCU in the config_MCU file:

- ATmega 328	config_328.h
- ATmega 324/644/1284	config_644.h
- ATmega 640/1280/1280	config_644.h

after submitting . . . Chapter 6.5 from page 62.
If the firmware gets too big, try less important options for yourself again deactivate.

2.5. Busses & Interfaces

2.5.1. I2C/SPI

Some displays and other hardware might need I2C or SPI for connecting to the MCU. Therefore the firmware includes drivers for both bus systems. To cope with different pin assignments of the various testers the bus drivers support bit-bang and hardware operation modes. The bit-bang mode can use any IO pins on the same port, while the hardware mode uses the dedicated bus pins of the MCU. The drawback of the bit-bang mode is its speed, it's slow. The hardware mode is much faster. You can spot the difference in speeds easily for high resolution color LCD displayes.

For ATmega 328 based testers the bit-bang mode is needed in most cases due to the circuit. The ATmega 324/644/1284 has more I/O pins and the different pin assignment for the circuit allows to use the dedicated bus pins for the hardware mode.

Since SPI or I2C are primarily used by the LCD displaye, they can be configured in the display section of config- <MCU>.h on page 62. Alternatively you can also enable I2C and SPI in config.h on page 55, and set ports and pins in dedicated sections in config- <MCU>.h (look for I2C_PORT on page 64 bzw. SPI_PORT) on page 64.
If you select bit-bang SPI and enable the read mode (SPI_RW) please make sure an example.

2.5.2. TTL Serial

The tester can also provide a TTL serial interface. In case it's used for communication with a PC it should be combined with a USB to TTL serial converter or a classic RS-232 driver. The firmware makes use of the MCU's hardware UART or a bit-bang software UART. The TTL serial interface is enabled in config.h on page 55 and the port pins are defined in config- <MCU>.h (look for SERIAL_PORT) on page 64.

The software UART has the drawback that the TX line will not stay high all the time when idle. This happens because of the way the MCU port pins are driven. To remedy this the port pin driving would have to be changed causing a larger firmware. But this issue doesn't seem to cause any trouble with most USB to TTL serial converters. In case you see any problem try to add a pull-up resistor (10-100k) to the TX pin to keep the signal at high level when idle.

The default setting for the TTL serial is 9600 8N1:
- 9600 bps
- 8 data bits
- no parity
- 1 stop bit
- no flow control

2.5.3. OneWire

Another supported bus is OneWire which can use either the probes/test pins (ONEWIRE_PROBES) or a dedicated I/O pin (ONEWIRE_IO_pin) on page 55. The driver is designed for default bus speed and clients to be powered externally (not parasitic-powered).

Pin assignment for probes:

Probe #1:	Gnd
Probe #2:	DQ (data)
Probe #3:	Vcc (current limited by 680 ohms resistor)

An external pull-up resistor of 4.7kOhms between DQ and Vcc is required!

2.6. Displays

At the moment following LCD controllers are supported:

- HD44780	(character display, 2-4 lines with 16-20 characters)	p.13
- ILI9163	(color graphic display 128x160)	p.13
- ILI9341/ILI9342	(color graphic display 240x320 or 320x240)	p.15
- ILI9481	(color graphic display 320x480, partly untested)	p.15
- ILI9486	(color graphic display 320x480, partly untested)	p.15
- ILI9488	(color graphic display 320x480, partly untested)	p.16
- PCD8544	(graphic display 84x48)	p.16
- PCF8814	(graphic display 96x65)	p.17
- SH1106	(graphic display 128x64)	p.17
- SSD1306	(graphic display 128x64)	p.18
- ST7036	(character display, 3 lines with 16 characters)	p.18
- ST7565R	(graphic display 128x64)	p.19
- ST7735	(color graphic display 128x160)	p.19
- ST7920	(graphic display up to 256x64)	p.19
- STE2007/HX1230	(graphic display 96x68)	p.21
- VT100 Terminal	(via serial interface)	p.20

Take care about the LCD's supply voltage and logic levels! Use a level shifter if required.
A simple level shifter with in-series resistors relying on the display controller's internal clamping diodes may work, but only for low speed busses like bit-bang SPI. Therefore I recommend to use proper level shifter ICs.

To save a few IO pins you can hardwire the /CS and /RES lines via pull-up/down resistors for nearly all displays and comment out the corresponding IO pins in the configuration (config_<MCU>.h) from page 62 as long as the display is the only device on the interface bus.

If the display doesn't show anything after double checking the wiring, please try different contrast settings (config_<MCU>.h) page 62.

For most displays you can hardwire the /CS and /RES lines via pull-up/down resistors and comment out the corresponding IO pins when the display is the only device on a bus.

Most graphic displays provide settings to change the image orientation, e.g. for rotating the image by 90° and mirroring the image horizontally or vertically. That way the image can be adjusted for each tester as needed. For color graphic displays additional settings are available. In the normal color mode the tester uses different colors which can be changed by editing the colors.h file. By commenting out LCD_COLOR the two-color mode is enabled and the pen color will be COLOR_PEN, while the background color is set to COLOR_BACKGROUND. In case the RGB base colors red and blue are reversed enable LCD_BGR to swap the red and blue color channels. Some displays have reversed RGB sub-pixels and the display controller doesn't know about that.

Hint for ATmega 328: If you connect a rotary encoder to PD2/PD3, please connect the displaye's /CS to PD5 and set LCD_CS in config_328.h (applies to graphic displays). Otherwise the rotary encoder would screw up the display by interfering with the data bus.

2.6.1. HD44780

The HD44780 is driven in 4 bit mode. The pin assignment for the parallel port is:

display	config-<MCU>.h	328 | 75.6%	644 | 37.9%	1280 | 13.1%	remark
DB4	LCD_DB4	PD0	PB4	PB0	
DB5	LCD_DB5	PD1	PB5	PB1	
DB6	LCD_DB6	PD2	PB6	PB2	
DB7	LCD_DB7	PD3	PB7	PB3	
RS	LCD_RS	PD4	PB2	PB4	
R/W	-	Gnd	Gnd	Gnd	
E	LCD_EN1	PD5	PB3	PB5	

Table 2.1. The pin assignment for the parallel port from HD44780.

You can also drive the LCD via a PCF8574 based I2C backpack which requires I2C to be enabled. The I2C address has to be specified too. The pin assignment defines how the LCD is connected to the PCF8574:

display	config-<MCU>.h	328 | 76.5%	644 | 38.2%	1280 | 9.8%	remark
DB4	LCD_DB4	PCF8574_P4			
DB5	LCD_DB5	PCF8574_P5			
DB6	LCD_DB6	PCF8574_P6			
DB7	LCD_DB7	PCF8574_P7			
RS	LCD_RS	PCF8574_P0			
R/W	LCD_RW	PCF8574_P1			
E	LCD_EN1	PCF8574_P2			
LED	LCD_LED	PCF8574_P3			

Table 2.2. The pin assignment for LCD is connected to the PCF8574.

2.6.2. ILI9163

The ILI9163 is driven by 4-wire SPI. The pin assignment is:

display	config-<MCU>.h	328 | 87.2%	644 | 43.8%	1280 | 11.2%	remark
/RESX	LCD_RES	PD4	PB2	PB4	optional
/CSK	LCD_CS	PD5	PB4	PB5	optional
D/CX	LCD_DC	PD3	PB3	PB7	
SCL	LCD_SCL	PD2	PB7	PB1	SPI clock
SDIO	LCD_SDA	PD1	PB5	PB2	SPI MOSI

Table 2.3. The pin assignment for ILI9163.

You might need to play with the x/y flip settings to get the correct orientation for your display. If necessary you can also offset the x direction. With LCD_LATE_ON enabled the tester starts with a cleared display causing a slight delay at power-on. Otherwise you'll see some random pixels for a moment.

2.6.3. ILI9341/ILI9342 is driven by SPIis driven by 4-line SPI or 8-bit parallel.

display	config-<MCU>.h	328 | 87.4%	644 | 51.7%	1280 | 13.2%	remark
RES	LCD_RES	PD4	PB2	PB4	optional
CS	LCD_CS	PD5	PB4	PB5	optional
DC	LCD_DC	PD3	PB3	PB7	
SCK	LCD_SCK	PD2	PD7	PB1	SPI clock
SDI	LCD_SDI	PD1	PB5	PB2	SPI MOSI
SDO	LCD_SDO	-	PB6	PB3	ILI9341 only,∗
∗ not used yet					

Table 2.4. The pin assignment 4-line SPI for ILI9341/ILI9342.

display	config-<MCU>.h	328	644	1280 | 13,1%	remark
	LCD_PORT			PORTB	
/RESX	LCD_RES			PB4	optional
CSX	LCD_CS			PB5	optional
D/CX	LCD_DC			PB7	
WRX	LCD_WR			PB0	
RDX	LCD_RD			PB6	
	LCD_PORT2			PORTL	
D0	LCD_DB0			PL0	LCD_PORT2 pin #0
D1	LCD_DB1			PL1	LCD_PORT2 pin #1
D2	LCD_DB2			PL2	LCD_PORT2 pin #2
D3	LCD_DB3			PL3	LCD_PORT2 pin #3
D4	LCD_DB4			PL4	LCD_PORT2 pin #4
D5	LCD_DB5			PL5	LCD_PORT2 pin #5
D6	LCD_DB6			PL6	LCD_PORT2 pin #6
D7	LCD_DB7			PL7	LCD_PORT2 pin #7

Table 2.5. The pin assignment 8-bit parallel for ILI9341/ILI9342.

You might need to play with the x/y flip and rotate settings to get the correct orientation for your display. And don't forget to set x and y dots based on the controller (ILI9341 is 240x320 and ILI9342 is 320x240). Some display modules disabled the ILI9341's extended command set (EXTC pin connected to Gnd). In that case you might see a blurry or ghostly output which can be fixed by enabling LCD_EXT_CMD_OFF.

Based on the relative high number of pixels the display output is somewhat slow via SPI. A complete screen clear takes about 3 seconds for bit-bang SPI and an 8MHz MCU clock. Better use harwdare SPI or the parallel bus.

2.6.4. ILI9481 & ILI9486

is driven by 8-bit parallel or 16-bit parallel.

Modul	config-<MCU>.h	328	644 | 51.6%	1280 | 13.2%	remark
	LCD_PORT			PORTB	
/RESX	LCD_RES		PC4	PB4	optional
/CSX	LCD_CS		PC3	PB5	optional
D/CX	LCD_DC		PC2	PB7	
WRX	LCD_WR		PC1	PB0	
RDX	LCD_RD		PC0	PB6	optional
	LCD_PORT2			PORTL	
D0	LCD_DB0		PB0	PL0	LCD_PORT2 pin #0
D1	LCD_DB1		PB1	PL1	LCD_PORT2 pin #1
D2	LCD_DB2		PB2	PL2	LCD_PORT2 pin #2
D3	LCD_DB3		PB3	PL3	LCD_PORT2 pin #3
D4	LCD_DB4		PB4	PL4	LCD_PORT2 pin #4
D5	LCD_DB5		PB5	PL5	LCD_PORT2 pin #5
D6	LCD_DB6		PB6	PL6	LCD_PORT2 pin #6
D7	LCD_DB7		PB7	PL7	LCD_PORT2 pin #7

Table 2.6. The pin assignment for ILI9481/ILI9486 8-Bit parallel.

Modul	config-<MCU>.h	328	644	1280	remark
LCD_D8	LCD_DB8			PC0	LCD_PORT3 pin #0
LCD_D9	LCD_DB9			PC1	LCD_PORT3 pin #1
LCD_D10	LCD_DB10			PC2	LCD_PORT3 pin #2
LCD_D11	LCD_DB11			PC3	LCD_PORT3 pin #3
LCD_D12	LCD_DB12			PC4	LCD_PORT3 pin #4
LCD_D13	LCD_DB13			PC5	LCD_PORT3 pin #5
LCD_D14	LCD_DB14			PC6	LCD_PORT3 pin #6
LCD_D15	LCD_DB15			PC7	LCD_PORT3 pin #7

Table 2.7. additionally the pin assignment for ILI9481/ILI9486 16-Bit parallel.

2.6.5. ILI9488

is driven by 8-bit parallel or 16-bit parallel. And it uses the same pin assignment as the ILI9481. Additionally, 4-line SPI is supported:

display	config-<MCU>.h	328 | 82.5%	644 | 39.1%	1280 | 10.0%	remark
RES	LCD_RES	PD4	PB2	PB4	optional
CS	LCD_CS	PD5	PB4	PB5	optional
D/C	LCD_DC	PD3	PB3	PB7	
SCL	LCD_SCL	PD2	PB7	PB1	SPI clock
SDA	LCD_SDA	PD1	PB5	PB2	SPI MOSI

Table 2.8. The pin assignment for ILI9488.

Because of the high resolution of the display and the RGB666 color schema (3 bytes per pixel) SPI is quite slow, even for hardware SPI and a 16 MHz MCU clock. So I wouldn't recommend to use the SPI interface.

2.6.6. PCD8544

The PCD8544 is driven by SPI. The pin assignment is:

display	config-<MCU>.h	328 | 82.5%	644 | 39.1%	1280 | 10.0%	remark
RES	LCD_RES	PD4	PB2	PB4	optional
SCE	LCD_SCE	PD5	PB4	PB5	optional
D/C	LCD_DC	PD3	PB3	PB7	
SCL	LCD_SCLK	PD2	PB7	PB1	SPI clock
SDIN	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Table 2.9. The pin assignment for PCD8544.

Since the display has just 84 pixels in x direction you'll get 14 chars per line with a 6x8 font. So up to two chars might be not displayed. To mitigate that you could shorten some texts in variables.h.

2.6.7. PCF8814

The PCF8814 is driven in the 3-wire SPI mode usually. The pin assignment for the 3-wire SPI (bit-bang only) is:

display	config-<MCU>.h	328 | 82.6%	644 | 41.3%	1280 | 10.6%	remark
/RES	LCD_RES	PD4	PB2	PB4	
/CS	LCD_CS	PD5	PB4	PB5	optional
SCLK	LCD_SCLK	PD2	PB7	PB1	SPI clock
SDIN	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Table 2.10. The pin assignment for PCD8814.

If necessary you can rotate the output via the y-flip setting and pulling the PCF8814's MX pin (x-flip) down or up.

2.6.8. SH1106

(partly untested) The SH1106 is driven by 3-wire SPI, 4-wire SPI or I2C. 3-wire SPI requires bit-bang mode and SPI_9 to be enabled.

display	config-<MCU>.h	328 | 83.0%	644 | 41.6%	1280 | 10.7%	remark
/RES	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
A0	LCD_DC	PD3	PB3	PB7	
SCL (D0)	LCD_SCL	PD2	PB7	PB1	SPI clock
SI (D1)	LCD_SI	PD1	PB5	PB2	SPI MOSI

Table 2.11. The pin assignment for SH1106 4-Line-SPI.

display	config-<MCU>.h	328 | 83.2%	644 | 41.6%	1280 | 10.7%	remark
/RES	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
A0		Gnd			
SCL (D0)	LCD_SCL	PD2	PB7	PB1	SPI clock
SI (D1)	LCD_SI	PD1	PB5	PB2	SPI MOSI

Table 2.12. The pin assignment for SH1106 3-Line-SPI (bit-bang only).

display	config-<MCU>.h	328 | 75,6%	644 | 37,9%	1280 | 13,1%	remark
/RES	LCD_RES	PD4	PB0	optional	
/CS		GND			
/SCL (D0)	I2C_SCL	PD1	PD0		
SDA (D1&2)	I2C_SDA	PD0	PD1		
SA0 (D/C)	Gnd (0x3c)			3.3V (0x3d)	

Table 2.13. The pin assignment for SH1106 I2C.

Using the x/y flip settings you can change the output orientation if neccessary. Many SH1106 based display modules need the x offset set to 2

2.6.9. SSD1306

The SSD1306 is driven by 3-wire SPI, 4-wire SPI or I2C. 3-wire SPI requires bit-bang mode and SPI_9 to be enabled. The pin assignment for 4-wire SPI is:

display	config-<MCU>.h	328 | 83.1%	644 | 41.6%	1280 | 10.7%	remark
/RES	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
DC	LCD_DC	PD3	PB3	PB7	
SCLK (D0)	LCD_SCLK	PD2	PB7	PB1	SPI clock
SDIN (D1)	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Table 2.14. The pin assignment for SSD1306 4-Line-SPI.

display	config-<MCU>.h	328 | 83.2%	644 | 41.6%	1280 | 10.7%	remark
/RES	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
SCLK (D0)	LCD_SCLK	PD2	PB7	PB1	SPI clock
SDIN (D1)	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Table 2.15. The pin assignment for SSD1306 3-Line-SPI (bit-bang only).

display	config-<MCU>.h	328 | 84.1%	644 | 42.0%	1280 | 10.7%	remark
RES	LCD_RES	PD4	PB2	PB0	optional
SCL (D0)	I2C_SCL	PD1	PC0	PD0	optional
SDA (D1&2)	I2C_SDA	PD0	PC1	PD1	PB0
SA0 (D/C)	Gnd (0x3c)				3.3V (0x3d)

Table 2.16. The pin assignment for SSD1306 I2C.

Using the x/y flip settings you can change the output orientation if neccessary.

2.6.10. ST7036 (untested)

The ST7036 is driven by a 4 bit parallel interface or 4-wire SPI.

display	config-<MCU>.h	328 | 76.2%	644 | 38.1%	1280 | 9.8%	remark
DB4	LCD_DB4	PD0	PB4	PB0	
DB5	LCD_DB5	PD1	PB5	PB1	
DB6	LCD_DB6	PD2	PB6	PB2	
DB7	LCD_DB7	PD3	PB7	PB3	
RS	LCD_RS	PD4	PB2	PB4	
R/W		Gnd	???	???	optional
E	LCD_EN	PD5	PB3	PB5	
XRESET		Vcc	???	???	optional

Table 2.17. The pin assignment for 4 bit parallel interface ST7036.

display	config-<MCU>.h	328 | 76.1%	644 | 38.1%	1280 | 9.8%	remark
XRESET	LCD_RESET	PD4	PB2	PB4	optional
CSB	LCD_CS	PD5	PB4	PB5	optional
RS	LCD_RS	PD3	PB3	PB7	
SCL (DB6)	LCD_SCL	PD2	PB7	PB1	SPI clock
SI (DB7)	LCD_SI	PD1	PB5	PB2	SPI MOSI

Table 2.18. The pin assignment for 4-wire SPI ST7036.

The ST7036i speaks I2C but isn't supported (yet). A special feature of the ST7036 is a dedicated pin to enable an extended instruction set (pin EXT) which is enabled usually. In case it's disabled the settings LCD_EXTENDED_CMD and LCD_CONTRAST need to be commented out.

2.6.11. ST7565R

The ST7565R is driven by 4/5 line SPI. The pin assignment is:

display	config-<MCU>.h	328 | 83.2%	644 | 41.7%	1280 | 10.1%	remark
/RES	LCD_RESET	PD0	PB2	PB4	optional
/CS1	LCD_CS	PD5	PB4	PB5	optional
A0	LCD_A0	PD1	PB3	PB7	
SCL (DB6)	LCD_SCL	PD2	PB7	PB1	SPI clock
SI (DB7)	LCD_SI	PD3	PB5	PB2	SPI MOSI

Table 2.19. The pin assignment for den 4/5-Line-SPI ST7565R.

You might need to play with the x/y flip and x-offset settings to get the correct orientation for your display.

2.6.12. ST7735

The ST7735 is driven by 4-wire SPI. The pin assignment is:

display	config-<MCU>.h	328 | 95.7%	644 | 47.9%	1280 | 12.2%	remark
/RESX	LCD_RES	PD4	PB2	PB4	optional
/CSK	LCD_CS	PD5	PB4	PB5	optional
D/CX	LCD_DC	PD3	PB3	PB7	
SCL	LCD_SCL	PD2	PB7	PB1	SPI clock
SDA	LCD_SDA	PD1	PB5	PB2	SPI MOSI

Table 2.20. The pin assignment for ST7735.

You might need to play with the x/y flip settings to get the correct orientation for your display. With LCD_LATE_ON enabled the tester starts with a cleared display causing a slight delay at power-on. Otherwise you'll see some random pixels for a moment.

2.6.13. ST7920

The ST7920 can be driven in 4 bit parallel mode or SPI.

display	config-<MCU>.h	328 | 83.5%	644 | xx.x%	1280 | xx.x%	remark
/XRESET	LCD_RES	PD4	PB2	PB4	optional
CS (RS)	LCD_CS	PD5	PB4	PB5	optional
SCLK (E)	LCD_SCLK	PD2	PB7	PB1	SPI clock
SID (RW)	LCD_SID	PD1	PB5	PB2	SPI MOSI

Table 2.21. The pin assignment for SPI ST7920.

display	config-<MCU>.h	328 | 83.8%	644 | 41.8%	1280 | 10.7%	remark
XRESET	LCD_RESET	Vcc	Vcc	Vcc	optional
E	LCD_EN	PD5	PB3	PB5	
RS	LCD_RS	PD4	PB2	PB4	
RW	LCD_RW	Gnd	Gnd	Gnd	
D4	LCD_DB4	PD0	PB4	PB0	
D5	LCD_DB5	PD1	PB5	PB1	
D6	LCD_DB6	PD2	PB6	PB2	
D7	LCD_DB7	PD3	PB7	PB3	

Table 2.22. The pin assignment for 4 bit parallel mode ST7920.

Because of the ST7920's poor design only fonts with a width of 8 pixels can be used. To cope with the horizontal 16 bit addressing grid I had to add a screen buffer for characters.

2.6.14. STE2007/HX1230

The STE2007 is driven in the 3-wire SPI mode usually (bit-bang only).

display	config-<MCU>.h	328 | 82.5%	644 | 41.3%	1280 | 10.6%	remark
/RES	LCD_RES	PD4	PB2	PB4	optional
/CS	LCD_CS	PD5	PB4	PB5	optional
SCLK	LCD_SCLK	PD2	PB7	PB1	SPI clock
SDIN	LCD_SDIN	PD1	PB5	PB2	SPI MOSI

Table 2.23. The pin assignment for 3-wireSPI ST7920.

If necessary you can rotate the output via the x/y flip settings.

2.6.15. VT100 Terminal

The VT100 driver replaces a LCD display and outputs everything to a VT100 serial terminal. The configuration section for VT100 includes already the activation of the TTL serial interface. Be aware that the VT100 driver will disable other options related to the serial interface which might interfere with the output.

2.7. Test push button and other input options

The tester's primary control is the test key, but additional input options are supported also for a more convenient operation, while some functions require those.

2.7.1. Test Key

The test key starts the tester and also **controls the user interface**. The tester distinguishes between:

1. **short key press**, usually used to continue a function or to select the next menu item,
2. **long key press** (> 0,3s), which performs a context-dependent action and
3. **Double click** who ends the action.

If the tester expects you to press a key it will tell you that by displaying a cursor at bottom right of the LCD. A steady cursor signals that more information will be displayed and a blinking cursor informs you that the tester will resume the probing loop. The cursor is supressed for menus and some tools, because it's obvious that a key press is neccessary.

Optionally you can enable key hints if your tester has additional keys and a display with a sufficient number of text lines (see UI_KEY_HINTS in config.h). A hint about the key usage is displayed instead of the cursor, if available. At the moment there's only one such hint for the probing (Menu/Test).

2.7.2. Rotary Encoder (hardware option)

With a rotary encoder you'll get some extra functionality with the user interface, but that's context specific. The additional functionality is described in the sections below, if applicable. Some functions make use of the encoder's turning velocity to allow larger changes or steps of values.

The algorithm for reading the encoder considers the number of Gray code pulses per step or detent (ENCODER_PULSES). Most rotary encoders have 2 or 4 Gray code pulses per detent. Also the number of steps or detents per complete 360° turn is taken into account (ENCODER_STEPS) on page 45. You can use that value to finetune the detection of the turning velocity to optimize the feedback. A higher value slows the velocity down, while a lower value speeds it up. In case the encoder's turning direction is reversed, simply swap the MCU pin definitions for A and B in <MCU>.h.

The detection of the turning velocity measures the time for two steps. So you need to turn the encoder at least by two steps for a mid-range velocity. For very high velocities it's three steps. A single step results in the lowest velocity.

2.7.3. Increase/Decrease Buttons (hardware option)

If you prefer push buttons over a rotary encoder you can add a pair of push buttons as alternative page 45. The push buttons are wired the same way as the rotary encoder (pull-up resistors, low active). For a speed-up functionality similar to the encoder's turning velocity keep pressing the push button.

A long button press will increase the "speed" as long as you keep pressing the button.

2.7.4. Touch Screen (hardware option)

Another input option is a touch screen. Please note that the screen should be large enough and support approximately 8 text lines with 16 characters each. To save precious space on the display the user interface doesn't show icons to touch. It simply has invisible touch bars at the left and right (each 3 characters wide), also at the top and the bottom (2 lines high) and one at the center area. The left and top bars are for decreasing a value or moving up in a menu, while the bottom and right bars are for increasing a value or moving down in a menu. Actually they do the same as a rotary encoder. Touching a bar longer results in a speed-up if supported by a function or tool (similar to turning the rotary encoder faster). The center bar acts as a software version of the test key, but it won't power the Zener diode test option for example.

The touch screen needs an adjustment for proper operation. The adjustment is automatically started after powering the tester on, when no adjustment values are stored in the EEPROM. You can also run the adjustment via the main menu. The procedure is straight forward. If you see an asterisk (yellow * on color displays), simply touch it. After that the tester clears the asterisk and displays the native x/y position. The first adjustment point is at the top right, and the second point at the bottom left. Based on the result the tester may try the adjustment up to three times. You can skip the procedure any time by pressing the test key.

If you have problems with the adjustment like bad x/y positions or an error after the first adjusment round, please check the orientation of the touch screen in relation to the display. The driver has options to flip or rotate the orientation. The display's top left is assumed to be the zero position. Some hints about the required settings for specific values of x and y:

* first adjustment point: top right.		
x	y	adjustemnt
low	low	TOUCH_FLIP_X
low	high	TOUCH_FLIP_X & TOUCH_FLIP_Y
high	low	keine
high	high	TOUCH_FLIP_Y
Don't forget to save the offsets, after a successful adjustemnt... (main menu: save).		

Table 2.24. adjustment point x/y.

Supported touch screen controllers: - ADS7843 / XPT2046

You'll find the configuration options below the display section in config_<MCU>.h (currently just config_644.h because of the lack of unused IO pins of the ATmega 328).

2.7.5. User Interface

There are several options to make the output for displays more fancy and easier to read.

For displays with more then two text lines:
- show key hints instead of cursor if available (UI_KEY_HINTS)

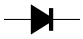
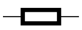
For graphic displays:
- all options listed before
- symbols incl. pinout for three-legged semiconductors (SW_SYMBOLS, not for low resolution diplays)

For color graphic displays:
- all options listed before
- color coding for probes/testpins (UI_PROBE_COLORS)
- dedicated color for titles (UI_COLORED_TITLES)
- dedicated color for cursor and key hints (UI_COLORED_CURSOR)
- color code for resistors (SW_R_E24_5_CC, SW_R_E24_1_CC and SW_R_E96_CC)

2.7.6. Communication with PC

The tester can support a TTL serial interface for communication with a PC. This could be a TX-only connection for outputting components found or a bidirectional one for automation. In both cases the TTL serial interface needs to be enabled in config.h (see section "Busses") page 55.
Special characters are replaced with standard ones, for example the Ω (Ohms) becomes a simple R.

conversion table:

	| > < |
	| |
	[ ]
Ω	R
μ	u

Hints:
- 9600 8N1
- newline is < CR > < LF >

2.7.7. Serial Output

The tester outputs components found to a PC running a simple terminal program when this feature is enabled (see UI_SERIAL_COPY in section "misc settings" in config.h) on page 50. The serial output follows the output on the LCD display but only for the components found. There is no serial output for menus and tools besides the results of the opto coupler check.

2.7.8. Automation

The automation feature allows you to control the tester by remote commands via a bidirectional serial connection. For enabling this feature please see UI_SERIAL_COMMANDS in section "misc settings" in config.h. The default behaviour of the tester will change slightly. The automation enforces the auto-hold mode and the tester won't automatically check for a component after powering on.

The command interface is fairly simply. You send a command and the tester will respond. The communication is based on ASCII textlines and the commands are case sensitive. Each command line has to be ended by a < CR > < LF >or <LF>newline. Be aware that the tester will only accept commands when waiting for user feedback after powering on, displaying a component or running a menu function. Response lines end with a < CR > < LF >newline. See section5 on page 38 for a list of commands and their explanation.

2.7.9. VT100 Output

The tester can output everything to a VT100 terminal instead of a LCD display (see VT100 in section "Displays" 2.6 on page 20). To keep the layout of the output undisturbed all other options for the serial interface are disabled.

Chapter 3 Operation

3.0.1. Power-On

A long key press while starting the tester selects the auto-hold mode. In that mode the tester waits for a short key press after displaying a result before it will continue. Otherwise the tester chooses the continuous (looping) mode by default. You can reverse the operation mode selection in config.h (UI_AUTOHOLD). After powering on, the firmware version is shown briefly.

A very long key press (2s) will reset the tester to firmware defaults. This might be handy if you have misadjusted the LCD contrast for example and can't read the display anymore.

If the tester detects a problem with the stored adjustments values, it will display a checksum error. That error indicates a corrupted EEPROM, and the tester will use firmware defaults instead.

For a tester with a manual power switch instead of the soft-latching one used by the reference design please enable POWER_SWITCH_MANUAL in config.c. In that case the tester won't be able to power itself off.

3.0.2. Probing

After the startup the tester looks for a connected component to check. In continuous mode it will automatically repeat the probing after a short pause. If no component is found for several times the tester will power itself off. In auto-hold mode (signaled by the cursor) the tester runs one probing cycle and waits for a key press or a right turn of the rotary encoder before it will proceed with the next cycle.

The cycle delay and automatic power-off for the continuous mode can be adjusted by changing CYCLE_DELAY page 47 and CYCLE_MAX in config.h page 50. There's an optional automatic power-off for the auto-hold mode (POWER_OFF_TIMEOUT) page 51 which is only active during probing cycles.

In both modes you can enter a menu with additional functions or power off the tester. For details please see below.

3.0.3. Battery Monitoring

Each probing cycling starts with the display of the battery voltage and a brief status (ok, weak, low). The tester will power off when the low voltage threshold is reached. The battery is checked regularly during operation page 52.

The default configuration for the battery monitoring is set for a 9V battery, but it can be changed for most other power sources. Please see section "power management" in config.h for all the settings.

The monitoring can be disabled by BAT_NONE, set to direct voltage check for power sources lower than 5V by BAT_DIRECT, or set for voltage check via a voltage divider, specified by BAT_R1 and BAT_R2, by BAT_DIVIDER. Some testers support an optional external power supply but don't allow its monitoring. In this case enable BAT_EXT_UNMONITORED to prevent problems with the automatic power-off by a low battery. This will also set the "ext" battery status when powered by the external power source.

The tresholds for a weak and a low battery are set by BAT_WEAK and BAT_LOW while BAT_OFFSET specifies any voltage drop caused by the circuit, e.g. a reverse polarity protection diode and a PNP power control transistor.

3.0.4. Power Off

While displaying the result of the last test a long key press powers the tester off. The tester will show a good bye message and then power off. As long as you press the key the tester stays powered on. This is caused by the implementaion of the power control circuit.

3.0.5. Menu

You'll enter the menu by two short key presses after the display of the last component found or function performed. Simply press the test key twice quickly (might need some practice :). If the rotary encoder option is enabled, a left turn will also enter the menu. The old method by short circuiting all three probes can be enabled too (UI_SHORT_CIRCUIT_MENU).

While in the menu, a short key press shows the next item in the menu and a long key press runs the shown item. On a 2-line display you'll see a navigation help at the bottom right. A ' > ' if another item follows, or a ' < ' for the very last item (will roll over to the first item). On a display with more than 2 lines the selected item is marked with an '*' at the left side.

With a rotary encoder you can move the items up or down based on the turning direction and a short key press will run the displayed item, instead of moving to the next item. Roll over is also enabled for the first item.

Some tools show you the pinout of the probe pins used before doing anything. That info will be displayed for a few seconds, but can be skipped by a short press of the test button.

For tools which create a signal probe #2 is used as output by default. In that case probe #1 and #3 are set to ground. If your tester is configured for a dedicated signal output (OC1B) the probes aren't used and no probe pinout will be displayed.

3.1. Menu

3.1.1. PWM Tool

This does what you would expect :) Before compiling the firmware please choose either the PWM generator with the simple user interface or the one with the fancy interface for testers with rotary encoder and large displays.

Pinout for signal output via probes:

Probe #2:	output (with (680 Ω resistor to limit current)
Probe #1 and #3:	Ground

3.1.2. Simple PWM

First you have to select the desired PWM frequency in a simple menu. Short key press for the next frequency and a long key press starts the PWM output for the shown frequency.

The duty ratio of the PWM starts at 50%. A short key press of the test button increases the ratio by 5%, a long key press decreases the ratio by 5%. To exit the PWM tool press the test key twice quickly.

If you have a rotary encoder you can use it to select the frequency in the menu and to change the PWM ratio in 1% steps.

3.1.3. Fancy PWM

Switch between frequency and ratio by pressing the test button. The selected value is marked by an asterisk. Turn the rotary encoder clockwise to increase the value or anti-clockwise to decrease it. As faster you turn the rotary encoder as larger the step size becomes. A long key press sets the default value (frequency: 1kHz, ratio: 50%).

Two short button presses exit the PWM tool.

3.1.4. Square Wave Signal Generator

The signal generator creates a square wave signal with variable frequency up to 1/4 of the MCU clock rate (2MHz for 8MHz MCU clock). The default frequency is 1000Hz and you can change it by turning the rotary encoder, The turning velocity determines the frequency change, i.e. slow turning results in small changes und fast turning in large changes. Since the signal generation is based on the MCU's internal PWM mode you can't select the frequency continuously, but in steps. For low frequencies the steps are quite small, but for high frequencies they become larger and larger. A long button press sets the frequency back to 1kHz, and two brief button presses exit the signal generator, as usual.

Pinout for signal output via probes:

Probe #2:	output (with (680 Ω resistor to limit current)
Probe #1 and #3:	Ground

Hint: Rotary encoder or other input option required!

3.1.5. Zener Tool (hardware option)

An onboard DC-DC boost converter creates a high test voltage for measuring the breakdown voltage of a Zener diode connected to dedicated probe pins. While the test button is pressed the boost converter runs and the tester displays the current voltage. After releasing the test button the minimum voltage measured is shown if the test button was pressed long enough for a stable test xvoltage. You may repeat this as long as you like. :) To exit the Zener tool press the test button twice quickly.

If your tester has just a 10:1 voltage divider without boost converter for measuring an external voltage or the boost converter runs all the time, you can choose the alternative mode (ZENER_UNSWITCHED) which measures the voltage periodically without pressing the test button. When you see the cursor at the bottom right between measurements you can exit the Zener tool by two short presses of the test button.
How to connect the Zener diode:

Probe +	cathode
Probe -	anode

3.1.6. ESR Tool

The ESR tool measures capacitors in-circuit and displays the capacity and ESR if the measurement detects a valid capacitor. Make sure that the capacitor is discharged before connecting the tester! Values could differ from the standard measurement (out-of-circuit) because any component in parallel with the capacitor will affect the measurement. For triggering a measurement please press the test key. Two quick short key presses will exit the tool.
How to connect the capacitor:

Probe #1:	positive
Probe #3:	negative (Gnd)

3.1.7. Capacitor Leakage Check

The cap leakage check charges a cap and displays the current and the voltage across the current shunt. It starts charging the cap using Rl (680 Ohms) and switches to Rh (470kOhms) when the current drops below a specific threshold.

Each cycle begins with the display of the pinout. After connecting the cap press the test button (or right turn in case of a rotary encoder) to start the charging process. To end charging press the test button again and the tester will discharge the cap while displaying its voltage until the voltage drops below the discharge threshold. To exit the check press the test button twice.
How to connect the capacitor:

Probe #1:	positive
Probe #3:	negative (Gnd)

Hint: Pay attention to the polarity of polarized caps!

3.1.8. R/C/L Monitor

The monitor tools measure continuously a passive component connected to probes #1 and #3. After starting a monitor tool the tester displays the probe pinout for a few seconds which can be skipped by pressing the test button.

There's a delay of one or two seconds between measurements, indicated by a cursor at the bottom right, during which you can exit the monitor by two short presses of the test button.

Available monitors:
- R Monitor (resistance)
- C Monitor (capacitance plus optionally ESR)
- L Monitor (inductance)
- R/C/L Monitor (R plus optionally L, or C plus optionally ESR)
- R/L Monitor (resistance plus optionally inductance)

3.1.9. LC Meter

The LC Meter hardware option is based on a simple LC oscillator circuit used by various inexpensive PIC LC meters. The common design (82μH and 1nF) has a base frequency of around 595 kHz, and connecting an additional capacitor or inductor will decrease that frequency. With the help of a reference capacitor with a known value, the measured frequencies and some math the value of the unknown capacitor/inductor can be calculated.

The PIC LC meters usually have measurement ranges of 10nH to 100mH, and 0.1pF to 900nF. They seem to use a gate time of 100ms for the frequency counter. The m-firmwmare uses autoranging with gate times of 100ms and 1000ms to improve the resolution for low value L/Cs. Thus the ranges start at about 1nH and 10fF (0.01pF). The maximum inductance supported is roughly 150mH. Regarding capacitance I've run into an issue with the circuit. At around 33nF the output signal starts having some spurs in the rising and falling edges causing the frequency counter to see more pulses than there really are. A user noted that this is a known problem of the LM311 based LC oscillator circuit. After trying a few modifications without ponderable success a second comparator or a Schmitt trigger logic gate seem to be the best choice to clean up the signal of the oscillator output. The CMOS quad NAND 4093 works fine for that. With a clean output signal the maximum capacitance is around 120nF (beyond that the LC oscillator becomes unstable). The tester enforces a lower frequency limit of 10kHz, i.e. the maximum values are theoretically 250nH or 3.5μF if the LC oscillator would run stable.

When starting the LC Meter the tester will run a self-adjustment indicated by an "adjusting..." message. After that you can connect the capacitor or inductor you like to check. A short button press switches between C and L measurement modes (default mode is C). The frequency of the LC oscillator drifts over time (up to 100 Hz) and needs a re-adjustment. If you see an increasing zero value or a "-" without any component connected please run the self-adjustment by a long key press. If there's any problem with the self-adjustment or you abort it by pressing some buttom the tester will exit the LC meter and report an error. Two short button presses will end the LC Meter tool.

Hints:
- The reference cap should be a low tolerance 1nF film cap. You can also use any common film cap around 1nF, measure its capacitance with a proper LCR meter, and update LC_METER_C_REF.
- If you're interested in the LC oscillator's frequency and its drifting enable LC_METER_SHOW_FREQ.

3.1.10. Frequency Counter (hardware option)

There are two versions of the frequency counter. The basic one is a simple |passive input for the T0 pin of the MCU. The extended version has an input buffer, two oscillators for testing crystals (low and high frequency) and an additional prescaler.
The circuit diagrams for both are depicted in Karl-Heinz' documentation. [5].

3.1.11. Basic Counter

With the basic frequency counter hardware option installed you can measure frequencies from about 10Hz up to 1/4 of the MCU clock with a resolution of 1Hz for frequencies below 10kHz. The frequency is measured and displayed continuously until you end the measurement by two short key presses. The autoranging algorithm selects a gate time between 10ms and 1000ms based on the frequency. The TO pin can be shared with a display.

3.1.12. Extended Counter

The extended frequency counter has an additional prescaler and allows to measure higher frequencies. The theoretical upper limit is 1/4 of the MCU's clock rate multiplied by the prescaler (16:1 or 32:1). The control lines are configured in config-<MCU>.h, and don't forget to set the correct prescaler in config.h.

The input channel (buffered input, low frequency crystal oscillator, high frequency crystal oscillator) is changed by pressing the test push button or turning the rotary encoder. And as always, two short button presses will exit the frequency counter.

3.1.13. Event counter (hardware option)

The event counter uses the T0 pin as dedicated input and is trigged by the rising edge of a signal. The T0 pin can't be shared with a display. Adding a simple input stage is recommended.

The counter is controlled by a small menu which also displays the counter values. Menu items are selected by a short key press and settings are changed by the rotary encoder or additional keys.

The first item is the counter mode:

- Counting	counting time and events
- time	count events for a given time
- events	count time for a given number of events

The second item "n" is the number of events. In the events mode it will show the trigger threshold which can be changed. A long key press resets the threshold to a default value (100). In other counting modes this item is blocked. The next item "t" is the time period in seconds. Same story, only for the time mode (default value: 60s). And the last item starts or stops counting by a long key press. When the counter runs the counted events and time elapsed are updated each second, and after stopping the results are displayed.

The limit for the time period is 43200s (12h) and for the events it's 4*10^9. If any of those limits is exceeded the counting is automatically stopped. The event limit or threshold (when in events mode) is checked every 200ms. Therefore some overshoot may occur in case of more than 5 events/s.

`-Trigger output` Optionally you can enable a trigger output (EVENT_COUNTER_TRIGGER_OUT) to control some other device using the probes. The trigger output is set high while counting, i.e. rising edge at start and falling edge at stop.
Pinout for trigger output via probes:

Probe #1:	Ground
Probe #2:	Output (with 680 Ω Ohms resistor to limit current)
Probe #3:	Ground

3.1.14. Rotary Encoder

This test checks rotary encoders while determining the pin-out. Your job is to connect the probes to the A, B and Common pin and to turn the encoder a few steps clockwise. The algorithm needs four greycode steps to determine the proper function and pin-out. The turning direction is important to distinguish the A nd B pins, because reversed pins cause a reversed direction.

When a rotary encoder is detected the tester will display the pin-out and wait for a key press (or a moment for continuous mode) before resuming testing.

To exit the rotary encoder test please press the test push button once while testing.

3.1.15. Contrast

You can adjust the contrast for some graphic LCD modules. A short key press increases the value and and a long key press decreases it. Two short key presses will exit the tool.

With a rotary encoder installed the value can also be adjusted by turning the encoder.

3.1.16. IR RC Detector/Decoder

This function detects and decodes signals from IR remote controls, and requires an IR receiver module, for example the TSOP series. When compiling the firmware you can choose between two variants how the IR receiver module is connected to the tester. The first one is to connect the IR module to the standard testpins. The second one is a fixed IR module connected to a dedicated MCU pin.

If a known protocol is detected the tester displays the protocol, address (when available), command, and in some cases optional data in hexadecimal.
The format is:
<protocol> <data fields>

For a malformend packet a "?" is shown as data field.

For a unknown protocol the tester displays the number of pauses and pulses, the duration of the first pulse and the first pause in units of 50 μs: ? <pulses> : <first pulse> - <first pause> .

When the number of pulses stay the same for different buttons of the RC, the modulation is most likely PDM or PWM. A changing number of pulses indicates bi-phase modulation.

To exit the tool please press the test key.

Supported protocols and their data fields:

- JVC
<address> : <command>

- Kaseikyo (Japancode, 48 Bit)

<manufacturer code> : <system> : <product>: <function>

- Matsushita (Panasonic MN6014, C6D6 / 11 bits)
<custom code> : <data code>

- Motorola
<command>

- NEC (standard & extended)
<address> : <command>
R for repeat sequence

- Proton / Mitsubishi (M50560)
<address> : <command>

- RC-5 (standard)
<address> : <command>

- RC-6 (standard)
<address> : <command>

- Samsung / Toshiba (32 bits)
<custom code> : <data code>

- Sharp / Denon
<address> : <command>

- Sony SIRC (12, 15 & 20 bits)
12 & 15: <command> : <address>
20: <command> : <address>: <extended>

Optional protocols (SW_IR_RX_EXTRA):
- IR60 (SDA2008/MC14497)
<command>

- Matsushita (Panasonic MN6014, C5D6 / 11 bits)
<custom code> : <data code>

- NEC μPD1986C
<data code>

- RECS80 (Standard & Erweitert)
<address> : <command>

- RCA
<address> : <command>

- Sanyo (LC7461)
<custom code> : <key>

- Thomson
<device> : <function>

The carrier frequency of the TSOP receiver module doesn't have to match the RC exactly. A mismatch reduces the possible range, but that doesn't matter much for this application.

- IR receiver module connected to probes.

Please connect the IR receiver module after entering the IR detector function.
How to connect the TSOP module:

Probe #1:	Gnd
Probe #2:	Vs (current limited by 680 Ωresistor)
Probe #3:	Data/Out

Hint: The current limiting resistor for Vs implies an IR receiver module with a supply voltage range of about 2.5 to 5V. If you have a 5V only module you can disable the resistor in the config.h file on your own risk. Any short circuit may destroy the MCU.

- Fixed IR receiver module

For the fixed IR module please set the port and pin used in config- <MCU>.h.

3.1.17. IR RC Transmitter The IR RC transmitter sends RC codes you've entered, and is meant to check IR RC receivers or remote controlled devices. This tool requires additional keys, such as a rotary encoder, a display with more than 4 lines, and a simple driver circuit for the IR LED.

The display shows you the protocol, the carrier frequency, the duty cycle of the carrier and a few data fields. By a short press of the test button you switch between the items. The selected item is indicated by an '*'. Use the rotary encoder (or other input option) to change the setting/value of an item. A long press of the test button and the tester sends the IR code as long as you keep the button pressed. And as usual, two short presses exit the tool.

When you change the protocol the carrier frequency and duty cycle are set to the protocol's default values. But you can change them if you wish. The carrier frequency can be set to 30 up to 56 kHz and the duty cycle to 1/2 (50%), 1/3 (33%) or 1/4 (25%). The data fields are the user settable parts of the IR code and are explained later on. In most cases it's just the address and the command.

Supported protocols and their data fields:
- JVC
<address:8> : <command:8>

- Kaseikyo (Japanese Code,)
<manufacturer code:16> <system:4> <product:8> <function:8>

- Matsushita (Panasonic MN6014 12 bits)
<custom code:6> <key data:6>

- Motorola
<command:9>

- NEC (standard)
<address:8> <command:8>

- NEC (extended)
<address:16> <command:8>

- Proton / Mitsubishi (M50560)
<address:8> <command:8>

- RC-5 (standard)
<address:5> <command:6>

- RC-6 (standard)
<address:8> <command:8>

- Samsung / Toshiba (32 bits)
<custom code:8> <key data:8>

- Sharp / Denon
<address:5> <command:8> <mask:1>

- Sony SIRC-12)
<command:7> <address:5>

- Sony SIRC-15)
<command:7> <address:8>

- Sony SIRC-20)
<command:7> <address:5> <extended:8>**Optional protocols** (SW_IR_RX_EXTRA):
- Thomson
<device:4> <function:7>

The data fields are separated by spaces and their syntax is:
`<field name> : <number of bits>`

Pinout for signal outpt via probes:

Probe #2:	signal output (with680 Ω resistor to limit current)
Probe #1 and #3:	Ground

The signal output (probe #2) has a current limiting resistor and can drive an IR LED with only about 5mA directly, which isn't sufficient for the IR LED's typical rating of 100mA. Therefore you need a simple driver circuit based on a switching transistor, the IR LED and a current limiting resistor for the LED.

Example for3.1 driving an IR LED (Vf 1.5V, If 100mA) with about 50mA:

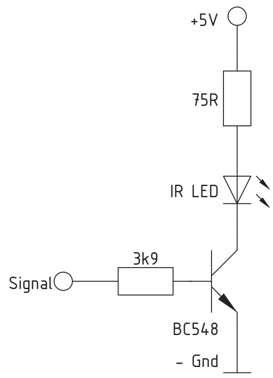


Figure 3.1. Example for 50mA driving an IR LED (Vf 1.5V, If 100mA,)

Hint: If the pulse/pause timing is incorrect please activate the alternative delay loop method SW_IR_TX_ALTDELAY page 47. This may be required when the C compiler optimizes the standard delay loop despite specific statements to keep the inline Assembler code.

3.1.18. Opto Coupler Tool

This tool checks opto couplers and shows you the LED's V_f, the CTR (also If), and t_on/t_off delays (BJT types). It supports standard NPN BJTs, NPN Darlington stages and TRIACs. For the CTR measurement the MCU's I/O pin is overloaded for about 3ms. The datasheet specifies a maximum output current of 20mA, but we overload the I/O pin up to about 100mA. Therefore the maximal CTR value is limited and any CTR over 2000% should be considered with caution.

The maximum drive current for the LED is about 5mA, which should be considered for TRIAC types. Relay types (MOSFET back to back) are detected as BJT and the CTR will be meaningless. Types with anti-parallel LEDs are ignored.

For testing you need a simple adapter with following three test points:

BJT type:

- LED's anode
- LED's cathode and BJT's emitter connected together
- BJT's collector

TRIAC type:

- LED's anode
- LED's cathode and TRIAC's MT1 connected together
- TRIAC's MT2

You may connect the adapter any way to the three probes of the component tester. It will detect the pinout automatically.

After entering the tool please connect the adapter and press the test key briefly to scan for an opto coupler. If one is found the tester displays the type and various details. Or it displays "none" when no opto coupler was detected. A blinking cursor indicates that you have to press the test key (or turn the rotary encoder) for a new scan. Two short key presses end the tool as usual.

3.1.19. Servo Check

This function outputs a PWM signal for RC servos which are driven by a 1-2ms PWM pulse. It supports the typical PWM frequencies of 50, 125, 250 and 333 Hz while the pulse length can be between 0.5 and 2.5 ms. There is also a sweep mode for sweeping between 1 and 2 ms pulse length with an adjustable sweep speed.

Please adjust the pulse width with the rotary encoder. Clockwise for a longer pulse, and counter-clockwise for a shorter pulse. A long button press resets the pulse to 1.5 ms.

You can switch between pulse and frequency selection mode with a short button press (mode marked by an asterisk). When in frequency selection mode use the rotary encoder to choose the PWM frequency. A long button press enables or disables the sweep mode (marked by a " <–> " after the frequency).

As long as the sweep mode is enabled, the pulse selection is replaced by the sweep period. The rotary encoder allows you to change the period.

As usual, two short button presses exit the function.

Pinout for signal output via probes:

Probe #2:	PWM output (with 680 Ω resistor to limit current)
Probe #1 and #3:	Ground

Hint: You have to provide an additional power supply for the servo.

Some pinouts of typical 3pin servo connectors:

Vendor	Pin 1	Pin 2	Pin3
Airtronics	PWM white/black	Gnd black	Vcc red
Futaba	PWM white	Vcc red	Gnd black
hitec	PWM yellow	Vcc red	Gnd black
JR Radios	PWM orange	Vcc red	Gnd brown

Table 3.1. Some pinouts of typical 3pin servo connectors

3.1.20. OneWire Scan

The scanning tool for OneWire lists the ROM codes of all connected devices. Please see section "Busses & Interfaces", on page 55, for the setup of the OneWire bus. When using the probes the tester will inform you about the pin assignment and waits until it detects the external pull-up resistor. You can skip this by a key press.

Each time you press the test button the tester will scan for the next device and display its ROM code (in hexadecimal). The first part of the output is the devices' family code and the the second part is its serial number. The CRC is omitted. A family code >= 0x80 (bit #7 set) indicates a customer specific code and the upper (left) three hexadecimal digits of the serial number are the customer's ID.

After the last device is found the tester will let you know. It will also inform you about CRC and bus errors. In case of a finished scan or bus error you can start a completely new scan process by pressing the test button. And as usual, two short button presses will exit the tool.

3.1.21. DS18B20 Temperature Sensor

This tool reads the OneWire temperature sensor DS18B20 and displays the temperature. Please see section "Busses & Interfaces", on page 55, for the setup of the OneWire bus. When using the probes the tester will inform you about the pin assignment and waits until it detects the external pull-up resistor. You can skip this by a key press.

After connecting the DS18B20 as the only client on the OneWire bus push the test button for reading the sensor (this may take nearly a second). To exit the tool press the test button twice quickly. And with a long button press you can select the auto mode (automatic updating) which is indicated by an "*" after the tool name.

Pin assignment for probes:

Probe #1:	Gnd
Probe #2:	DQ (data)
Probe #3:	Vcc (current limited by 680 ohms resistor)

An external pull-up resistor of 4.7kOhms between DQ and Vcc is required!

3.1.22. DHTxx Sensors

Tool for reading DHT11, DHT22 and compatible temperature & humidity sensors. First the tester displays the pinout and then waits for the external pull-up resistor. After that it shows the selected sensor type (default: DHT11) and a short press of the test button reads the sensor. On a successful read the tester outputs the measured values, on any error the result will be a "-". A long button press changes the sensor type, and two short button presses exit the tool. When changing the sensor you also have the option to activate automatic reading (each second) which is indicated by an "*" after the sensor name.

Supported sensors:

DHT11:	DHT11, RHT01
DHT22:	DHT22, RHT03, AM2302
	DHT21, RHT02, AM2301, HM2301
	DHT33, RHT04, AM2303
	DHT44, RHT05

Pin assignment for probes:

Probe #1:	Gnd
Probe #2:	Data
Probe #3:	Vdd (current not limited)

An external pull-up resistor of $4k7\ \Omega$ between Data and Vdd is required! Some sensor modules include already a $10k\ \Omega$ pull-up resistor which works also fine with short cables.

Hint: Because of the sensor's power demand the 680 Ω test resistor can't be used to limit current. Be aware that any short circuit may destroy the MCU.

3.1.23. Self Test

If you entered the self-test by the menu you'll be asked to short circuit all three probes and the tester will wait until you have. In case of any problem you can abort that by a key press. That will also skip the complete self-test.

The self-test function runs each test just 5 times. You can skip a test by a short key press or skip the entire selfttest by a long key press.

In test #4 you have to remove the short circuit of the probes. The tester will wait until you removed the short circuit.

The test steps are:
- T1 internal bandgap reference (in mV)
- T2 comparison of Rl resistors (offset in mV)
- T3 comparison of Rh resistors (offset in mV)
- T4 remove short circuit of probes
- T5 leakage check for probes in pull-down mode (voltage in mV)
- T6 leakage check for probes in pull-up mode (voltage in mV)

3.1.24. Self Adjustment

The self-adjustment measures the resistance and the capacitance of the probe leads, i.e. the PCB, internal wiring and probe leads as a sum, for creating a zero offset. It also measures the internal resistance of the MCU port pins in pull-down and pull-up mode. If the tests are skipped or strange values are measured the default values defined in config.h are used. If everything went fine the tester will display and use the new values gained by the self adjustment (they will be not stored automatically in the EEPROM, see "Save/Load" below).

The voltage offset of the analog comparator is automatically adjusted by the capacitance measurement (in normal probing mode, outside of the self adjustment) if the capacitor is in the range of 100 nF up to 3.3 μF. Also the offset of the internal bandgap reference is determined in the same way. Before running the self-adjustment the first time, please measure a film capacitor with a value between 100 nF and 3.3 μF three times at least to let the tester self-adjust the offsets mentioned above. Typically the first measurement will result in a slightly low value, the second in a high one and the third will be fine. This is caused by the self adjusting offsets. Both offsets are displayed at the end of the self-adjustment.

With a fixed cap for self-adjustment the automatic offset handling in the capacitance measurement is replaced by a dedicated function run during the self-adjustment procedure. So you don't have to measure a film cap before that.

In case the capacitance offsets vary across the probe pairs you can enable probe pair specific offsets in config.h (CAP_MULTIOFFSET). The same is possible for resistance offsets (R_MULTIOFFSET) page 53.

The self-adjustment is very similar to the self-test regarding the procedure and user interface. The adjustments steps are:
- A1 offsets for bandgap reference and analog comparator
  (only for fixed cap option)
- A2 resistance of probe leads/pins (in 10n Ω)
- A3 remove short circuit of probes
- A4 MCU's internal pin resistance for Gnd (voltage across RiL)
- A5 MCU's internal pin resistance for Vcc (voltage across RiH)
- A6 capacitance of probe leads/pins (in pF)

Limits:
- probe resistance <1.50 Ω for two probes in series
- probe capacitance <100 pF

Hint: When the resistance values for the probe leads/pins vary too much, there could be a contact problem.

Remember: Adjustment is not calibration! Calibration is the procedure to compare measurement results with a known traceable standard and noting the differences. The goal is to monitor the drift over time. Adjustment is the procedure to adjust a device to meet specific specs.

3.1.25. Save/Load

After running the self-adjustment the "Save" function will update the adjustment values stored in the MCU's EEPROM. The next time you power on the tester the updated values (profile #1) will be loaded and used automatically.
For your convenience you can save and load two profiles, e.g. if you have two different probe sets.

The idea of the save function is to prevent automatic saving of adjustment values. If you need to use other probe leads for some tests, you'd simply adjust the tester for the temporary probe leads and perform the tests. When you switch back to the standard probe leads you don't need to re-adjust the tester because the old values are still stored. Just powercycle the tester.

3.1.26. Show Values

This displays the current adjustment values and offsets used. The usage of an external 2.5V voltage reference is indicated by a '*' behind Vcc.

3.1.27. Font

Display of font for test purposes (SW_FONT_TEST

3.1.28. Power Off

This function will power off the tester if enabled by SW_POWER_OFF on page 51.

3.1.29. Exit

If you've entered the menu by mistake you can exit it by this command.

Chapter 4 Measuring

4.1. Resistors

Resistors are measured twice (both directions) and the values are compared. If the values differ too much the tester assumes that there are two resistors instead of just a single one. In that case the tester displays the result as two resistors with the same pins, like "1 – 2 – 1", and the two different resistance values. For resistors lower than 10 Ohms an extra measurement with a higher resolution is performed.

In some rare cases the tester might not be able to detect a very low resistance. If that happens simply re-run the test.

When the optional check for E series norm values (SW_R_E*) on page 49 is enabled the tester takes the next lower and next higher norm value and compares them with the measured resistance while also considering component tolerances. There are two output modes. In the text mode the tester displays the E series and the tolerance applied followed by matching norm values. A "-" indicates that no norm value matches. In the color-code output mode the tester displays the E series and the resistor's color code including the color band for tolerance. Be aware that colors can differ with the display module and used color combinations. If any color is off simply adjust the color value (COLOR_CODE_*) in the file colors.h. An Internet search for "RGB565 tool" will list many online tools for creating/picking RGB565 color values.

4.2. Capacitors

The measurement of capacitance is split into three methods. Large caps $>47\mu$F are measured by the charging cycle method with 10ms pulses. Mid-sized caps between 4.7μF and 47μF are processed the same way but with 1ms charging pulses. And small caps are done by the analog comparator method. That way the accuracy of the measurement of caps is optimized.

Large capacitances require a correction. Without correction the measured values are too large. IMHO, that is caused by the measurement method, i.e. the ADC conversion after each charging pulse needs some time and the cap looses charge due to internal leakage during the same time. Also the ADC conversion itself needs some charge to operate. So it takes longer to charge the cap, and the cap seems to have a larger capacitance. A discharge measurement later on tries to compensate this, but is able to do it just partially. The correction factors (CAP_FACTOR_SMALL, CAP_FACTOR_MID and CAP_FACTOR_LARGE in config.c page 53) are choosen to work with most tester models. In some cases you might have to change them.

A logic for preventing large caps to be detected as resistors was added. Resistors $<10\Omega$ are checked for being large caps.

A measured capacitance value more than 5pF (incl. the zero offset) is considered valid. Lower values are too uncertain and could be caused by placing the probe leads a little bit differently and things like that.

The tester tries to measure the ESR for capacitors larger than 10nF. Alternatively you can also enable the old ESR measurement method starting at 180nF. But since the ESR measurement isn't done via an AC signal with a specific frequency, please don't expect a solid result. The method used might be comparable with a 1kHz test. Anyway, the results are good enough to check electrolytic caps. For low value film caps you could get different results based on the MCU clock rate. I'd guess Mr. Fourier is able to explain this.

Alternatively you can also enable the old ESR measurement method.

Another measurement taken is the self-discharge leakage current for capacitors larger than 4.7μF. It gives a hint about the state of an electrolytic cap.

From my tests the typical value for a good electrolytic cap seems to be about:

- 10-220μF	1-3μA
- 330-470μF	4-5μA
- 470-820μF	4-7μA
- >1000μF	5-7μA per 1000μF

The optional check for E series norm values is also available for capacitors (SW_C_E*) page 49, but only in text mode because there are simply too many different color-codes for caps.

4.3. Inductors

The inductance measurement isn't very accurate, and things like the MCU clock speed and the PCB layout has an impact on the results. Basically it's based on measuring the time between switching on current flow and reaching a specific current. For high inductances there's a low current check, and for low inductances a high current check, which exceeds the MCU's pin drive limit for a very short time (up to about 25 micro seconds).

While investigating the effects of the MCU clock and other things I've found a pattern of deviations, which can be used for compensation. Based on the tester you have some custom tweaking might be necessary. In inductance.c in the function MeasureInductor() there the variable "Offset" for compensation. That variable is an offset for the reference voltage. A positive offset will decrease the inductance, and a negative value will increase the inductance.

The compensation for the high current check is based on the MCU clock, and its divided in three time ranges, each one with a dedicated offset. For the low current check there's just a simple compensation at the moment, as it needs further tests. If you see any major deviations when compairing the measurement results with a proper LCR meter, you can adjust the offsets to match your tester.

If you like to have the check for E series norm values please enable the SW_L_E* switches in config.h page 49 (text mode only).

Hints:

- When getting unexpected results please re-run the test.
- The inductance measurement is only performed when the inductors's resistance is lower than 2k Ohms.

4.4. Discharging Components

The tester tries to discharge any connected component before and while measuring. When it can't discharge the component below a specified threshold (CAP_DISCHARGED) it will output an error displaying the probe number and remaining voltage. In case of a battery the displayed voltage isn't the battery's voltage.

The discharge function isn't based on a fixed timeout, it adapts itself to the discharging rate. That way a battery will be identified faster (about 2s) and large caps have more time to discharge. If a large cap is identified as a battery please repeat the check. In a noisy environment you might need to adjust CAP_DISCHARGED to about 3mV. The remaining voltage displayed will help you to choose an appropriate value.

4.5. ADC Oversampling

The ADC function is modified to support a variable oversampling (1-255 times). The default value is 25 samples. You can try to improve the accuracy of the measurements by increasing the number of samples. Note that more samples will take more time resulting in slower measurements.

4.6. Displaying Results

Some names and abbreviations are changed. The output for several parts might be splitted into multiple pages to support displays with just a few lines.

For a single diode the low current Vf (measured with 10μA) is shown in braces if the voltage is below 250mV. That should give you a hint for germanium diodes. Most datasheets of germanium diodes specify Vf at 0.1mA which the tester doesn't support. At a higher current Vf is expected to be around 0.7V which makes it hard to distinguish germanium from silicon diodes.

The leakage current I_R for a single diode or I_CEO for a BJT will be displayed if it exeeds 50nA. Germanium BJTs have a leakage current of a few μA up to around 500μA. Germanium diodes are around a few μA usually.

For some components the capacitance is shown also. In case the capacitance is below 5pF or the measurement failed for some reason the value displayed will be 0pF.

If a depletion-mode FET with symmetrical Drain and Source is found, e.g. a JFET, the pinout shows an 'x' instead of a 'D' or 'S' because both can't be distinguished, they are functionally identical. Please see the FET's datasheet if you need more details about the pinout.

The pinout for a Triac is shown with the pin IDs 'G', '1' and '2'. '1' is MT1 and '2' is MT2. And for a UJT, in case the detection is enabled, it's '1' for B1, '2' for B2 and 'E' for the Emitter.

When the fancy pinout option is enabled (by selecting a symbols file in config.h) a component symbol with the corresponding probe pin numbers will be shown for 3-pin semiconductors.

If there's not enough space left on the display for the symbol, the pinout will be skipped.

4.7. Additional Hints

BJTs A lowercase letter following the hFE value indicates the test circuit type used for measuring hFE:
- e: common emitter circuit
- c: common collector circuit

If you have enabled the output of the hFE test current (SW_HFE_CURRENT) then the tester will display I_C for common emitter circuit and I_E for common collector circuit. When checking for diodes Vf is measured with Rl (high test current) and Rh (low test current), and both voltages are stored. The output function for BJTs looks up the matching diode for V_BE and interpolates the two Vf measurements based on the transistors hFE for a virtual test current. That way we get more suitable results for different kinds of transistors, since Vf of a small signal BJT isn't measured with the same test current as for a power BJT.
In case of a BJT with a base emitter resistor the tester displays that resistor. Be aware that the B-E resistor has an impact on V_BE and hFE. If the BJT also has a freewheeling diode the BJT might be detected as BJT or two diodes based on the value of the base emitter resistor (low value resistor -> 2 diodes). In the latter case the tester shows the two diodes and the resistor while hinting at a possible NPN or PNP BJT. Unfortunately the low value base emitter resistor prevents the correct detection of the BJT.
Another special case is a BJT with an integrated freewheeling diode on the same subtrate as the BJT. That integrated diode junction creates a parasitic transistor. A NPN BJT will have a parasitic PNP and vice versa. If such a BJT is found the tester shows a '+' behind the BJT type.
For a Schottky transistor (Schottky-clamped BJT) the clamping diode between base and collector and it's V_f are displayed if the detection is enabled (SW_SCHOTTKY_BJT) page 53. Note that a Schottky transistor has an increased I_CEO.

TRIACs can be used in three or four different operation modes, also known as quadrants. Usually some parameters will differ for each quadrant, like the gate trigger current (I_GT). In some cases it's possible that the tester's test current is sufficient to trigger the gate in one quadrant but not in another one. Since two test runs are needed to figure out the pins for MT1 and MT2 the tester won't be able to distinguish between them in those cases, i.e. the pins could be swapped.
You might also have TRIACs which can be triggered by the tester but have a too high holding current (I_H) preventing their correct detection. If a TRIAC's gate trigger current is too high the tester will detect just a resistor typically.

CLDs The diode check identifies a CLD (Current Limiting Diode) as a standard diode and displays I_F as the leakage current. Note that anode and cathode of a CLD are reversed vs. a standard diode. A dedicated check for a CLD is hard to implement, since the leakage current of a Germanium or high-current Schottky diode is in the range of I_F ($>33\mu$A). If a diode has an unusual forward voltage, a quite low V_f for the low current check (2nd value in braces) and no capacitance could be measured then it's most likely a CLD.

Unsupported Components Any semiconductor which requires a high current or high voltage to trigger conduction can't be detected, since the tester only provides about 7mA current and 5V voltage at maximum. So components like a DIAC with a V_BO of 20-200V can't be checked. Same for SCRs and TRIACs with a high trigger current.

Workarounds for some Testers
If your tester has one of the following issues you can try to enable a workaround:
- hFE way too high.

Problem:
Using the common collector circuit with Rl as base resistor the base voltage is memasured too low for some unknown reason. So the base current appears to be lower also, and causes a too high hFE value.
Affected testers:
Hiland M644
Workaround:
Enable NO_HFE_C_RL in config.h!

4.8. Helpful Links

- German forum
https://www.mikrocontroller.net/topic/248078 [10]
- English forum
https://www.eevblog.com/forum/testgear/$20-lcr-esr-transistor-checker-project/ [11]
- Russian forum
https://vrtp.ru/index.php?showtopic=16451 [12]
- Info on various clones, like images, schematics and firmwares
(by indman@EEVblog)
https://yadi.sk/d/yW8xa5NJgUo5z [13]
- Instruction guide and files for WinAVR (by indman@EEVblog)
https://drive.google.com/file/d/1-IJA8uTcsCA_6SYHEuMydjfS2vNgmwdH/edit [14]

4.9. Change Log

Please see the CHANGES file! Chapter 9 on page 80.

Chapter 5 Remote Control

5.1. Remote Commands

When the tester accepts remote commands it will respond with following text strings besides command specific answers containing data:

5.1.1. ERR

- unknown command
- command unsupported in component specific context
- buffer overflow

5.1.2. OK

- command executed

(some commands may need some time for processing)

5.1.3. N/A

- information/value not available

Responses with data will never start with any of the standard text strings above to prevent any possible confusion.

5.2. Basic Commands

5.2.1. VER - returns firmware version

- to verify connectivity and to determine command set based on version
- example response: "1.33m"

5.2.2. OFF

- powers off tester
- tester responds with an "OK" before powering off
- example response: "OK" < tester powers off >

5.3. Probing Commands

5.3.1. PROBE

- probes component and skips any pauses waiting for user feedback
- tester responds with an "OK" after probing is finished
- example response: < some time elapses for probing > "OK"

5.3.2. COMP

- returns component type ID
- see COMP_* in common.h for IDs
- example response for BJT: "30"

5.3.3. MSG

- returns error message
- applies only when an error has occured (COMP: 1)
- response may vary with the language of the user interface
- example response: "Battery? 1:1500mV"

5.3.4. QTY

- returns component quantity
- example response for BJT: "1"

5.3.5. NEXT

- selects second component
- applies if two components are found (QTY: 2)
- example response: "OK"

5.3.6. TYPE

- returns more specific type of component
- applies to BJT, FET and IGBT
- types available:

- NPN	NPN (BJT)
- PNP	PNP (BJT)
- JFET	JFET (FET)
- MOSFET	MOSFET (FET)
- N-ch	n-channel (FET, IGBT)
- P-ch	p-channel (FET, IGBT)
- enh.	enhancement mode (FET, IGBT)
- dep.	depletion mode (FET, IGBT)

- example response for BJT: "NPN"
- example response for FET (MOSFET): "MOSFET n-ch enh."

5.3.7. HINT

- returns hints on special features of a component
- applies to diode, BJT, FET and IGBT
- hints available:

- NPN	possibly a NPN BJT (diode)
- PNP	possibly a PNP BJT (diode)
- R_BE	base-emitter resistor (diode, BJT)
- BJT+	integrated flyback diode on same substrate creating a parasitic second BJT (BJT)
- D_FB	body/flyback diode (BJT, FET, IGBT)
- SYM	symmetrical drain and source (FET)

- example response for BJT: "D_FB R_BE"
- example response for FET (MOSFET): "D_FB"

5.3.8. MHINT

- returns hints on measurements
- applies to BJT
- hints avaiable:

- h_FE_e	h_FE measurement performed with common emitter circuit (BJT)
- h_FE_c	h_FE measurement performed with common collector circuit (BJT)

- example response for BJT: "h_FE_e"

5.3.9. PIN

- returns pinout of component
- identifiers used:

- resistor	x = connected,		- = not connected	
- capacitor	x = connected,		- = not connected	
- diode	A = anode,	C = cathode,	- = not connected	
- BJT	B = aase,	C = collektor,	E = emitter	
- FET	G = gate,	S = cource,	D = drain,	x = drain/source
- IGBT	G = gate,	C = collektor,	E = emitter	
- SCR	G = gate,	A = anode,	C = kathode	
- TRIAC	G = gate,	2 = MT2,	1 = MT1	
- PUT	G = gate,	A = anode,	C = kathode	
- UJT	E = emitter,	2 = B2,	1 = B1	

- Format der Antwort:
<probe #1 identifier > < probe #2 identifier > < probe #3 identifier >
- example response for resistor: "xx-"
- example response for diode: "C-A"
- example response for BJT: "EBC"

5.3.10. R
- returns resistance value
- applies to resistor (includes inductor)
- example response: "122R"

5.3.11. C
- returns capacitance value
- applies to capacitor
- example responses: "98nF" "462µF"

5.3.12. L
- returns inductance value
- applies to resistor (includes inductor)
- example response: "115µH"

5.3.13. ESR
- returns ESR value (Equivalent Series Resistance)
- requires ESR measurement to be enabled
- applies to capacitor
- example response: "0.21R"

5.3.14. I_l
- returns I_leak value (self-discharge equivalent leakage current)
- applies to capacitor
- example response: "3.25µA"

5.3.15. V_F
- returns V_F value (forward voltage)
- applies to diode and PUT
- also applies to body diode of MOSFET and flyback diode of BJT or IGBT
- example response: "654mV"

5.3.16. V_F2
- returns V_F value of low current measurement (forward voltage)
- applies to diode
- example response: "387mV"

5.3.17. C_D
- returns C_D value (diode capacitance)
- applies to diode
- example response: "8pF"

5.3.18. I_R
- returns I_R value (reverse current)
- applies to diode
- example response: "4.89µA"

5.3.19. R_BE
- returns R_BE value (base-emitter resistor)
- applies to diode and BJT
- example responses: "38.2R" "5171R"

5.3.20. h_FE
- returns h_FE value (DC current gain)
- applies to BJT
- example response: "234"

5.3.21. h_FE_r
- returns reverse h_FE value (collector and emitter reversed)

- applies to BJT
- example response: "23"

5.3.22. I_C

- returns I_C test current for hFE measurement
- requires output of test current for hFE measurement to be enabled
- for hFE measurement with common emitter circuit
- applies to BJT
- example response: "3245uA"

5.3.23. I_E

- returns I_E test current for hFE measurement
- requires output of test current for hFE measurement to be enabled
- for hFE measurement with common collector circuit
- applies to BJT
- example response: "3245uA"

5.3.24. V_BE

- returns V_BE value (base-emitter voltage)
- applies to BJT
- example response: "657mV"

5.3.25. I_CEO

- returns I_CEO value (collector-emitter current, open base)
- applies to BJT
- example response: "460.0μA"

5.3.26. V_th

- returns V_th value (threshold voltage)
- applies to FET (MOSFET) and IGBT
- example response: "2959mV"

5.3.27. C_GS

- returns C_GS value (gate-source capacitance)
- applies to FET (MOSFET)
- example response: "3200pF"

5.3.28. R_DS

- returns R_DS_on value (drain-source on-resistance)
- applies to FET (MOSFET)
- example response: "1.20R"

5.3.29. V_GS_off

- returns V_GS(off) value (cutoff voltage)
- applies to FET (depletion mode)
- example response: "-3072mV"

5.3.30. I_DSS

- returns R_DS_on value (drain-source on-resistance)
- applies to FET (MOSFET)
- example response: "1.20R" - returns I_DSS value (drain-source current, zero bias / shorted gate)
- applies to FET (depletion mode)
- example response: "6430μA"

5.3.31. C_GE

- returns C_GE value (gate-emitter capacitance)
- applies to IGBT
- example response: "724pF"

5.3.32. V_GT

- returns V_GT value (gate trigger voltage)
- applies to SCR and TRIAC
- example response: "865mV"

5.3.33. V_T
- returns V_T value (offset voltage)
- applies to PUT
- example response: "699mV"

5.3.34. R_BB
- returns R_BB value (interbase resistance)
- requires UJT detection to be enabled
- applies to UJT
- example response: "4758R"

5.3.35. V_Z
- returns V_Z value (Zener or external voltage)
- requires Zener check during normal probing to be enabled
- applies to Zener diode or external voltage (hardware option)
- example response: "6750mV"

Chapter 6 Code

As already mentioned, the firmware can be adapted for different testers and additional functions. There are some settings in the Makefile, in config.h and config<MCU>.h. This chapter explains the settings. The makefile controls the translation of the source code and contains basic things, such as the MCU types and ISP programmers. In the file confih.h there are general settings for operation and functions. And config<MCU>.h is responsible for things at the hardware level, so for LCD modules and the assignment of the pins.

6.1. Makefile

in the makefile the required parts are deselected by deselecting the # Symbols selected, or if a NAME is assigned in the option, entered after the = character.

6.1.1. MCU model

```
# avr-gcc: MCU model
# - ATmega 328/328P : atmega328
# - ATmega 328PB    : atmega328pb
# - ATmega 324P/324PA : atmega324p
# - ATmega 640      : atmega640
# - ATmega 644/644P/644PA : atmega644
# - ATmega 1280     : atmega1280
# - ATmega 1284/1284P : atmega1284
# - ATmega 2560     : atmega2560
MCU = atmega328
```

Listing 6.1. What is chosen is atmega328

6.1.2. MCU freqency

```
# MCU freqency:
# - 1MHz : 1
# - 8MHz : 8
# - 16MHz : 16
# - 20MHz : 20
FREQ = 8
```

Listing 6.2. 8MHz is selected

6.1.3. oscillator type

```
# oscillator type
# - internal RC oscillator : RC
# - external full swing crystal : Crystal
# - external low power crystal : LowPower
OSCILLATOR = Crystal
```

Listing 6.3. What is chosen is Crystal

6.1.4. Avrdude MCU

```
# avrdude: part number of MCU
# - ATmega 328 : m328
# - ATmega 328P : m328p
# - ATmega 328PB : m328pb
# - ATmega 324P : m324p
# - ATmega 324PA : m324pa
# - ATmega 640 : m640
# - ATmega 644 : m644
# - ATmega 644P : m644p
# - ATmega 644PA : m644p
# - ATmega 1280 : m1280
# - ATmega 1284 : m1284
# - ATmega 1284P : m1284p
# - ATmega 2560 : m2560
PARTNO = m328p
```

Listing 6.4. m328p is selected

6.1.5. Avrdude ISP Programmer

The programmer's necessary: Name; Port end BitClock.

```
# Arduino as ISP
#PROGRAMMER = stk500v1
#PORT = /dev/ttyACM0
#OPTIONS = -b 19200

# Bus Pirate
#PROGRAMMER = buspirate
#PORT = /dev/bus_pirate
#OPTIONS = -B 10.0

# Diamex ALL-AVR/AVR-Prog
PROGRAMMER = avrispmkII
PORT = usb
OPTIONS = -B 1.0

# Pololu USB AVR Programmer
#PROGRAMMER = stk500v2
#PORT = /dev/ttyACM0
#OPTIONS = -B 1.0

# USBasp
#PROGRAMMER = usbasp
#PORT = usb
#OPTIONS = -B 20

# USBtinyISP
#PROGRAMMER = usbtiny
#PORT = usb
#OPTIONS = -B 5.0

# Arduino Uno bootloader via serial/USB
#PROGRAMMER = arduino
#PORT = /dev/ttyACM0
#OPTIONS = -D -b 115200

# Arduino Mega2560 bootloader via serial/USB
#PROGRAMMER = wiring
#PORT = /dev/ttyACM0
#OPTIONS = -D -b 115200
```

Listing 6.5. Selectet is Diamex

If your programmer is not listed, you have to make up for it.
For further information please look to the manual pages of avrdude and online documentation [9].

6.2. Config.h

This file is for setting operation and functions. Since it is a normal C header file, the known commentary rules used for C. To make something active, delete the "//" at the beginning of the line. To the Deactivate inserts a "//" at the beginning of the line. Some settings require a numerical value, which may be adjust is.

6.2.1. Hardware options

rotary encoder for user interface

- default pins: PD2 & PD3 (ATmega 328)
- could be in parallel with LCD module
- see ENCODER_PORT for Pins (config-<MCU>.h)

//#define HW_ENCODER - uncomment to enable

Number of Gray code pulses per step or detent

- a rotary encoder's pulse is the complete sequence of 4 Gray code pulses
- adjust value to match your rotary encoder
- typical values: 2 or 4, rarely 1

#define ENCODER_PULSES . . . 4 - adjust value to match your rotary encoder

Number of detents or steps * - this is used by the detection of the rotary encoder's turning velocity

- it doesn't have to match exactly and also allows you to finetune the the feedback (higher: slow down, lower: speed up)
- typical values: 20, 24 or 30

#define ENCODER_STEPS . . . 24 - adjust value to match your rotary encoder

increase/decrease push buttons for user interface - alternative for rotary encoder

- see KEY_PORT for port pins (config-<MCU>.h)

//#define HW_INCDEC_KEYS - uncomment to enable

2.5V voltage reference for Vcc check - default pin: PC4 (ATmega 328)

- should be at least 10 times more precise than the voltage regulator
- see TP_REF for port pin (config-<MCU>.h)

and also adjust UREF_25 below for your voltage reference

#define HW_REF25 * - comment out to deactivate

Typical voltage of 2.5V voltage reference (in mV)

- see datasheet of the voltage reference
- or use >= 5.5 digit DMM to measure the voltage

#define UREF_25 . . . 2495 ** -If necessary change value

Probe protection relay for discharging caps

- default pin: PC4 (ATmega 328)
- low signal: short circuit probe pins

high signal via external reference: remove short circuit

//#define HW_DISCHARGE_RELAY - uncomment to enable

voltage measurement up to 50V DC / Zener check

- default pin: PC3 (ATmega 328)
- 10:1 voltage divider (standard: 10:1)
- DC-DC boost converter controled by test push button
- see TP_ZENER for port pin

#define HW_ZENER * - comment out to deactivate

non-standard voltage divider for Zener check

- standard voltage divider is 10:1
- ZENER_R1: top resistor in Ohms
- ZENER_R2: bottom resistor in Ohms

//#define ZENER_DIVIDER_CUSTOM - uncomment to enable
#define ZENER_R1 180000 * -If necessary change value
#define ZENER_R2 20000 * -If necessary change value

alternative mode for Zener check don't switch boost converter
- when the DC-DC boost converter runs all the time
- when measuring an external voltage (circuit without boost converter)
//#define ZENER_UNSWITCHED - uncomment out to enable

Zener check during normal probing
- requires boost converter running all the time (ZENER_UNSWITCHED)
- The min/max voltages are meant for the detection of a valid Zener voltage.
The min. voltage should be higher than the noise floor, while the max.
voltage should be lower than the boost converter's output voltage.
//#define HW_PROBE_ZENER - uncomment to enable
#define ZENER_VOLTAGE_MIN 1000 /* min. voltage in mV */ -If necessary change
#define ZENER_VOLTAGE_MAX 30000 /max. voltage in mV/ -If necessary change

fixed signal output - in case MCU's OC1B pin is wired as dedicated signal output
instead of driving Rl probe resistor for test pin #2
//#define HW_FIXED_SIGNAL_OUTPUT * - uncomment to enable

basic frequency counter
- default pin: T0 (PD4 ATmega 328)
- uses T0 directly as frequency input
- counts up to 1/4 of MCU clock rate
- might be in parallel with LCD module
//#define HW_FREQ_COUNTER_BASIC - uncomment to enable

extended frequency counter
- low and high frequency crystal oscillators
and buffered frequency input
- prescalers 1:1 and 16:1 (32:1)
- see COUNTER_PORT for port pins (config-<MCU>.h)
- requires a display with more than 2 text lines
- select the circuit's prescaler setting: either 16:1 or 32:1
#define HW_FREQ_COUNTER_EXT -Comment on deactivating
#define FREQ_COUNTER_PRESCALER . . . 16 /* 16:1 */ - prefix

event counter
- default pin: T0 (PD4 ATmega 328)
- uses T0 directly as event/pulse input (rising edge)
- no shared operation with displays possible for T0
- requires additional keys (e.g. rotary encoder) and a display with
more than 5 lines
- only for MCU clock of 8, 16 or 20MHz
//#define HW_EVENT_COUNTER - uncomment to enable

trigger output for event counter
- uses probe #2 as trigger output, probes #1 and #3 are Gnd
- sets trigger output to high while counting
//#define HW_EVENT_COUNTER_TRIGER_OUT - uncomment to enable

IR remote control detection/decoder (via dedicated MCU pin)
- requires IR receiver module, e.g. TSOP series
- module is connected to fixed I/O pin
- see IR_PORT for port pin (config-<MCU>.h)
- for additional protocols also enable SW_IR_RX_EXTRA
//#define HW_IR_RECEIVER - uncomment to enable

fixed cap for self-adjustment
* - see TP_CAP and ADJUST_PORT for port pins (config-<MCU>.h)
#define HW_ADJUST_CAP * - comment out to deactivate

L/C meter
- uses T0 directly as frequency input
- see LC_CTRL_PORT in config-<MCU>.h for port pins
//#define HW_LC_METER - uncomment to enable

L/C meter value of reference capacitor C_p (in 0.1 pF)
- should be around 1000pF
#define LC_METER_C_REF 10000 -Comment on deactivating

L/C meter also display frequency of LC oscillator
- helps to spot the oscillator's frequency drifting
- requires display with more than two text lines
//#define LC_METER_SHOW_FREQ - uncomment to enable

relay for parallel cap [sampling ADCC)
//#define HW_CAP_RELAY - uncomment to enable

6.2.2. software options

PWM generator with simple user interface
- signal output via OC1B
#define SW_PWM_SIMPLE -Comment on deactivating

PWM generator with fancy user interface
- signal output via OC1B
- requires additional keys and display with more than 2 text lines
//#define SW_PWM_PLUS - uncomment to enable

Inductance measurement
#define SW_INDUCTOR -Comment on deactivating

ESR measurement
- requires MCU clock >= 8 MHz
#define SW_ESR -Comment on deactivating
- choose SW_OLD_ESR for old method starting at 180nF
//#define SW_OLD_ESR - uncomment to enable

ESR Tool (in-circuit ESR measurement)
- requires SW_ESR or SW_OLD_ESR to be enabled
//#define SW_ESR_TOOL - uncomment to enable

check for rotary encoders
//#define SW_ENCODER - uncomment to enable

squarewave signal generator
- signal output via OC1B
- requires additional keys
#define SW_SQUAREWAVE -Comment on deactivating

IR remote control detection/decoder (via probes)
- requires IR receiver module, e.g. TSOP series
//#define SW_IR_RECEIVER - uncomment to enable

current limiting resistor for IR receiver module
- for 5V only modules
- Warning: any short circuit may destroy your MCU
//#define SW_IR_DISABLE_RESISTOR * - uncomment to enable

additional protocols for IR remote control detection/decoder
- uncommon protocols which will increase flash memory usage ;)
//#define SW_IR_RX_EXTRA - uncomment to enable

IR remote control sender
- signal output via OC1B
- requires additional keys and display with more than 4 text lines
- also requires an IR LED with a simple driver
//#define SW_IR_TRANSMITTER - uncomment to enable

Alternative delay loop for IR remote control sender for IR remote control sender
- in case the the C compiler screws up the default delay loop
and causes incorrect pulse/pause timings
//#define SW_IR_TX_ALTDELAY - uncomment to enable

additional protocols for IR remote control for IR remote control
- uncommon protocols which will increase flash memory usage ;)
//#define SW_IR_TX_EXTRA - uncomment to enable

check for opto couplers
//#define SW_OPTO_COUPLER - uncomment to enable

check for Unijunction Transistors
#define SW_UJT -Comment on deactivating
check for Schottky Transistor (Schottky-clamped BJT)
#define SW_SCHOTTKY_BJT -Comment on deactivating

Servo Check
- signal output via OC1B
- requires additional keys and display with more than 2 text lines
//#define SW_SERVO - uncomment to enable

DS18B20 - OneWire temperature sensor
- also enable ONEWIRE_PROBES or ONEWIRE_IO_PIN (see section 'Busses')
//#define SW_DS18B20 - uncomment to enable

OneWire: read and display ROM code
- option for OneWire related tools
- requires display with more than 2 text lines
//#define ONEWIRE_READ_ROM - uncomment to enable

scan OneWire bus for devices and list their ROM codes
- requires display with more than 2 text lines
- also enable ONEWIRE_PROBES or ONEWIRE_IO_PIN (see section 'Busses')
//#define SW_ONEWIRE_SCAN - uncomment to enable

capacitor leakage check
- requires display with more than two lines
//#define SW_CAP_LEAKAGE - uncomment to enable

display reverse hFE for BJTs
- hFE for collector and emitter reversed
#define SW_REVERSE_HFE - uncomment to enable

display I_C/I_E test current for hFE measurement
- I_C for common emitter circuit
I_E for common collector circuit
//#define SW_HFE_CURRENT - uncomment to enable

R/C/L monitors on probes #1 and #3
- monitors for L require SW_INDUCTOR to be enabled
- for ESR either SW_ESR or SW_OLD_ESR needs to be enabled

//#define SW_MONITOR_R /* just R */
//#define SW_MONITOR_C /* just C plus ESR */
//#define SW_MONITOR_L /* just L */
//#define SW_MONITOR_RCL /* R plus L, or C plus ESR */
//#define SW_MONITOR_RL /* R plus L * - uncomm.(one or more)

DHT11, DHT22 and compatible humidity & temperature sensors
//#define SW_DHTXX - uncomment to enable

display font for test purposes
//#define SW_FONT_TEST - uncomment to enable

check resistor for matching E series norm value
- requires a display with more than 2 text lines
- color-code mode requires a color graphics display

//#define SW_R_E24_5_T E24 5% tolerance, text
//#define SW_R_E24_5_CC E24 5% tolerance, color-code
//#define SW_R_E24_1_T E24 1% tolerance, text
//#define SW_R_E24_1_CC E24 1% tolerance, color-code
//#define SW_R_E96_T E96 1% tolerance, text
//#define SW_R_E96_CC E96 1% tolerance, color-code - uncomm.(one or more)

check capacitor for matching E series norm value
- requires a display with more than 2 text lines

//#define SW_C_E6_T E6 20% tolerance, text
//#define SW_C_E12_T E12 10% tolerance, text - uncomm.(one or more)

check inductor for matching E series norm value
- requires a display with more than 2 text lines

//#define SW_L_E6_T E6 20% tolerance, text
//#define SW_L_E12_T E12 10% tolerance, text - uncomm.(one or more)

Disable hFE measurement with common collector circuit and Rl as base resistor
* - problem:
hFE values are too high because base voltage is measured too low
- affected testers:
Hiland M664 (under investigation)
//#define NO_HFE_C_RL - uncomment to enable

Oscillator startup cycles (after wakeup from power-safe mode):
- typical values
- internal RC: 6
- full swing crystal:. . . 16384 (also 256 or 1024 based on fuse settings)
- low power crystal:. . . 16384 (also 256 or 1024 based on fuse settings)
* - Please change value if it doesn't match your tester!

```
#ifndef OSC_STARTUP
  #define OSC_STARTUP 16384
#endif
```

Listing 6.6. Please change value if it doesn't match your tester!

6.2.3. user interface

Languange of user interface Standard is ISO 8859 -1.
- Choice _2 is ISO 8859 -2. - Russian is always Windows -1251.

```
#define UI_ENGLISH
//#define UI_CZECH
//#define UI_CZECH_2
//#define UI_DANISH
//#define UI_GERMAN
//#define UI_ITALIAN
//#define UI_POLISH
//#define UI_POLISH_2
//#define UI_ROMANIAN
//#define UI_RUSSIAN
//#define UI_RUSSIAN_2
//#define UI_SPANISH
```

Listing 6.7. When voice dialing is commented out

Use comma instead of dot to indicate a decimal fraction.
#define UI_COMMA* * - comment out to deactivate

Display temperatures in Fahrenheit instead of Celsius.
//#define UI_FAHRENHEIT - uncomment to enable

Display hexadecimal values in uppercase instead of lowercase.
//#define UI_HEX_UPPERCASE - uncomment to enable

Set the default operation mode to auto-hold
- instead of continous mode
//#define UI_AUTOHOLD - uncomment to enable

Trigger the menu also by a short circuit of all three probes.
- former default behaviour
//#define UI_SHORT_CIRCUIT_MENU - uncomment to enable

Show key hints instead of cursor if available..
- currently only "Menu/Test"
- requires additional keys and display with a sufficient number of text lines (recommended: >= 8 lines)
//#define UI_KEY_HINTS - uncomment to enable

select adjustment profile menu after powering on.
//#define UI_CHOOSE_PROFILE - uncomment to enable

Output components found also via TTL serial interface
- also enable SERIAL_BITBANG or SERIAL_HARDWARE (see section 'Busses')
//#define UI_SERIAL_COPY - uncomment to enable

Control tester via TTL serial interface.
- also enable SERIAL_BITBANG or SERIAL_HARDWARE plus SERIAL_RW
//#define UI_SERIA_COMMANDS - uncomment to enable

Maximum time to wait after probing (in ms)
- applies to continuous mode only.
- Time between printing the result and starting a new probing cycle.
#define CYCLE_DELAY . . . 3000 * - comment out to deactivate

Maximum number of probing runs without any component found in a row.
- applies to continuous mode only
- If this number is reached the tester will power off.
- When set to zero the tester will run only once and turn off after CYCLE_DELAY.
#define CYCLE_MAX . . . 5 * - comment out to deactivate

Automatic power-off when no button is pressed for a while (in s).
* - applies to auto-hold mode only
//#define POWER_OFF_TIMEOUT . . . 60 - uncomment to enable

component symbols for fancy pinoutcolor coding for 3-pin semiconductors
- requires graphics display and symbol bitmap
#define SW_SYMBOLS * - comment out to deactivate

color coding for probes
- requires color graphics display
- edit colors.h to select correct probe colors
(COLOR_PROBE_1, COLOR_PROBE_2 and COLOR_PROBE_3)
#define UI_PROBE_COLORS * - comment out to deactivate

colored titles
- requires color graphics display
- edit colors.h to select prefered color (COLOR_TITLE)
//#define UI_COLORED_TITLES - uncomment to enable

colored cursor and key hints
- requires color graphics display
- edit colors.h to select prefered color (COLOR_CURSOR)
//#define UI_COLORED_CURSOR - uncomment to enable

main menu: power off tester
#define SW_POWER_OFF - uncomment to enable

Round some values if appropriate.
- for DS18B20 (0.1 °C/F)
//#define UI_ROUND_DS18B20 - uncomment to enable

storage of firmware data (texts, tables etc)
- self-adjustment data is always stored in EEPROM
- fonts and symbols are always stored in Flash
#define DATA_EEPROM /* store data in EEPROM * - comment out to deactivate
//#define DATA_FLASH /* store data in Flash - uncomment to enable

6.2.4. power management

type of power switch

- soft-latching power switch (default)
as in the tester's reference circuit
tester is able to power itself off
- manual power switch
tester isn't able to power itself off
//#define POWER_SWITCH_SOFT
#define POWER_SWITCH_MANUAL - enable one

Battery monitoring mode

- BAT_NONE disable battery monitoring completely
- BAT_DIREKT direct measurement of battary voltage (< 5V)
- BAT_DIVIDER measurement via voltage divider
//#define BAT_NONE
//#define BAT_DIRECT
#define BAT_DIVIDER -prefix

Unmonitored optional external power supply

Some circuits supporting an additional external power supply are designed in a way that prevents the battery monitoring to measure the voltage of the external power supply. This would trigger the low battery shut-down. The switch below will prevent the shut-down when the measured voltage is below 0.9V (caused by the diode's leakage current).
//#define BAT_EXT_UNMONITORED * - uncomment to enable

Voltage divider for battery monitoring

- BAT_R1: top resistor in Ω
- BAT_R2: bottom resistor in Ω
#define BAT_R1 . . . 10000 ** - optimize this value
#define BAT_R2 . . . 3300 ** - optimize this value

Voltage drop by reverse voltage protection diode and power management transistor (in mV):

- or any other circuitry in the power section
- Get your DMM and measure the voltage drop!
- Schottky diode about 200mV / PNP BJT about 100mV.
#define BAT_OFFSET . . . 290 ** - optimize this value

Battery weak voltage (in mV)

- Tester warns if BAT_WEAK is reached.
- Voltage drop BAT_OFFSET is considered in calculation.
#define BAT_WEAK dots 7400 ** - optimize this value

Battery low voltage (in mV)

- Tester powers off if BAT_LOW is reached.
- Voltage drop BAT_OFFSET is considered in calculation.
#define BAT_LOW . . . 6400 ** - optimize this value

Enter sleep mode when idle to save power

#define SAVE_POWER -Comment on deactivating

6.2.5. measurement settings and offsets

ADC voltage reference based on Vcc(in mV))
#define UREF_VCC . . . 5001 ** - optimize this value

Offset for the internal bandgap voltage reference (in mV): -100 up to 100
- To compensate any difference between real value and measured value.
- The ADC has a resolution of about 4.88mV for V_ref = 5V (Vcc) and 1.07mV for V_ref = 1.1V (bandgap).
- Will be added to measured voltage of bandgap reference.
#define UREF_OFFSET . . . 0 ** - optimize this value

Exact values of probe resistors
- Standard value for Rl is 680 Ω
- Standard value for Rh is 470 kΩ
/* Rl in Ω */
#define R_LOW . . . 680 ** -If necessary optimize this value
/* Rh in Ω */
#define R_HIGH . . . 470000 ** -If necessary optimize this value

Offset for systematic error of resistor measurement with Rh (470k) in Ω
- if resistors >20k measure too high or low adjust the offset accordingly
- standard offset is 350 Ω
#define RH_OFFSET . . . 350 ** -If necessary optimize this value

Resistance of probes (in 0,01 Ω)
- default offset for PCB tracks and probe leads
- resistance of two probes in series
- assuming all probes have same/similar resistance
- will be updated by self-adjustment
#define R_ZERO . . . 20 ** -If necessary optimize this value

Use probe pair specific resistance offsets instead of an average value for all probes.
#define R_MULTIOFFSET - uncomment to enable

Capacitance of probes (in pF)
- default offset for MCU, PCB tracks and probe leads
- will be updated by self-adjustment
- capacitance length Examples:

3pF	about	10cm
9pF	about	30cm
15pF	about	50cm

- maximum value: . . . 100
#define C_ZERO . . . 43 ** -If necessary optimize this value

Use probe pair specific capacitance offsets instead of an average value for all probes
//#define CAP_MULTIOFFSET * - uncomment to enable

Maximum voltage at which we consider a capacitor being discharged (in mV)
#define CAP_DISCHARGED . . . 2 ** -If necessary optimize this value

Correction factors for capacitors (in 0,1%))
- positive factor increases capacitance value
- negative factor decreases capacitance value

CAP _FACTOR _SMALL	für Caps	< 4,7 µF
CAP _FACTOR _MID	für Caps	4,7 - 47 µF
CAP _FACTOR _LARGE	für Caps	> 47 µF

#define CAP_FACTOR_SMALL	0	no corr.	** If necessary optimize this value
#define CAP_FACTOR_MID . . .	-40	-4.0%	** If necessary optimize this value
#define CAP_FACTOR_LARGE . . .	-90	-9.0%	** If necessary optimize this value

Number of ADC samples to perform for each mesurement
- Valid values are in the range of 1 - 255.
#define ADC_SAMPLES . . . 25 **-If necessary optimize this value

100nF AREF buffer capacitor
- used by some MCU boards
- will increase measurement time
- recommendation: replace with 1nF capacitor
#define ADC_LARGE_BUFFER_CAP - comment out to deactivate

6.2.6. R & D - meant for firmware developers

Enable read functions for display module.
- display driver and interface settings have to support this
//#define LCD_READ * - uncomment to enable

Read ID of display controller ID is shown at welcome screen (after firmware version)
- requires display read functions (LCD_READ)
- recommended: serial output (UI_SERIAL_COPY)
//#define SW_DISPLAY_ID * - uncomment to enable

Read registers of display controller and output them via TTL serial.
- requires display read functions (LCD_READ)
- recommended: serial output (UI_SERIAL_COPY)
//#define SW_DISPLAY_REG * - uncomment to enable

6.2.7. Busses

I2C bus - might be required by some hardware
- could be enabled already in display section (config_<MCU>.h)
- for bit-bang I2C port and pins see I2C_PORT (config_<MCU>.h)
- hardware I2C (TWI) uses automatically the proper MCU pins
- uncomment either I2C_BITBANG or I2C_HARDWARE to enable
- uncomment one of the bus speed modes

#define I2C_BITBANG	bit-bang I2C
#define I2C_HARDWARE	MCU's hardware TWI
#define I2C_STANDARD_MODE	100kHz bus speed
#define I2C_FAST_MODE	400kHz bus speed
#define I2C_RW	enable I2C read support (untested)

SPI bus - might be required by some hardware
- could be enabled already in display section (config_<MCU>.h)
- for bit-bang SPI port and pins see SPI_PORT (config_<MCU>.h)
- hardware SPI uses automatically the proper MCU pins
- uncomment either SPI_BITBANG or SPI_HARDWARE to enable

#define SPI_BITBANG	bit-bang SPIC
#define SPI_HARDWARE	hardware SPI
#define SPI_RW	enable SPI read support

TTL serial interface - could be enabled already in display section (config_<MCU>.h)
- for bit-bang serial port and pins see SERIAL_PORT (config_<MCU>.h)
- hardware serial uses automatically the proper MCU pins
- uncomment either SERIAL_BITBANG or SERIAL_HARDWARE to enable

#define SERIAL_BITBANG	bit-bang seriell
#define SERIAL_HARDWARE	hardware seriell
#define SERIAL_RW	enable serial read support

OneWire bus - for dedicated I/O pin please see ONEWIRE_PORT (config_<MCU>.h)
* - uncomment either ONEWIRE_PROBES or ONEWIRE_ to enable

//#define ONEWIRE_PROBES	via probes
//#define ONEWIRE_IO_PIN	via dedicated I/O pin

6.3. Config_328.h

contains low-level settings for ads, Keys and so on. When translating the firmware becomes the appropriate file automatically integrated according to the MCU. It is also about one again C header file, i. the comment rules apply to C. Next to the grqq // " for individual lines Block comments are also used with "#if 0 ... #endif ". To a block comment, just insert a "// " in front of the corresponding "#if 0 " and "#endif "; to comment out the reverse path. You can also comment on a block, by deleting the lines with the "#if 0 " and "#endif ".

6.3.1. LCD module

The existing display and settings are also in chapter 2.6 on page 12.
An example is the ST7565R module where #if 0 and #endif are commented out with "//" .

ST7565R

```
//#if 0
#define LCD_ST7565R           /* display controller ST7565R */
```

Listing 6.8. Drouqh //#if 0 end #define . . . is ST7565R active

```
//#endif
```

Listing 6.9. The end of block is comment

6.3.2. port and pin assignments

Test probes

- Must be an ADC port :-)
- Lower 3 pins of the port must be used for probe pins.
- Please don't change the definitions of TP1, TP2 and TP3!
- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define TP1         PC0     /* test pin / probe #1 */
#define TP2         PC1     /* test pin / probe #2 */
#define TP3         PC2     /* test pin / probe #3 */
```

Listing 6.10. Please don't change the definitions

Options

- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define TP_ZENER    PC3     /* test pin for 10:1 voltage divider */
#define TP_REF      PC4     /* test pin for 2.5V reference and relay */
#define TP_BAT      PC5     /* test pin for battery (4:1 voltage divider) */
```

Listing 6.11. Please don't change the definitions

Probe resistors

- For PWM/squarewave output via probe #2 R_RL_2 has to be PB2/OC1B.
- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define R_RL_1      PB0     /* Rl (680R) for test pin #1 */
#define R_RH_1      PB1     /* Rh (470k) for test pin #1 */
#define R_RL_2      PB2     /* Rl (680R) for test pin #2 */
#define R_RH_2      PB3     /* Rh (470k) for test pin #2 */
#define R_RL_3      PB4     /* Rl (680R) for test pin #3 */
#define R_RH_3      PB5     /* Rh (470k) for test pin #3 */
```

Listing 6.12. Please don't change the definitions

dedicated signal output via OC1B - don't change this!

```
#define SIGNAL_OUT PB2      /* MCU's OC1B pin */
```

Listing 6.13. don't change this

power control - can't be same port as ADC_PORT or R_PORT

```
#define POWER_CTRL PD6      /* control pin (1: on / 0: off) */
```

Listing 6.14. - comment or edit port pin

test push button - can't be same port as ADC_PORT or R_PORT

```
#define TEST_BUTTON PD7   /* test/start push button (low active) */
```

Listing 6.15. - comment or edit port pin

rotary encoder

```
#define ENCODER_A  PD3    /* rotary encoder A signal */
#define ENCODER_B  PD2    /* rotary encoder B signal */
```

Listing 6.16. - comment or edit port pin

increase/decrease push buttons

```
#define KEY_INC    PD2    /* increase push button (low active) */
#define KEY_DEC    PD3    /* decrease push button (low active) */
```

Listing 6.17. - comment or edit port pin

frequency counter

- basic and extented version
- input must be pin PD4/T0

```
#define COUNTER_IN PD4    /* signal input T0 */
```

Listing 6.18. - comment or edit port pin

IR detector/decoder

- fixed module connected to dedicated I/O pin

```
#define IR_DATA    PC6    /* data signal */
```

Listing 6.19. - comment or edit port pin

6.3.3. Busse

SPI - hardware SPI uses: SCK PB5, MOSI PB3, MISO PB4 and /SS PB2
- could be already set in display section
- unused signals can be ignored
- /SS is set to output mode for hardware SPI but not used

```
/* for bit-bang and hardware SPI */
#ifndef SPI_PORT
#define SPI_PORT   PORTB  /* port data register */
#define SPI_DDR    DDRB   /* port data direction register */
#define SPI_PIN    PINB   /* port input pins register */
#define SPI_SCK    PB5    /* pin for SCK */
#define SPI_MOSI   PB3    /* pin for MOSI */
#define SPI_MISO   PB4    /* pin for MISO */
#define SPI_SS     PB2    /* pin for /SS */
#endif
```

Listing 6.20. - comment or edit port pin

I2C - hardware I2C (TWI) uses SDA PC4 and SCL PC5
- could be already set in display section

```
#ifndef I2C_PORT
#define I2C_PORT   PORTC  /* port data register */
#define I2C_DDR    DDRC   /* port data direction register */
#define I2C_PIN    PINC   /* port input pins register */
#define I2C_SDA    PC4    /* pin for SDA */
#define I2C_SCL    PC5    /* pin for SCL */
#endif
```

Listing 6.21. - comment or edit port pin

TTL serial interface
- hardware USART uses
USART0: Rx PD0 and Tx PD1

```
/* for hardware RS232 */
#define SERIAL_USART 0    /* use USART0 */
/* for bit-bang RS232 */
#define SERIAL_PORT PORTD /* port data register */
#define SERIAL_DDR DDRD   /* port data direction register */
#define SERIAL_PIN PIND   /* port input pins register */
#define SERIAL_TX  PD1    /* pin for Tx (transmit) */
#define SERIAL_RX  PD0    /* pin for Rx (receive) */
#define SERIAL_PCINT 16   /* PCINT# for Rx pin */
```

Listing 6.22. - comment or edit port pin

OneWire - dedicated I/O pin

```
#define ONEWIRE_PORT PORTC /* port data register */
#define ONEWIRE_DDR DDRC  /* port data direction register */
#define ONEWIRE_PIN PINC  /* port input pins register */
#define ONEWIRE_DQ PC6    /* DQ (data line) */
```

Listing 6.23. - comment or edit port pin

6.4. Config_644.h

contains low-level settings for ads, Keys and so on. When translating the firmware becomes the appropriate file automatically integrated according to the MCU. It is also about one again C header file, i. the comment rules apply to C. Next to the grqq // " for individual lines Block comments are also used with "#if 0 ... #endif ". To a block comment, just insert a "// " in front of the corresponding "#if 0 " and "#endif "; to comment out the reverse path. You can also comment on a block, by deleting the lines with the "#if 0 " and "#endif ".

6.4.1. LCD module

The existing display and settings are also in chapter 2.6.
An example is the ILI9486 module where #if 0 and #endif are commented out with "//" . on page 12.

ILI9481/ILI9486

```
#if 0
//#define LCD_ILI9481          /* display controller ILI9481 */
#define LCD_ILI9486           /* display controller ILI9486 */
```

Listing 6.24. Drouqh //#if 0 end #define . . . is ILI9586 active

```
#endif
```

Listing 6.25. The end of block is comment

6.4.2. port and pin assignments

Test probes

- Must be an ADC port :-)
- Lower 3 pins of the port must be used for probe pins.
- Please don't change the definitions of TP1, TP2 and TP3!
- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define TP1        PA0    /* test pin / probe #1 */
#define TP2        PA1    /* test pin / probe #2 */
#define TP3        PA2    /* test pin / probe #3 */
```

Listing 6.26. Please don't change the definitions

Options

- Must be an ADC port :-)
- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define TP_ZENER   PA3    /* test pin for 10:1 voltage divider */
#define TP_REF     PA4    /* test pin for 2.5V reference and relay */
#define TP_BAT     PA5    /* test pin for battery (4:1 voltage divider) */
#define TP_CAP     PA7    /* test pin for self-adjustment cap */
```

Listing 6.27. Please don't change the definitions

Probe resistors

- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define R_RL_1     PD2    /* Rl (680R) for test pin #1 */
#define R_RH_1     PD3    /* Rh (470k) for test pin #1 */
#define R_RL_2     PD4    /* Rl (680R) for test pin #2 */
#define R_RH_2     PD5    /* Rh (470k) for test pin #2 */
#define R_RL_3     PD6    /* Rl (680R) for test pin #3 */
#define R_RH_3     PD7    /* Rh (470k) for test pin #3 */
```

Listing 6.28. Please don't change the definitions

dedicated signal output via OC1B - don't change this!

```
#define SIGNAL_OUT PD4    /* MCU's OC1B pin */
```

Listing 6.29. don't change this

power control - can't be same port as ADC_PORT or R_PORT

```
#define POWER_CTRL PC6    /* control pin (1: on / 0: off) */
```

Listing 6.30. - comment or edit port pin

test push button - can't be same port as ADC_PORT or R_PORT

```
#define TEST_BUTTON PC7   /* test/start push button (low active) */
```

Listing 6.31. - comment or edit port pin

rotary encoder

```
#define ENCODER_A  PC3    /* rotary encoder A signal */
#define ENCODER_B  PC4    /* rotary encoder B signal */
```

Listing 6.32. - comment or edit port pin

increase/decrease push buttons

```
#define KEY_INC    PC4    /* increase push button (low active) */
#define KEY_DEC    PC3    /* decrease push button (low active) */
```

Listing 6.33. - comment or edit port pin

frequency counter

- basic and extented version
- input must be pin PB0/T0

```
#define COUNTER_IN    PB0    /* signal input T0 */
```

Listing 6.34. - comment or edit port pin

control for extended frequency counter

```
#define COUNTER_CTRL_DIV PC0  /* prescaler (low 1:1, high x:1) */
#define COUNTER_CTRL_CH0 PC1  /* channel addr #0 */
#define COUNTER_CTRL_CH1 PC2  /* channel addr #1 */
```

Listing 6.35. - comment or edit port pin

L/C Meter

- frequency input must be pin PB0/T0

```
#define LC_CTRL_CP PC0    /* reference cap (low: on / high: off) */
#define LC_CTRL_LC PC1    /* L/C selection (low: C / high: L */
```

Listing 6.36. - comment or edit port pin

IR detector/decoder

- fixed module connected to dedicated I/O pin

```
#define IR_DATA    PC2    /* data signal */
```

Listing 6.37. - comment or edit port pin

fixed cap for self-adjustment

- ADC pin is TP_CAP from above
- settings are for 470k resistor
- should be film cap with 100nF - 1000nF

```
#define ADJUST_RH  PC5    /* Rh (470k) for fixed cap */
```

Listing 6.38. - comment or edit port pin

relay for parallel cap (sampling ADC)

- TP1 & TP3
- cap should have 10nF - 27nF

```
#define CAP_RELAY_CTRL PC2 /* control pin */
```

Listing 6.39. - comment or edit port pin

6.4.3. Busse

SPI - hardware SPI uses: SCK PB7, MOSI PB5, MISO PB6 and /SS PB4
- could be already set in display section
- unused signals can be ignored
- /SS is set to output mode for hardware SPI but not used

```
/* for bit-bang and hardware SPI */
#ifndef SPI_PORT
#define SPI_PORT   PORTB   /* port data register */
#define SPI_DDR    DDRB    /* port data direction register */
#define SPI_PIN    PINB    /* port input pins register */
#define SPI_SCK    PB7     /* pin for SCK */
#define SPI_MOSI   PB5     /* pin for MOSI */
#define SPI_MISO   PB6     /* pin for MISO */
#define SPI_SS     PB4     /* pin for /SS */
#endif
```

Listing 6.40. - comment or edit port pin

I2C - hardware I2C (TWI) uses SDA PC1 and SCL PC0
- could be already set in display section

```
/* for bit-bang I2C */
#ifndef I2C_PORT
#define I2C_PORT   PORTC   /* port data register */
#define I2C_DDR    DDRC    /* port data direction register */
#define I2C_PIN    PINC    /* port input pins register */
#define I2C_SDA    PC1     /* pin for SDA */
#define I2C_SCL    PC0     /* pin for SCL */
#endif
```

Listing 6.41. - comment or edit port pin

TTL serial interface
- hardware USART uses

USART0: Rx PD0 and Tx PD1 USART2: Rx PD2 and Tx PD3

```
/* for hardware TTL serial */
#define SERIAL_USART 0     /* use USART0 */
/* for bit-bang TTL serial */
#define SERIAL_PORT PORTD /* port data register */
#define SERIAL_DDR DDRD   /* port data direction register */
#define SERIAL_PIN PIND   /* port input pins register */
#define SERIAL_TX  PD1    /* pin for Tx (transmit) */
#define SERIAL_RX  PD0    /* pin for Rx (receive) */
#define SERIAL_PCINT 24   /* PCINT# for Rx pin */
```

Listing 6.42. - comment or edit port pin

OneWire - dedicated I/O pin

```
#define ONEWIRE_PORT PORTC /* port data register */
#define ONEWIRE_DDR DDRC   /* port data direction register */
#define ONEWIRE_PIN PINC   /* port input pins register */
#define ONEWIRE_DQ PC2     /* DQ (data line) */
```

Listing 6.43. - comment or edit port pin

6.5. Config_1280

contains low-level settings for ads, Keys and so on. When translating the firmware becomes the appropriate file automatically integrated according to the MCU. It is also about one again C header file, i. the comment rules apply to C. Next to the grqq // " for individual lines Block comments are also used with "#if 0 ... #endif ". To a block comment, just insert a "// " in front of the corresponding "#if 0 " and "#endif "; to comment out the reverse path. You can also comment on a block, by deleting the lines with the "#if 0 " and "#endif ".

6.5.1. LCD module

The existing display and settings are also in chapter 2.6 on page 12.
An example is the ILI9486 module where #if 0 and #endif are commented out with "//" .

ILI9481/ILI9486

```
//#if 0
//#define LCD_ILI9481          /* display controller ILI9481 */
#define LCD_ILI9486          /* display controller ILI9486 */
```

Listing 6.44. Drouqh //#if 0 end #define . . . is ILI9586 active

```
//#endif
```

Listing 6.45. The end of block is comment

6.5.2. port and pin assignments

Test probes

- Must be an ADC port :-)
- Lower 3 pins of the port must be used for probe pins.
- Please don't change the definitions of TP1, TP2 and TP3!
- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define TP1        PF0    /* test pin / probe #1 */
#define TP2        PF1    /* test pin / probe #2 */
#define TP3        PF2    /* test pin / probe #3 */
```

Listing 6.46. Please don't change the definitions

Options

- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define TP_ZENER   PF3    /* test pin for 10:1 voltage divider */
#define TP_REF     PF4    /* test pin for 2.5V reference and relay */
#define TP_BAT     PF5    /* test pin for battery (4:1 voltage divider) */
```

Listing 6.47. Please don't change the definitions

Probe resistors

- For PWM/squarewave output via probe #2 R_RL_2 has to be PB6/OC1B.
- Don't share this port with POWER_CTRL or TEST_BUTTON!

```
#define R_RL_1     PK0    /* Rl (680R) for test pin #1 */
#define R_RH_1     PK1    /* Rh (470k) for test pin #1 */
#define R_RL_2     PK2    /* Rl (680R) for test pin #2 */
#define R_RH_2     PK3    /* Rh (470k) for test pin #2 */
#define R_RL_3     PK4    /* Rl (680R) for test pin #3 */
#define R_RH_3     PK5    /* Rh (470k) for test pin #3 */
```

Listing 6.48. Please don't change the definitions

dedicated signal output via OC1B - don't change this!

```
#define SIGNAL_OUT PB6    /* MCU's OC1B pin */
```

Listing 6.49. don't change this

power control - can't be same port as ADC_PORT or R_PORT

```
#define POWER_CTRL PA6    /* control pin (1: on / 0: off) */
```

Listing 6.50. - comment or edit port pin

test push button - can't be same port as ADC_PORT or R_PORT

```
#define TEST_BUTTON PA7   /* test/start push button (low active) */
```

Listing 6.51. - comment or edit port pin

rotary encoder

```
#define ENCODER_A  PA3    /* rotary encoder A signal */
#define ENCODER_B  PA1    /* rotary encoder B signal */
```

Listing 6.52. - comment or edit port pin

increase/decrease push buttons

```
#define KEY_INC    PA3    /* increase push button (low active) */
#define KEY_DEC    PA1    /* decrease push button (low active) */
```

Listing 6.53. - comment or edit port pin

frequency counter

- basic and extented version
- input must be pin PD7/T0

```
#define COUNTER_IN    PD7    /* signal input T0 */
```

Listing 6.54. - comment or edit port pin

control for extended frequency counter

```
#define COUNTER_CTRL_DIV PD4  /* prescaler (low 1:1, high x:1) */
#define COUNTER_CTRL_CH0 PD5  /* channel addr #0 */
#define COUNTER_CTRL_CH1 PD6  /* channel addr #1 */
```

Listing 6.55. - comment or edit port pin

L/C Meter

- frequency input must be pin PB0/T0

```
#define LC_CTRL_CP PC4    /* reference cap (low: on / high: off) */
#define LC_CTRL_LC PC5    /* L/C selection (low: C / high: L */
```

Listing 6.56. - comment or edit port pin

IR detector/decoder

- fixed module connected to dedicated I/O pin

```
#define IR_DATA    PA0    /* data signal */
```

Listing 6.57. - comment or edit port pin

fixed cap for self-adjustment

- ADC pin is TP_CAP from above
- settings are for 470k resistor
- should be film cap with 100nF - 1000nF

```
#define ADJUST_RH  PA5    /* Rh (470k) for fixed cap */
```

Listing 6.58. - comment or edit port pin

relay for parallel cap (sampling ADC)

- TP1 & TP3
- cap should have 10nF - 27nF

```
#define CAP_RELAY_CTRL PA2 /* control pin */
```

Listing 6.59. - comment or edit port pin

6.5.3. Busse

SPI - hardware SPI uses: SCK PB1, MOSI PB2, MISO PB3 and /SS PB0
- could be already set in display section
- unused signals can be ignored
- /SS is set to output mode for hardware SPI but not used

```
/* for bit-bang and hardware SPI */
#ifndef SPI_PORT
#define SPI_PORT   PORTB   /* port data register */
#define SPI_DDR    DDRB    /* port data direction register */
#define SPI_PIN    PINB    /* port input pins register */
#define SPI_SCK    PB1     /* pin for SCK */
#define SPI_MOSI   PB2     /* pin for MOSI */
#define SPI_MISO   PB3     /* pin for MISO */
#define SPI_SS     PB0     /* pin for /SS */
#endif
```

Listing 6.60. - comment or edit port pin

I2C - hardware I2C (TWI) uses SDA PD1 and SCL PD0
- could be already set in display section

```
/* for bit-bang I2C */
#ifndef I2C_PORT
#define I2C_PORT   PORTD   /* port data register */
#define I2C_DDR    DDRD    /* port data direction register */
#define I2C_PIN    PIND    /* port input pins register */
#define I2C_SDA    PD1     /* pin for SDA */
#define I2C_SCL    PD0     /* pin for SCL */
#endif
```

Listing 6.61. - comment or edit port pin

TTL serial interface

- hardware USART uses

USART0: Rx PE0 and Tx PE1
USART1: Rx PD2 and Tx PD3
USART2: Rx PH0 and Tx PH1
USART3: Rx PJ0 and Tx PJ1

```
/* for hardware TTL serial */
#define SERIAL_USART 0    /* use USART0 */
/* for bit-bang TTL serial */
#define SERIAL_PORT PORTE /* port data register */
#define SERIAL_DDR DDRE   /* port data direction register */
#define SERIAL_PIN PINE   /* port input pins register */
#define SERIAL_TX  PE1    /* pin for Tx (transmit) */
#define SERIAL_RX  PE0    /* pin for Rx (receive) */
#define SERIAL_PCINT 8    /* PCINT# for Rx pin */
```

Listing 6.62. - comment or edit port pin

OneWire - dedicated I/O pin

```
#define ADJUST_PORT PORTA /* port data register */
#define ADJUST_DDR DDRA   /* port data direction register */
#define ADJUST_RH  PA5    /* Rh (470k) for fixed cap */
```

Listing 6.63. - comment or edit port pin

Chapter 7 Collection of settings

this chapter provides the collection of settings for different tester models. Here you can find the necessary settings also from your tester. If he should not be there and you have the settings from this manual, or experimentally found out, please send the settings via email to the author [8] to help other users with it.

7.1. Arduino Nano, Uno or Mega 2560

Nano/Uno: ATmega 328, 16MHz clock Mega 2560: ATmega 2560, 16MHz clock
- Download the Arduino pinout diagram to get the mapping between Arduino and ATmega pins on page 75.
- only the essential settings are listed

```
#define HW_FIXED_SIGNAL_OUTPUT
```

Listing 7.1. Hardware Options

```
#define UI_AUTOHOLD
```

Listing 7.2. User interface

```
#define POWER_SWITCH_MANUAL
#define BAT_NONE
```

Listing 7.3. Power management

```
#define ADC_LARGE_BUFFER_CAP
```

Listing 7.4. Measurement settings and offsets

7.2. DIY Kit "‘AY-AT"’ or GM328A

- ATmega328;
- ST7735 color LCD module (bit-bang SPI)
- rotary encoder (PD1 & PD3, in parallel with display)
- external 2.5V voltage reference (TL431)
- basic frequency counter with dedicated input (PD4)
- measurement of external voltage up to 45V (PC3)
- settings provided by flywheelz@EEVBlog

```
#define HW_ENCODER
#define ENCODER_PULSES 4      /* usually 4 pulses per step */
#define ENCODER_STEPS 20      /* usually 20 detents */
#define HW_REF25
#define HW_ZENER
#define HW_FREQ_COUNTER_BASIC
```

Listing 7.5. Hardware Options

```
#define ENCODER_PORT PORTD /* port data register */
#define ENCODER_DDR DDRD  /* port data direction register */
#define ENCODER_PIN PIND  /* port input pins register */
#define ENCODER_A  PD3    /* rotary encoder A signal */
#define ENCODER_B  PD1    /* rotary encoder B signal */
```

Listing 7.6. Rotary Encoder

```
#define LCD_SEMI_ST7735
#define LCD_GRAPHIC               /* graphic display */
#define LCD_COLOR                 /* color display */
#define LCD_SPI                   /* SPI interface */
#define LCD_PORT    PORTD         /* port data register */
#define LCD_DDR     DDRD          /* port data direction register */
#define LCD_RES     PD0           /* port pin used for /RESX */
#define LCD_CS      PD5           /* port pin used for /CSX (optional) */
#define LCD_DC      PD1           /* port pin used for D/CX */
#define LCD_SCL     PD2           /* port pin used for SCL */
#define LCD_SDA     PD3           /* port pin used for SDA */
#define LCD_DOTS_X 160            /* number of horizontal dots */
#define LCD_DOTS_Y 128            /* number of vertical dots */
#define LCD_LATE_ON               /* turn on LCD after clearing it */
#define FONT_8X8_HF               /* 8x8 font */
#define SYMBOLS_30X32_HF          /* 30x32 symbols */
#define SPI_BITBANG               /* bit-bang SPI */
#define SPI_PORT    LCD_PORT      /* SPI port data register */
#define SPI_DDR     LCD_DDR       /* SPI port data direction register */
#define SPI_SCK     LCD_SCL       /* port pin used for SCK */
#define SPI_MOSI    LCD_SDA       /* port pin used for MOSI */
```

Listing 7.7. LCD module

If you prefer that the tester starts with a cleared display uncomment LCD_LATE_ON.

```
#define LCD_DOTS_X 160            /* number of horizontal dots */
#define LCD_DOTS_Y 128            /* number of vertical dots */
#define LCD_LATE_ON               /* turn on LCD after clearing it */
#define FONT_8X8_HF               /* 8x8 font */
#define SYMBOLS_30X32_HF          /* 30x32 symbols */
```

Listing 7.8. For the ST7735 semi-compatible display use

Input for the frequency counter is PD4 (T0).
Inductance compensation offsets for 20MHz model - provided by indman@EEVBlog
- edit the section for high current mode in function MeasureInductor() in inductor.c

```
    #if CPU_FREQ == 20000000
    /* 20 MHz */
    if (Temp < 1500)  /* < 1.5us / < 100uH */
    {
      Offset = -10;
    }
    else if (Temp < 5000) /* 1.5-5us / 100-330uH */
    {
      Offset = -10;
    }
    else              /* > 5us / > 330uH */
    {
      Offset = -30;
    }
    #endif
```

Listing 7.9. Inductance compensation offsets

7.3. BSide ESR02

(DTU-1701) - ATmega 328, 8MHz clock
- ST7565 display (bit-bang SPI)
- external 2.5V voltage reference (TL431)
- settings provided by indman@EEVblog

```
#define HW_REF25
```

Listing 7.10. Hardware Options.

```
#define BAT_DIVIDER
#define BAT_EXT_UNMONITORED
#define BAT_R1     47000
#define BAT_R2     47000
#define BAT_OFFSET 420
```

Listing 7.11. Power management

```
#define LCD_ST7565R             /* display controller ST7565R */
#define LCD_GRAPHIC             /* graphic display */
#define LCD_SPI                 /* SPI interface */
#define LCD_PORT   PORTD        /* port data register */
#define LCD_DDR    DDRD         /* port data direction register */
#define LCD_RESET  PD0          /* port pin used for /RES (optional) */
#define LCD_A0     PD1          /* port pin used for A0 */
#define LCD_SCL    PD2          /* port pin used for SCL */
#define LCD_SI     PD3          /* port pin used for SI (LCD's data input) */
#define LCD_DOTS_X 128          /* number of horizontal dots */
#define LCD_DOTS_Y 64           /* number of vertical dots */
#define LCD_OFFSET_X            /* enable x offset of 4 dots */
#define LCD_FLIP_X              /* enable horizontal flip */
#define LCD_FLIP_Y              /* enable vertical flip */
#define LCD_START_Y 0           /* start line (0-63) */
#define LCD_CONTRAST 15         /* default contrast (0-63) */
#define FONT_8X8_VF             /* 8x8 font */
#define SYMBOLS_24X24_VFP       /* 24x24 symbols */
#define SPI_BITBANG             /* bit-bang SPI */
#define SPI_PORT   LCD_PORT     /* SPI port data register */
#define SPI_DDR    LCD_DDR      /* SPI port data direction register */
#define SPI_SCK    LCD_SCL      /* port pin used for SCK */
#define SPI_MOSI   LCD_SI       /* port pin used for MOSI */
```

Listing 7.12. LCD module

7.4. Fish8840 TFT

- ATmega328, 8MHz clock; ST7565 display (bit-bang SPI)
- external 2.5V voltage reference (TL431)
- settings provided by indman@EEVBlog / bdk100@vrtp.ru

```
#define LCD_ST7735
#define LCD_GRAPHIC               /* graphic display */
#define LCD_COLOR                 /* color display */
#define LCD_SPI                   /* SPI interface */
#define LCD_PORT    PORTD         /* port data register */
#define LCD_DDR     DDRD          /* port data direction register */
#define LCD_RES     PD3           /* port pin used for /RESX (optional) */
//#define LCD_CS     PD5           /* port pin used for /CSX (optional) */
#define LCD_DC      PD2           /* port pin used for D/CX */
#define LCD_SCL     PD0           /* port pin used for SCL */
#define LCD_SDA     PD1           /* port pin used for SDA */
#define LCD_DOTS_X 128            /* number of horizontal dots */
#define LCD_DOTS_Y 156            /* number of vertical dots */
#define LCD_OFFSET_X              /* enable x offset of 4 dots */
#define LCD_OFFSET_Y              /* enable y offset of 2 dots */
#define LCD_FLIP_X                /* enable horizontal flip */
//#define LCD_FLIP_Y               /* enable vertical flip */
#define LCD_ROTATE                /* switch X and Y (rotate by 90Grad) */
#define LCD_LATE_ON               /* turn on LCD after clearing it */
#define FONT_10X16_HF             /* 10x16 font */
#define SYMBOLS_30X32_HF          /* 30x32 symbols */
#define SPI_BITBANG               /* bit-bang SPI */
#define SPI_PORT    LCD_PORT      /* SPI port data register */
#define SPI_DDR     LCD_DDR       /* SPI port data direction register */
#define SPI_SCK     LCD_SCL       /* port pin used for SCK */
#define SPI_MOSI    LCD_SDA       /* port pin used for MOSI */
```

Listing 7.13. LCD module

7.5. GM328 !NOT GM328A!

Which is described above as 7.2.
- ATmega328, 8MHz clock; ST7565 display (bit-bang SPI)
- settings provided by rddube@EEVblog

```
#define LCD_ST7565R
#define LCD_GRAPHIC               /* graphic display */
#define LCD_SPI                   /* SPI interface */
#define LCD_PORT    PORTD         /* port data register */
#define LCD_DDR     DDRD          /* port data direction register */
#define LCD_RESET   PD0           /* port pin used for /RES (optional) */
#define LCD_A0      PD1           /* port pin used for A0 */
#define LCD_SCL     PD2           /* port pin used for SCL */
#define LCD_SI      PD3           /* port pin used for SI (LCD's data input) */
#define LCD_CS      PD5           /* port pin used for /CS1 (optional) */
#define LCD_DOTS_X 128            /* number of horizontal dots */
#define LCD_DOTS_Y 64             /* number of vertical dots */
#define LCD_START_Y 0             /* start line (0-63) */
#define LCD_CONTRAST 11           /* default contrast (0-63) */
#define FONT_8X8_VF               /* 8x8 font */
#define SYMBOLS_24X24_VFP         /* 24x24 symbols */
#define SPI_BITBANG               /* bit-bang SPI */
#define SPI_PORT    LCD_PORT      /* SPI port data register */
#define SPI_DDR     LCD_DDR       /* SPI port data direction register */
#define SPI_SCK     LCD_SCL       /* port pin used for SCK */
#define SPI_MOSI    LCD_SI        /* port pin used for MOSI */
```

Listing 7.14. LCD module

7.6. Hiland M644

- ATmega 644, 8MHz clock; ST7565 display (bit-bang SPI)
- rotary encoder (PB7 & PB5, in parallel with display)
- external 2.5V voltage reference (TL431)
- boost converter for Zener check
- extended frequency counter
(but no input buffer stage for direct frequency measurement)
- fixed adjustment cap
(in case of problems replace MLCC with 220nF film cap)
- settings provided by Horst O. (obelix2007@mikrocontroller.net)

```
#define HW_ENCODER
#define ENCODER_PULSES 4      /* 4 */
#define HW_REF25
#define HW_ZENER
#define HW_FREQ_COUNTER_EXT
#define FREQ_COUNTER_PRESCALER 16 /* 16:1 */
#define HW_ADJUST_CAP
```

Listing 7.15. Hardware Options.

```
#define NO_HFE_C_RL           /* if hFE values too high */
```

Listing 7.16. Workarounds! !Set NO HFE C RL!

```
#define LCD_ST7565R
#define LCD_GRAPHIC           /* graphic display */
#define LCD_SPI               /* SPI interface */
#define LCD_PORT   PORTB      /* port data register */
#define LCD_DDR    DDRB       /* port data direction register */
#define LCD_RESET  PB4        /* port pin used for /RES (optional) */
//#define LCD_CS    PB2        /* port pin used for /CS1 (optional) */
#define LCD_A0     PB5        /* port pin used for A0 */
#define LCD_SCL    PB6        /* port pin used for SCL */
#define LCD_SI     PB7        /* port pin used for SI (LCD's data input) */
#define LCD_DOTS_X 128        /* number of horizontal dots */
#define LCD_DOTS_Y 64         /* number of vertical dots */
//#define LCD_OFFSET_X        /* enable x offset of 4 dots */
//#define LCD_FLIP_X           /* enable horizontal flip */
#define LCD_FLIP_Y            /* enable vertical flip */
#define LCD_START_Y 0         /* start line (0-63) */
#define LCD_CONTRAST 3        /* default contrast (0-63) */
#define FONT_8X8_VF           /* 8x8 font */
#define SYMBOLS_24X24_VFP     /* 24x24 symbols */
#define SPI_BITBANG           /* bit-bang SPI */
#define SPI_PORT   LCD_PORT   /* SPI port data register */
#define SPI_DDR    LCD_DDR    /* SPI port data direction register */
#define SPI_SCK    LCD_SCL    /* port pin used for SCK */
#define SPI_MOSI   LCD_SI     /* port pin used for MOSI */
```

Listing 7.17. LCD module

```
#define POWER_PORT PORTB  /* port data register */
#define POWER_DDR  DDRB   /* port data direction register */
#define POWER_CTRL PB1    /* controls power (1: on / 0: off) */
```

Listing 7.18. Pinout for power control

```
#define ENCODER_PORT PORTB /* port data register */
#define ENCODER_DDR DDRB  /* port data direction register */
#define ENCODER_PIN PINB  /* port input pins register */
#define ENCODER_A  PB5    /* rotary encoder A signal */
#define ENCODER_B  PB7    /* rotary encoder B signal */
```

Listing 7.19. Rotary Encoder

```
#define KEY_PORT   PORTB  /* port data register */
#define KEY_DDR    DDRB  /* port data direction register */
#define KEY_PIN    PINB   /* port input pins register */
#define KEY_INC    PB5    /* increase push button (low active) */
#define KEY_DEC    PB7    /* decrease push button (low active) */
```

Listing 7.20. Increase/decrease push buttons)

7.7. M12864 DIY Transistor Tester

- ATmega328, 8MHz clock
- ST7565 display (bit-bang SPI)
- rotary encoder (PD1 & PD3, in parallel with display)
- external 2.5V voltage reference (TL431)

```
#define HW_ENCODER
#define ENCODER_PULSES 4  /* not confirmed yet, could be also 2 */
#define ENCODER_STEPS 24  /* not confirmed yet */
#define HW_REF25
```

Listing 7.21. Hardware Options

```
#define LCD_ST7565R
#define LCD_GRAPHIC            /* graphic display */
#define LCD_SPI                /* SPI interface */
#define LCD_PORT   PORTD       /* port data register */
#define LCD_DDR    DDRD        /* port data direction register */
#define LCD_RESET  PD0         /* port pin used for /RES */
#define LCD_A0     PD1         /* port pin used for A0 */
#define LCD_SCL    PD2         /* port pin used for SCL */
#define LCD_SI     PD3         /* port pin used for SI (LCD's data input) */
#define LCD_DOTS_X 128         /* number of horizontal dots */
#define LCD_DOTS_Y 64          /* number of vertical dots */
//#define LCD_OFFSET_X          /* enable x offset of 4 dots */
#define LCD_FLIP_Y             /* enable vertical flip */
#define LCD_START_Y 0          /* start line (0-63) */
#define LCD_CONTRAST 11        /* default contrast (0-63) */
#define FONT_8X8_VF            /* 8x8 font */
#define SYMBOLS_24X24_VFP      /* 24x24 symbols */
#define SPI_BITBANG            /* bit-bang SPI */
#define SPI_PORT   LCD_PORT    /* SPI port data register */
#define SPI_DDR    LCD_DDR     /* SPI port data direction register */
#define SPI_SCK    LCD_SCL     /* port pin used for SCK */
#define SPI_MOSI   LCD_SI      /* port pin used for MOSI */
```

Listing 7.22. LCD module

```
#define ENCODER_PORT PORTD /* port data register */
#define ENCODER_DDR DDRD  /* port data direction register */
#define ENCODER_PIN PIND  /* port input pins register */
#define ENCODER_A  PD3    /* rotary encoder A signal */
#define ENCODER_B  PD1    /* rotary encoder B signal */
```

Listing 7.23. Rotary Encoder

```
#define KEY_PORT   PORTD  /* port data register */
#define KEY_DDR    DDRD  /* port data direction register */
#define KEY_PIN    PIND   /* port input pins register */
#define KEY_INC    PD3    /* increase push button (low active) */
#define KEY_DEC    PD1    /* decrease push button (low active) */
```

Listing 7.24. Increase/decrease push buttons)

7.8. MK-328

- ATmega328, 8 MHz Takt; ST7565 Farbdisplay (Bit-Bang-SPI)
- external 2.5V voltage reference (TL431)
- settings provided by brunosso@EEVblog

```
#define HW_REF25
```

Listing 7.25. Hardware Options

```
#define LCD_ST7565R
#define LCD_GRAPHIC              /* graphic display */
#define LCD_SPI                  /* SPI interface */
#define LCD_PORT   PORTD         /* port data register */
#define LCD_DDR    DDRD          /* port data direction register */
#define LCD_RESET  PD0           /* port pin used for /RES (optional) */
#define LCD_CS     PD5           /* port pin used for /CS1 (optional) */
#define LCD_A0     PD1           /* port pin used for A0 */
#define LCD_SCL    PD2           /* port pin used for SCL */
#define LCD_SI     PD3           /* port pin used for SI (LCD's data input) */
#define LCD_DOTS_X 128           /* number of horizontal dots */
#define LCD_DOTS_Y 64            /* number of vertical dots */
#define LCD_START_Y 0            /* start line (0-63) */
#define LCD_CONTRAST 11          /* default contrast (0-63) */
#define FONT_8X8_VF              /* 8x8 font */
#define SYMBOLS_24X24_VFP        /* 24x24 symbols */
#define SPI_BITBANG              /* bit-bang SPI */
#define SPI_PORT   LCD_PORT      /* SPI port data register */
#define SPI_DDR    LCD_DDR       /* SPI port data direction register */
#define SPI_SCK    LCD_SCL       /* port pin used for SCK */
#define SPI_MOSI   LCD_SI        /* port pin used for MOSI */
```

Listing 7.26. LCD module

7.9. T3/T4

- ATmega328, 8MHz clock
- ST7565 display (bit-bang SPI)
- settings provided by tom666@EEVblog

```
#define LCD_ST7565R
#define LCD_GRAPHIC              /* graphic display */
#define LCD_SPI                  /* SPI interface */
#define LCD_PORT   PORTD         /* port data register */
#define LCD_DDR    DDRD          /* port data direction register */
#define LCD_RESET  PD4           /* port pin used for /RES */
#define LCD_A0     PD3           /* port pin used for A0 */
#define LCD_SCL    PD2           /* port pin used for SCL */
#define LCD_SI     PD1           /* port pin used for SI (LCD's data input) */
#define LCD_CS     PD5           /* port pin used for /CS1 (optional) */
#define LCD_DOTS_X 128           /* number of horizontal dots */
#define LCD_DOTS_Y 64            /* number of vertical dots */
#define LCD_START_Y 0            /* start line (0-63) */
#define LCD_CONTRAST 11          /* default contrast (0-63) */
#define FONT_8X8_VF              /* 8x8 font */
#define SYMBOLS_24X24_VFP        /* 24x24 symbols */
#define SPI_BITBANG              /* bit-bang SPI */
#define SPI_PORT   LCD_PORT      /* SPI port data register */
#define SPI_DDR    LCD_DDR       /* SPI port data direction register */
#define SPI_SCK    LCD_SCL       /* port pin used for SCK */
#define SPI_MOSI   LCD_SI        /* port pin used for MOSI */
```

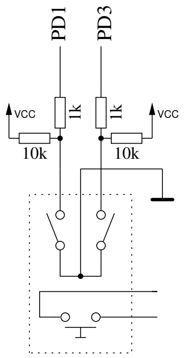
Listing 7.27. LCD module

Some T4 variants use a slightly different pin assignment:

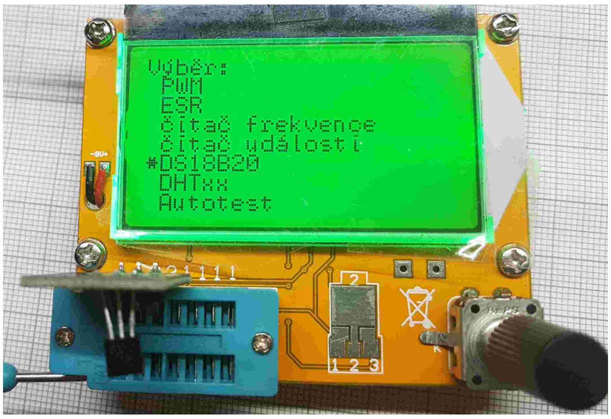
```
#define LCD_RESET  PD0           /* port pin used for /RES */
#define LCD_A0     PD1           /* port pin used for A0 */
#define LCD_SCL    PD2           /* port pin used for SCL */
#define LCD_SI     PD3           /* port pin used for SI (LCD's data input) */
```

Listing 7.28. LCD module

`T4 variants use Rotary Encoder` by Karl-Heinz Kübbeler [4]:



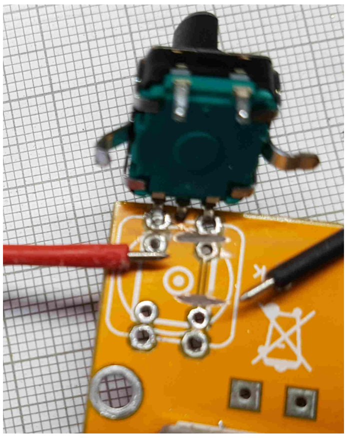
(a) Rotary scheme



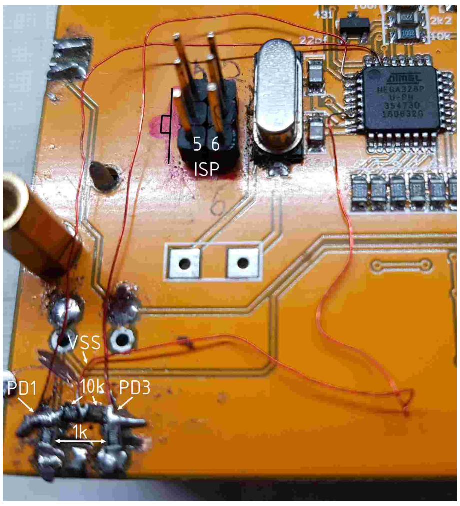
(b) T4 with Rotary Encoder. Font 6x8 ISO8859_2 CZ.

Figure 7.1. T4 with rotary encoder

Rebuild picture



(a) Front side



(b) Back side

Figure 7.2. T4 with rotary encoder

```
#define HW_ENCODER
#define ENCODER_PULSES 4
#define ENCODER_STEPS 24
#define HW_REF25
#define UREF_25     2495
#define HW_FREQ_COUNTER_BASIC
#define HW_EVENT_COUNTER
#define EVENT_COUNTER_TRIGGER_OUT
#define SW_PWM_SIMPLE
#define SW_INDUCTOR
#define SW_ESR
#define SW_DS18B20
#define SW_REVERSE_HFE
#define SW_DHTXX
#define UI_COMMA
#define UI_AUTOHOLD
#define UI_KEY_HINTS
#define POWER_OFF_TIMEOUT 30
#define SW_POWER_OFF
#define UI_ROUND_DS18B20
#define ONEWIRE_PROBES
```

Listing 7.29. Misc settings config.h:

7.10. Multifunktionstester TC-1 and family (T7)

ATmega324 (very poor pin assignment), 16MHz clock
later models may have an ATmega644
- ST7735 color display (bit-bang SPI)
- external 2.5V voltage reference (TL431)
- fixed IR receiver module
- boost converter for Zener check
(runs all the time, non-standard voltage divider 100k/12k)
- fixed adjustment cap
(in case of problems replace MLCC with 220nF film cap)
- powered by Li-Ion cell 3.7V
- sample testers provided by jellytot@EEVblog and joystik@EEVblog
- initial information provided by indman@EEVblog

Hints:
- Control MCU U4 needs to be replaced with a simple control circuit (TC1-Mod, see source repository for Hardware/Markus/TC1-Mod.kicad.tgz, 5μA standby current in total) or reprogrammed with a modified firmware (see https://github.com/atar-axis/tc1-u4).
- Set extended fuse byte to 0xfd (brown-out detection).
- If D2 (rectifier diode for Zener test voltage) gets hot replace it with a Schottky diode rated for 80V reverse voltage or higher, e.g. SS18.
- Replace C11 and C12 (filter caps for Zener test voltage) with a 10 or 22μF low-ESR electrolytic cap rated for 100V or higher because of the MLCC DC bias capacitance derating issue.
- Based on the LCD module used you might have to set LCD_FLIP_X instead of LCD_FLIP_Y.
- The TC-1 can't provide signal output (PWM/squarewave/etc.) on probe #2.
Use PD4 (OC1B) as dedicated signal output (add a resistor to limit current) and enable HW_FIXED_SIGNAL_OUTPUT in config.h.
- If you like to add a rotary encoder or in/decrease buttons please use PB5 (display's D/C) and PB6 (display's SDA).
- You can also get the frequency counter by using PB0 (T0) as input und adding a simple input stage.
- PD0 solder bridge (unused by m-firmware)

```
#define HW_REF25
#define HW_ZENER
#define ZENER_DIVIDER_CUSTOM
#define ZENER_R1   100000
#define ZENER_R2   12000
#define ZENER_UNSWITCHED
#define HW_IR_RECEIVER
#define HW_ADJUST_CAP
```

Listing 7.30. Hardware Options

```
#define HW_PROBE_ZENER
#define ZENER_VOLTAGE_MIN 1000 /* min. voltage in mV */
#define ZENER_VOLTAGE_MAX 40000 /* max. voltage in mV */
```

Listing 7.31. Misc settings

```
#define IR_PORT    PORTD   /* port data register */
#define IR_DDR     DDRD    /* port data direction register */
#define IR_PIN     PIND    /* port input pins register */
#define IR_DATA    PD3     /* data signal */
```

Listing 7.32. Pinout for fixed IR detector/decoder

```
#define ADJUST_PORT PORTC /* port data register */
#define ADJUST_DDR DDRC   /* port data direction register */
#define ADJUST_RH  PC6    /* Rh (470k) for fixed cap */
```

Listing 7.33. Pinout for fixed cap for self-adjustment

```
#define TP_ZENER   PA4    /* test pin with 10:1 voltage divider */
#define TP_REF     PA3    /* test pin with 2.5V reference */
#define TP_BAT     PA5    /* test pin with 4:1 voltage divider */
#define TP_CAP     PA7    /* test pin for self-adjustment cap */
```

Listing 7.34. Pinout for test probes

```
#define R_PORT     PORTC  /* port data register */
#define R_DDR      DDRC   /* port data direction register */
#define R_RL_1     PC0    /* Rl (680R) for test pin #1 */
#define R_RH_1     PC1    /* Rh (470k) for test pin #1 */
#define R_RL_2     PC2    /* Rl (680R) for test pin #2 */
#define R_RH_2     PC3    /* Rh (470k) for test pin #2 */
#define R_RL_3     PC4    /* Rl (680R) for test pin #3 */
#define R_RH_3     PC5    /* Rh (470k) for test pin #3 */
```

Listing 7.35. Pinout for probe resistors

```
#define POWER_PORT PORTD  /* port data register */
#define POWER_DDR  DDRD   /* port data direction register */
#define POWER_CTRL PD2    /* controls power (1: on / 0: off) */
```

Listing 7.36. Pinout for power control

```
#define BUTTON_PORT PORTD /* port data register */
#define BUTTON_DDR DDRD   /* port data direction register */
#define BUTTON_PIN PIND   /* port input pins register */
#define TEST_BUTTON PD1   /* test/start push button (low active) */
```

Listing 7.37. Pinout for test button

```
#define LCD_ST7735
#define LCD_COLOR                 /* color graphic display */
#define LCD_SPI                   /* SPI interface */
#define LCD_PORT   PORTB          /* port data register */
#define LCD_DDR    DDRB           /* port data direction register */
#define LCD_RES    PB4            /* port pin used for /RESX (optional) */
//#define LCD_CS     PB?            /* port pin used for /CSX (optional) */
#define LCD_DC     PB5            /* port pin used for D/CX */
#define LCD_SCL    PB7            /* port pin used for SCL */
#define LCD_SDA    PB6            /* port pin used for SDA */
#define LCD_DOTS_X 128            /* number of horizontal dots */
#define LCD_DOTS_Y 160            /* number of vertical dots */
#define LCD_OFFSET_X 2            /* enable x offset of 2 or 4 dots */
#define LCD_OFFSET_Y 1            /* enable y offset of 1 or 2 dots */
//#define LCD_FLIP_X                /* enable horizontal flip */
#define LCD_FLIP_Y                /* enable vertical flip */
#define LCD_ROTATE                /* switch X and Y (rotate by 90Grad) */
#define LCD_LATE_ON               /* turn on LCD after clearing it */
#define FONT_10X16_HF             /* 10x16 font */
#define SYMBOLS_30X32_HF          /* 30x32 symbols */
#define SPI_BITBANG               /* bit-bang SPI */
#define SPI_PORT   LCD_PORT       /* SPI port data register */
#define SPI_DDR    LCD_DDR        /* SPI port data direction register */
#define SPI_SCK    LCD_SCL        /* port pin used for SCK */
#define SPI_MOSI   LCD_SDA        /* port pin used for MOSI */
```

Listing 7.38. LCD module

7.11. Arduino MEGA pinout

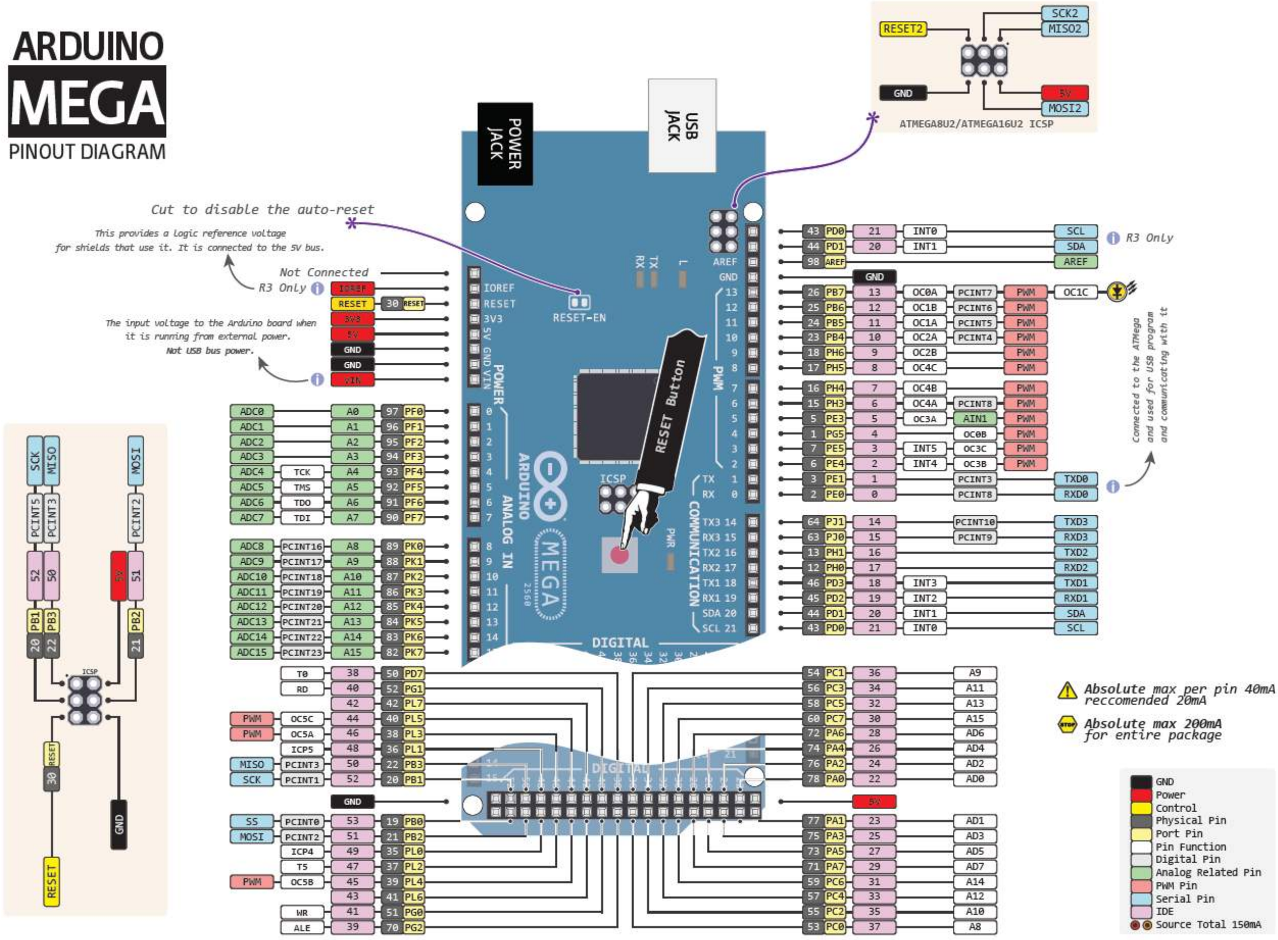


Figure 7.3. Arduino Mega

This image was copied from: https://duino4projects.com/arduino-mega-pinout-diagram/

Chapter 8 Programming the Component Tester

Translated with www.DeepL.com/Translator (free version)
In order to spare other colleagues the despair and „sleepless nights” that the author of this chapter suffered after he had acquired a clone tester without any AVR experience and wanted to „teach” it the German language, this chapter was written. The experience gained here should help other „willing” inexperienced ones, SUCCESSFULLY program their tester.

This opportunity is used to thank the author and developer of the Transistortester Karl-Heinz Kübbeler see [3] for his commitment and patience, because the following pages would never have been created without his help.

So that the translating of the firmware and burning into the MCU succeeds and at the same time . . . „the wheel would not have to be reinvented”, a part of the following pages was taken from the Description of the Transistor Tester by Karl-Heinz Kübbeler see [3].

So again . . . HUGE THANKS.

8.1. Configure the Component Tester

Please read chapter 1.8 on page 9.

8.2. Programming the microcontroller

The programming of the tester is controlled by the Makefile file.
The Makefile ensures that the software is compiled according to the options previously set in the Makefile. is translated according to the options previously set in the Makefile.

The result of the translation has the file extension .hex and .eep.
Usually the files are called ComponentTester.hex and ComponentTester.eep.

The .hex file contains the data for the program memory (flash) of the ATmega processor.

The .eep file contains the data for the EEPROM of the ATmega. Both files must be loaded into the correct memory.

Additionally the ATmega must be configured correctly with the „fuses”.
If you use the Makefile together with the program avrdude [9], you need no need to know the details of the fuses.

You only need to call "make fuses" if you don’t use crystal or you need to call need to call "make fuses-crystal" if you have a $8MHz$ crystal installed on the board.

If you are not sure about the fuses, leave them as they were set by the set by the factory and run the tester in this state. It may be that the program runs too slowly if you use the program data generated for $8MHz$-operation, but this is not the case. but this can be corrected later!

But wrong set fuses can prevent the later ISP programming.

8.3. Operation System Linux

Programming below Linux brings many advantages, because this OS was developed by experts who are oriented on the users’ wishes. In addition, the environment is available free of charge and perfectly maintained.

Another advantage is the security of OS itself but also when using the Internet.
The current editions are much easier to use than the competing OSes.
This guide should encourage all „non” Linux users to test it NOW by programming their tester with it.

As an example we use Linux Mint in the current version, which is available on the Internet. The installation is possible in different ways.

8.4. Use under Linux

on newly installed OS.
For those who don't like to write, Linux offers an easier way.

Copy this manual to a USB stick and open it in Linux.
Then move the mouse to the name of the document, press the left mouse button here and drag the document to the left edge of the screen until a possible frame appears. Now release the mouse. The guide now takes the left half of the screen.
The next step is to press Ctrl+Alt]+ t simultaneously to open the command window. This is now moved to the right edge of the screen in the same way.

8.5. Install program packages

Now we need internet access.
In order to program the tester, we first need to install program packages:

'binutils-avr', 'avrdude', 'avr-libc' and 'gcc-avr'. Further the version management 'git'.
Now navigate to this page in this document, up to this text:

sudo apt-get install avrdude avr-libc binutils-avr gcc-avr git
Highlight the text with the left mouse button pressed, move the mouse to the cursor of the right command window and press the middle mouse button (scroll wheel) to paste **further abbreviated as SW**.

After confirming with Enter, 'sudo' will ask for user password.

Now all software packages will be downloaded and installed by 'apt'. You may have to confirm questions in between with J.
Please note that Linux is case sensitive.
So don't answer with j but with J! You can check if the version control system 'git' was installed successfully with the command:

git version
command. The program should respond with the output of its version number.

8.6. Download the sources

and the documentation from the Git archive is achieved with a statement:

git clone https://github.com/Mikrocontroller-net/transistortester
The files are now in the Linux [personal folder] on (/home/„user") under the name „transistortester". Checking the presence. Open the terminal window,

ls
and confirm with Enter or ↲. To download new updates, it is enough in the future:

cd ~/tra
followed by Tab and enter ↲, and now in this directory

git pull
followed by ↲.

8.7. Using the interfaces

prepare for the user (user).
USB devices can be detected by typing 'lsusb' in the command window. Enter 'lsusb' first without and then with a connected USB programmer.
A comparison of the results locates the USB programmer.

A sample of the result of lsusb you can see here:

```
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 046d:c050 Logitech, Inc. RX 250 Optical Mouse
Bus 002 Device 058: ID 03eb:2104 Atmel Corp. AVR ISP mkII
Bus 002 Device 059: ID 2341:0042 Arduino SA Mega 2560 R3 (CDC ACM)
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub}
```

A Device 58 is detected here a a AVR ISP mkII type (DIAMEX ALL-AVR). The also detected USB device 59 is a USB-serial type device. The ID 03eb is a vendor ID and the ID

2104 is a product ID of this ISP-programmer. Both ID's are required for a entry in the file /etc/udev/rules.d/90-atmel.rules and can be added with:
sudo xed /etc/udev/rules.d/90-atmel.rules
In this example, the 90-atmel.rules file consists of one line:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="2104", MODE="0660",
GROUP="plugdev"
```

This entry allows access to the device for members of the 'plugdev' group.
To use most programmers, the following text in 90-atmel.rules is recommended:

```
# Copy this file to /etc/udev/rules.d/90-atmel.rules
# AVR ISP mkII - DIAMEX ALL-AVR
SUBSYSTEM=="usb", ATTRS {idVendor}=="03eb", ATTS {idProduct}=="2104", MODE="0660",
 GROUP = "plugdev",
# USB ISP-programmer für Atmel AVR
SUBSYSTEM=="usb", ENV {DEVTYPE}=="usb_device", SYSFS {idVendor}=="16c0", MODE="0666",
 SYSFS {idProduct} == "05dc",
# USB asp programmer
ATTRS {idVendor}=="16c0", ATTRS {idProduct}=="05dc", GROUP="plugdev", MODE="0660"
# USBtiny programmer
ATTRS {idVendor}=="1781", ATTRS {idProduct}=="0c9f", GROUP="plugdev", MODE="0660"
# Pololu programmer
SUBSYSTEM=="usb", ATTRS {idVendor}=="1ffb", MODE="0666"
```

After the file has been created, the creation and content can be controlled with:
less /etc/udev/rules.d/90-atmel.rules
The USB device Arduino SA Mega 2560 system, also recognized as Device 59, generates one Access to the serial device „/dev/ttyACM0" for members of the 'dialout' group.

8.8. Group membership

Therefore the own user ID should be member of the group 'plugdev' as well as of the group of the group 'dialout'. The command:
sudo usermod -a -G dialout,plugdev $USER
should ensure the affiliation. Now you should be able to access both devices with avrdude. You can check it with the command: 'id'.
If you have problems you can also check the membership with:
menu » system administration » users and groups » ?password, now a window with two tabs will appear. If you click on your name in the Users tab, you will see your profile and group memberships on the right. With the button Add it is now possible to add new groups.

8.9. working environment

preparation.
First navigate to /transistortester/Software/Markus/ in the taskbar with green folder icon (Nemo), right click on ComponentTester-1.(highest no.)m.tgz
click, in the selection <unzip here> the folder decompress and close Nemo.
To keep the original and because the terminal window always opens in ../home/ "user", it is recommended to put there its working directory named **Mytester**.

Using the already known method, select the following directory and paste it in the terminal window with the SW.
cd ~/transistortester/Software/Markus/
After confirmation and command: 'ls' you see all packed folders (ending .tgz) and only one folder where this ending is missing -> so our (before unpacked) folder .
For the following two commands, insert them first **ONLY**, **without** Enter to press: !
cp -r 'MyT' Mytester/
Mark the directory of the needed model with the mouse. Now position the blinking cursor to the last character of the text 'MyT' using ← and delete these characters. After deleting the last character, press the SW key of the mouse. Only now use Enter. Now the working environment is created. The control of the existence and content is possible with:

diff 'MyT' Mytester/
where 'MyT', as before, must be substituted for the directory of the 'required tester model'. With the last statement:
ln -s ~/transistortester/software//Markus/Mytester ~/Mytester
the link to the working directory will be created. From now on you can reach this directory very easily with:
Ctrl+Alt]+t, cd Space My Tab Enter
and you are already in the required directory. With 'ls' you can see the content. Further it goes to the processing of Makefile with, in the meantime known instruction:
xed Ma Tab Enter
The most important thing here is to log in the PRESENT USB programmer. See the chapter 6.1.5, on the page 44, topic PROGRAMMER.

8.10. Building the firmware

After editing the Makefile, config.h and config-<MCU>.h please run 'make' or whatever toolchain you have to compile the firmware. This will create two files:

- ComponentTester.hex	firmware in Intel hex format
- ComponentTester.eep	EEPROM data in Intel hex format

The firmware will be written to the Flash and the EEPROM data to the EEPROM. The data contains two sets of default adjustment values, texts and tables. When you update the firmware and like to keep the old adjustment values in the EEPROM you can enable DATA_FLASH in config.h to move texts and tables into the firmware. In that case only the firmwmare needs be programed, the EEPROM stays untouched.
The Makefile provides following additional targets:

- clean	to delete all object and firmware files
- fuses	to set the ATmega's fuse bits (via avrdude)
- upload	to program the firmware and EEPROM data (via avrdude)
- prog_fw	to program only the firmware (via avrdude)
- prog_ee	to program only the EEPROM data (via avrdude)

Now only the joy remains over the achieved success.

Chapter 9 Changelog

9.1. v1.43m 2021-03

- Added R&D display driver for identifying display controllers.
- LCD_RD control signal is now optional for ILI9341, ILI9481, ILI9486 and ILI9488.
- Added modified driver for ST7735 semi-compatible displays (suggested by b0hoon4@gmail.com).
- Option to run Zener check during normal probing (HW_PROBE_ZENER, suggested by indman@EEVblog). Requires ZENER_UNSWITCHED. Also added corresponding remote control command (V_Z).
- Added option to specify a custom voltage divider for the Zener check (ZENER_DIVIDER_CUSTOM, ZENER_R1 and ZENER_R2).
- Removed option for low resolution in Zener check and also the ZENER_HIGH_RES configuration switch.
- Updated Russian texts (thanks to indman@EEVblog).
- Optional display of pulse duration for PWM generators (PWM_SHOW_DURATION, suggested by guest "hamburger" @mikrocontroller.net). - Source archive includes "dep" directory to make some IDEs happy (suggested by DAIRVINE@EEVblog).
- Renamed configuration switch SW_PROBE_COLORS to UI_PROBE_COLORS.
- New feature to color titles differently (UI_COLORED_TITLES) and also cursor and key hints (UI_COLORED_CURSOR).
- Updated Romanian texts (thanks to Dumidan@EEVblog).
- Updated alternative Polish texts (thanks to Jacon@EEVblog).
- Updated Spanish texts (thanks to pepe10000@EEVblog).

9.2. v1.42m 2020-12

- Updated 6x8 fonts (thanks to Bohu).
- New ISO8859-2 based 6x8 font (FONT_6X8_ISO8859_2_HF, thanks to Bohu).
- Added LC meter hardware option (HW_LC_METER). It's based on a simple LC oscillator circuit, similar to some inexpensive LC meter kits.
- Changed Display_Value() to support also fempto (f).
- When CYCLE_MAX is set to 255 the automatic power-off is disabled and the tester keeps running until it's powered off manually.
- Fixed issue with display of strange frequencies in the extended frequency counter in a specific situation (reported by Szybkijanek@EEVblog).
Also changed display output of a missing signal or a too low frequency from '0Hz' to '-'.
- The basic counter displays now a '-' instead of '0Hz' in case of a missing signal or a too low frequency.
- Added option to display I_C/I_E test current for hFE measurement SW_HFE_CURRENT, (suggested by Obelix2007@EEVblog).
Also added corresponding remote control commands (I_C and I_E).
- Added Romanian texts (thanks to Dumidan@EEVblog).
- Added configuration switch for ESR tool (SW_ESR_TOOL, suggested by indman@EEVblog).
- Changed text output of E series norm values to also display unit (suggested by indman@EEVblog).
- Updated alternative Polish texts (thanks to Jacon@EEVblog).
- Last textline on display was simply overwritten when UI_KEY_HINTS is enabled (reported by Obelix2007@EEVblog).
Added function to wait for user feedback before clearing last line.
- Updated Spanish texts (thanks to pepe10000@EEVblog).

- Corrected error in "#define" for E96 norm values (reported by Obelix2007@EEVblog).
- Added driver for ILI9488 based displays 8/16 bit parallel and 4-line SPI, (thanks to Bohu for a sample display).

9.3. v1.41m 2020-09

- New remote command "MHINT" for hints about measurements
(at the moment just the test circuit type for h_FE).
- Added auto mode (automatic update) to DS18B20 tool
(suggested by Obelix2007@EEVblog).
- Added additional discharge function to resolve ESR issue on some tester clones
(reported by indman@EEVblog). ESR value was way too high for cap >= 470μF connected to probes #1 and #2, in ESR tool, C monitor and RCL monitor. And for cap >= 4700μF in all modes. Affected testers are Hiland M644 and TC-1 variants.
- Updated degree character in font_16x26_hf.h and font_16x26_iso8859-2_hf.h
(changes provided by Bohu).
- Driver for OLED controller SH1106 3-wire SPI, 4-wire SPI and I2C;
(thanks to Old-Papa for providing a sample display).
- Fixed issue with missing setup of optional /RES line for I2C in SSD1306 driver.
- Added output of test circuit type for hFE.
- Changed output of hexadecimal values to lowercase by default and added switch to enable uppercase (UI_HEX_UPPERCASE).
- Option for OneWire related tools to also read and display the ROM code
ONEWIRE_READ_ROM, (suggested by indman@EEVblog).
- Swapped DQ and Vcc probes for OneWire bus to match sensor pinout
(suggested by indman@EEVblog).
- Added detection for Schottky-clamped BJTs (SW_SCHOTTKY_BJT).
- Improved detection of depletion mode FETs by adding filter for Schottky- clamped BJTs.
- Optimized text line management for clearing the display.
- Added special display driver for identifying display controllers.
- Added R/C/L monitor SW_MONITOR_RCL, (suggested by indman@EEVblog).
- Fixed problem with jumping inductance values in L and R/L monitor
(reported by indman@EEVblog).
- Added measurement of V_GS(off) for depletion-mode FETs (suggested by joshto@EEVblog).
Also added "V_GS_off" command to remote command option.
- Added support for 8-bit parallel bus to ILI9341 driver. Also added a configuration switch for ILI9341 with diabled extended command set LCD_EXT_CMD_OFF,
(thanks to Bohu for providing a sample display).
- Added 16x26 cyrillic font FONT_16X26_WIN1251_HF, (provided by Yuriy_K@VRTP.RU).
- Fixed missing μ (micro) in multiple ISO8859-2 fonts
(reported by indman@EEVblog and Obelix2007@EEVblog).
- Fixed transposed digits for ATmega 2560 in Makefile (reported by Bohu).
- Colored battery status for color displays (suggested by indman@EEVblog).
- Replaced alternative 8x16 win1251 font with an updated version provided by indman@EEVblog.
- Fixed issue with validity check of low value resistors in CheckResistor()
(reported by indman@EEVblog).
- Updated SmallResistor(), MeasureInductance() and GetGateThreshold() to support
ADC_LARGE_BUFFER_CAP.
- Added missing update of ADC reference source to MeasureInductance() and GetGateThreshold().
- Added option to select the adjustment profile after powering on UI_CHOOSE_PROFILE,
(suggested by Bohu).

- Added dedicated monitoring tools for resistors and inductors
SW_MONITOR_R, SW_MONITOR_L, (suggested by indman@EEVblog).
- Updated Spanish texts (thanks to pepe10000@EEVblog).

9.4. v1.40m 2020-06

- Added drivers for ILI9481 and ILI9486 based displays.
- Fixed logic error for color displays when color feature is disabled.
- Added configuration switch for color displays to swap red and blue color channels (LCD_BGR).
- Improved R/L monitor to reduce spread of measurement values (reported by indman@EEVblog). Might not help in all cases.
- Fixed "#ifdef" issue for Display_HexByte() and Display_HexDigit() (reported by AlcidePiR2@EEVBlog).
- Support for ATmega 640/1280/2560.
- The detection of the rotation direction in ReadEncoder() was reversed. Changed to correct direction and updated settings in config_<MCU>.h and Clones.
- Fixed initialisation issue for hardware SPI.
- Added alternative mode to Zener check in case the boost converter runs always or the circuit hasn't one (ZENER_UNSWITCHED, suggested by indman@EEVblog).
- Configuration switch for 100nF AREF buffer cap instead of 1nF to deal with some MCU boards (ADC_LARGE_BUFFER_CAP).
- Added alternative cyrillic fonts 8x16 (FONT_8X16ALT_WIN1251_HF) and 8x8 (FONT_8X8ALT_WIN1251_VF, thanks to indman@EEVblog).
- Updated Russian texts (thanks to indman@EEVblog).
- Option to use a manual power switch instead of the default soft-latching one (POWER_SWITCH_MANUAL).
- Added detection of two short presses of the center bar in ReadTouchScreen().
- Fixed fault in Touch_CharPos() regarding TOUCH_FLIP_X/TOUCH_FLIP_Y (reported by Bohu).
- Replaced some ISO8859-2 fonts with updated versions provided by Bohu.
- Option for checking resistors for E series norm value (SW_R_E*). Also for capacitors (SW_C_E*) and inductors (SW_L_E*).

9.5. v1.39m 2020-03

- Added Polish texts based on ISO 8859-2 (Thanks to Jacon).
- Fixed issue with preprocessor macro in SPI.c and syntax error in ADS7843.c (reported by Bohu).
- Added option to store firmware data to Flash instead of EEPROM (DATA_FLASH, suggested by Vitaliy).
- Renamed cyrillic fonts to "win1251" and replaced Czech fonts with ISO8859-2 fonts provided by Bohu.
- Display of font for test purposes (SW_FONT_TEST).
- OneWire scan tool for listing ROM codes of connected devices (SW_ONEWIRE_SCAN).
- Option for probe-pair specific resistance offset (R_MULTIOFFSET, suggested by Vitaliy).

9.6. v1.38m 2019-12

- Optional rounding of temperature value for DS18B20 (UI_ROUND_DS18B20, suggested by Obelix2007@EEVblog)
- Support for DHT11, DHT22 and compatible sensors (SW_DHTXX, thanks to indman@EEVblog and Obelix2007@EEVblog for testing).
- Added two thin cyrillic fonts (Thanks to Andrey@EEVblog).
- Changed output of BJTs to show V_BE and hFE also in case of a B-E resistor. Adapted remote commands accordingly as well.
- Added alternative Czech texts and several fonts with Czech characters (thanks to Bohu).
- Added tools for monitoring R/C/L on probes #1 and #3 (SW_MONITOR_RL and SW_MONITOR_C, suggested by indman@EEVblog).
- Added option for trigger output for event counter (suggested by Bohu).
- Updated Czech texts (thanks to Bohu).

- Added option to disable hFE measurement with common collector circuit and Rl as base resistor (NO_HFE_C_RL) to cope with some testers (issue reported by Obelix2007@EEVblog).
- Added option to output Zener voltage in high resolution (ZENER_HIGH_RES, suggested by Andbro@EEVblog).
- Improved OneWire_Probes() to minimize misdetection.
- Updated Russian texts (thanks to indman@EEVblog).
- Updated Spanish texts (thanks to pepe10000@EEVblog).

9.7. v1.37m 2019-09

- Fixed error in DS18B20 _Tool() when ONEWIRE_IO_PIN is enabled (reported by bm-magic).
- Fixed problem displaying the watchdog error message on color displays.
- New function: Event counter (HW_EVENT_COUNTER, suggested by bm-magic).
- The simple frequency counter now uses TestKey() for user input. To the Short press the button twice (was previously a button press).
- option to display the inverse hFE value of transistors (SW_REVERSE_HFE, suggestion from towe96 @ EEVblog). Also remote control commands extended by the command "‘h_FE_r"’.
- Bitclock setting (BITCLOCK) for avrdude in Makefile (suggestion of bm-magic).
- Problem with TRIAC detection in case of too high I_GT in Q3 or too high I_H fixed (I_GT problem reported by petroid).
- Text Tester_str, PWM_str, Hertz_str and CTR_str in language specific Header files moved (suggested by indman @ EEVblog).
- output of frequency values (hertz) changed to fixed string (previously "‘H"’ as unit for DisplayValue() plus additional "z").
- Accessibility Option (UI_KEY_HINTS). Currently only "‘ Menu/Test"’(suggestion of carrascoso@EEVblog).
- Polish texts updated (C szpila@EEVblog).
- Russian texts (Thanks to indman@EEVblog).
- Spanish texts (thanks to pepe10000@EEVblog).

9.8. v1.36m 2019-05

- Added optional 6x8 font to ST7565R driver.
- Added optional mainmenu item to power off tester (SW_POWER_OFF).
- Integrated battery monitoring into TestKey() and Zener_Tool().
- Added detection of two short presses of the test key to TestKey() and removed redundant functionality in multiple functions to reduce firmware size.
- Driver for ST7036 based displays (4-bit parallel & 4-wire SPI, untested).
- Moved power control and battery monitoring to dedicated functions for better integration with other functions.
- Driver for PCF8814 based displays (3-line SPI, thanks to Mahmoud Laouar for testing).
- Driver for STE2007/HX1230 based displays (3-line SPI).
- Fixed bug in LCD_Clear() of PCD8544 driver.
- Added missing cyrillic font to ST7565R driver (reported by Andrey@EEVblog).
- Updated font_8x16_cyrillic_vfp.h (thanks to Andrey@EEVblog).
- Fixed issue with bad character in font_HD44780_cyr.h.

9.9. v1.35m 2019-02

- Added option to use probe pair specific capacitance offsets instead of an average offset for all probes (CAP_MULTIOFFSET).
- Corrected pin definition for ST7920 4-bit parallel mode in config_644.h (reported by jakeisprobably@EEVblog).
- Added support for 3-wire SPI to SSD1306 driver.
- Extended SPI driver to support sending 9 bit frames (bitbang only).
- Fixed issue with increasing deviation of resistors between 7k5 and 19k5Ω in CheckResistor() (reported by Vitaliy).

- Added alternative delay loop in IR_Send_Pulse() which is enabled by SW_IR_TX_ALTDELAY (thanks to Vitaliy).
- The configuration switch for additional IR protocols SW_IR_EXTRA was replaced by SW_IR_RX_EXTRA for the receiver/decoder and SW_IR_TX_EXTRA for the sender.
- Fixed issue with missing newline in Display_NextLine() for remote commands.
- Changed output for SIRC in IR_Decode() to reflect native protocol (suggested by Vitaliy).
- Fixed bug in IR_Send_Code() for SIRC-20 (reported by Vitaliy).
- Updated var_russian.h (thanks to indman@EEVblog).
- Added automatic power-off for auto-hold mode (POWER_OFF_TIMEOUT).
- Separated pin configuration for test push button and power control (CONTROL_PORT – > POWER_PORT and BUTTON_PORT).
- Several minor improvements.

9.10. v1.34m 2018-10

- Added leakage check for capacitors.
- Changed default value for RH_OFFSET to 350 Ohms.
- Fixed missing menu entry for fixed IR receiver module.
- Polish texts (thanks to Szpila).
- Display driver for output via VT100 serial terminal.
- Support for temperature sensor DS18B20.
- Driver for OneWire bus.

9.11. v1.33m 2018-05

- Fixed orientation of TRIAC symbol in symbols_32x32_hf.h.
- Added remote commands for automation (via TTL serial).
- The x & y offsets for the ST7735 driver can be changed now.
- Entering the menu by a short circuit of the probes is an option now (UI_SHORT_CIRCUIT_MENU).
- Fixed problem with discharge relay when using rotary encoder.
- Added configuration switch to disable MCU sleep modes.
- Added RX support to TTL serial driver (bit-bang & hardware USART).
- Fixed error in serial text output and added serial output for results of the optocoupler check.
- Danish texts (provided by glenndk@mikrocontroller.net)
- Settings for capacitor correction factors.

9.12. v1.32m 2018-02

- Additional output of components found to TTL serial interface.
- Driver for TTL serial interface (hardware and bit-bang).
- Updated var_russian.h (thanks to indman@EEVblog).
- Added support for X&Y offsets to ST7735 driver.
- Changed configuration of battery monitoring.
Added switches to disable battery monitoring and to support an unmonitored external power supply.
- Added configuration switch to reverse operation mode selection at startup (UI_AUTOHOLD).
- Improved filter for Germanium BJTs with high leakage current in detection function for depletion mode FETs.
- Added fancy pinout to PCD8544 driver.
Also fixed error in the PCD8544 driver's function LCD_CharPos() for rotated output.
- Improved functions for fancy pinout of 3-pin semiconductors and moved some functions to display.c. Output of pinout on separate screen if required.
- Added indicator for usage of external voltage reference (Show Values).
- Improved IR decoder and added optional protocols.
- Added more protocols to IR RC transmitter.

9.13. v1.31m 2017-12

- New tool: IR RC transmitter.
- Added support for dedicated signal output via OC1B, when OC1B isn't used for test pin #2's probe resistor.
- Changed battery monitoring settings to support also other power options.
- Driver for SSD1306 based graphic OLED modules.
- Color support for item selection (menues and tools).
- Driver for ILI9163 based graphic color LCD modules.
- Fixed tiny issue in squarewave signal generator.
- Added support for 180° rotated output to PCD8544 LCD driver.
- Fixed edit error in Servo_Check().

9.14. v1.30m 2017-10

- Option to use comma instead of dot to indicate a decimal fraction.
- Support for extended frequency counter (buffered frequency input, LF crystal, and HF crystal).
- Minor improvements for basic frequency counter.
- Fixed gate time issue in frequency counter for frequencies below 10kHz when the MCU runs at 20MHz.
- Modified ESR measurement to support a shorter charging pulse, i.e. ESR can be measured for caps starting at 10nF. If you prefer the old method, you can enable that one alternatively.
- Fixed bug in the probes' short circuit detection.
- Added supported for 180° rotated output to ST7920 LCD driver.

9.15. v1.29m 2017-07

- Added touch screen support and driver for ADS7843 compatible touch controllers.
- Fixed bug in contrast setting for PCD8544.
- Fixed silly error in CheckSum().
- Driver for ST7920 based LCD modules with 64x128 pixels.
- Optimized SmallResistor() and changed detecton logic in CheckResistor() to cope better with low value resistances and possible probe contact issues.
- Changed control logic and treshold for Darlington BJTs in Get_hFE_C() to fix issue with some NPN types.
- Global driver for SPI bus. Modified display drivers and configuration accordingly.
- Italian text provided by Gino_09@EEVblog.
- Support for HD44780 with Cyrillic font by hapless@EEVblog.

9.16. v1.28m 2017-04

- Increase/Decrease push buttons as alternative for a rotary encoder (HW_INCDEC_KEYS).
- Added reset to default frequency to squarewave generator.
- Further improvements of the detection of the rotary encoder's turning velocity (ENCODER_STEPS). Also changes to functions making use of the turning velocity.
- Added reset to default values to alternative PWM generator.
- Russian text provided by indman@EEVblog (only 8x16 font, horizontally aligned)
- Added support for fixed cap for self adjustment of voltage offsets.
- Fixed potential bug in the V_ref offset handling in SmallCap().
- Added configuration switch for ST7735 based displays to start with a cleared display (no random pixels).

9.17. v1.27m 2017-02

- Added high current meassurement to GetLeakageCurrent() for CLDs.
Thanks to texaspyro@EEVblog for sending a few sample diodes.
- Fixed bug in MilliSleep().
- Fixed issue with large inductance in diode detection.
- Compensation for inductance measurement in the mH range.
- Support for PCF8574 based LCD backpacks in HD44780 driver.

- Driver for bit-bang and hardware I2C.
- Fixed bug in the handling of the variable pin assignment for HD44780 based displays.
- Color support for probe pinout of several tools.
- Check function for RC servos.
- Alternative PWM generator with variable frequency and ratio.
Requires rotary encoder and large display.
- Output of R_DS for MOSFETs and Vf for their body diode.
- Support for fixed IR receiver module in IR RC detector/decoder.
- Dropped edition in name, because the Classic edition is obsolete by now.

9.18. v1.26m 2016-12

- Added compensation for inductance measurement (more work required).
- Added Spanish texts. Translation provided by pepe10000@EEVblog.
- Changed FrequencyCounter() to support also ATmega 324/644/1284.
- Fixed problem in inductance measurement logic. Reported by indman@EEVblog.
- Fixed error in voltage reference handling for ATmega 324/644/1284.
- Improved detection of turning velocity of rotary encoder to cope with different values of pulses per step or detent.
- Added hardware SPI to all drivers for SPI based displays.

9.19. v1.25m 2016-09

- A lot of changes to support the ATmega 324/644/1284.
- Modified test resistor management to support variable port pins.
- Added software option for probe color coding.
- Centralized color management.
- Added file listing settings for various tester versions/clones.
- Fixed small issue with 24x24 VP symbol bitmap in config.h.
Reported by lordstein@EEVblog and hapless@EEVblog.

9.20. v1.24m 2016-08

- Measurement of self-discharge leakage current for caps $>4.7\mu$F.
- Added type detection logic for BJTs with diode on the same substrate.
- Improved leakage current measurement to support currents in the nA range.
The leakage will be shown for diodes and BJTs, when it's larger than 50nA.
- Improved the display of instrinsic/flyback diodes for transistors to check for the proper diode (pins and polarity).
- Fixed an error in the display of a BJT's flyback diode.
- Added a function for searching a specific diode and adapted several functions to use the new diode search instead of the old local search.
- Improved detection of diodes to support also Germanium types with very low Vf at low currents.
- Fixed problem with LCD_ClearLine(0) for ILI9341 and ST7735.
- Improved detection of depletion mode FETs to exclude Germanium BJTs with high leakage current. Added detection of FETs with low I_DSS. Added measurement of I_DSS.

9.21. v1.23m 2016-07

- Added support for PCD8544 and ST7735 based LCD modules.
Thanks to hansibull@EEVblog for sponsering a PCD8544 display.
- Extended wait.S for 20MHz MCU clock.
- Changed MeasureESR() to support also non 125KHz ADC clocks.
- Added detection of PUTs (Programmable Unijunction
Transistor) and UJTs (Unijunction Transistor).
Thanks to edavid@EEVblog for sending some UJTs for testing.
- Minor code optimization for ILI9341 and ST7565R.
- Fixed multi-page font problem for ST7565R, again.
- /RES port pin assigned for ILI9341 was ignored. Fixed that and also wrong delays for

hardware reset.
- Support of individual data lines for HD44780 based LCD displays.
- Support user-defined resistor divider for battery voltage.
- Added output of If for opto couplers.
- Changed probe pins of ESR tool to 1-3 to be compatible with k-firmware.
- Moved MCU specific global settings to dedicated header files.
Several minor changes to add support for ATmega664/1284.
- Updated Czech texts, thanks to Kapa.

9.22. v1.22m 2016-03

- Added opto coupler check with display of the LED's V_f, the CTR and t_on/ t_off delays (BJT types). Thanks to all_repair@EEVblog for some samples.

9.23. v1.21m 2016-01

- Licensed under the EUPL V.1.1
- Improved storage managment of adjustment values and added support for two adjustment profiles.
- Added detection and decoding of RC-6 to IR detector. Solved issue with test button when the IR receiver module is removed too early. Added configuration switch to disable current limiting resistor for Vs in case of a 5V only IR receiver module.

9.24. v1.20m 2015-12

- Added IR RC detector and decoder function (requires TSOP IR receiver module).
- Changed MainMenu() to reduce RAM usage.

9.25. v1.19m 2015-11

- Implemented a fancy pinout displaying symbols and probe numbers for 3 pin semiconductors.
- Added color support.
- Changed ShowDiode() to output the number of diodes directy (not via Show_Fail() anymore) when more than 3 diodes are found (hint by hapless@EEVblog).
- Extended LCD_ClearLine() in all display drivers to clear the remaining space of the current line
to speed up things, especially for graphic displays The idea is to display the text first and then to clear the remaining space, instead of clearing the complete line and then printing the text.
- Added display driver for ILI9341/ILI9342 based modules.
Thanks to Overtuner@EEVblog forum for providing two LCD modules.
- Fixed problem with μ/micro character in font files.
- Fixed character issue (when larger than 8x8) in LCD_Char() for ST7565R.
- Updated Czech texts, thanks to Kapa.
- Fixed a minor issue in MenuTool(), when rolling over from last to first item.

9.26. v1.18m 2015-07

- Improved MenuTool() to update items only when the list has changed. Otherwise just the selection indicator is updated.
- Fixed variable management bug in config.h.
- Added feature to reset to firmware defaults when powering on.
- Optimized functions for NVRAM management (values stored in EEPROM).
- Added driver functions for ST7565R grafic modules.
- Designed a simple framework which allows to add different display modules or controllers.
Moved high level display functions to display.c. Each controller got a dedicated source and header file including controller specific functions. Adapted old HD44780 to new framework.
- Changed UI to handle multiline displays in a flexible way.
- Removed everything specific to ATmega168 (too small ;).
- Optimized feedback processing in MenuTool().
- Forked a new firmware edition, which also supports graphic displays, named

"Trendy Edition". The old firmware is called "Classic Edition" now.

9.27. v1.17m 2015-02

- Improved CheckDiode() to support measured Vcc in resistor check and fixed detection issue for resistors around 2k if the optional DC/DC boost converter is installed (HW_ZENER).
- Fixed some incorrect comments.
- Cleaned up integer data types.

9.28. v1.16m 2014-09

- Added test for rotary encoders.
- Fixed some minor issues in MeasureInductance() to increase accuracy.
- Changed ShowAdjust() to display absolute value of Vcc and the internal bandgap reference (suggestion from Vlastimil Valouch).
- Some minor improvements.

9.29. v1.15m 2014-09

- Improved TestKey() to detect the dynamic turning velocity of a optional rotary encoder.
- Added a squarewave signal generator with variable frequency.
- Changed MeasureInductance() to return time in ns and adapted processing in MeasureInductor() (thanks to Vlastimil Valouch).

9.30. v1.14m 2014-08

- Changed user interface for rotary encoder.
- Fixed compiler warning about R_Pin2 in ShowDiode() (thanks to Milan Petko).
- Resistors between 1.5k and 3k Ohms were detected as double diodes. Changed tolerance of resistor test in CheckDiode() (thanks to nessatse).
- Modified ShortCircuit() to allow user to abort creating a short circuit in case of any problem.
- Added frequency counter (hardware option).

9.31. v1.13m 2014-07

- Added Czech texts, thanks to Kapa.
- ESR and PWM tools display the probes pins used.
- Improved handling of pre-compiler directives for extra features.
- Added support for rotary encoders for the user interface.

9.32. v1.12m 2014-03

- Fixed umlaut problem for German UI (thanks to Andreas Hoebel).
- Added ESR measurement for capacitors $>0.18\mu$F.
- Optimized display output handling to save some bytes Flash.

9.33. v1.11m 2014-03

- Improved pinout detection for Triacs (G and MT1) and changed display to report MT1 and MT2.
- Added pinout display function for semiconductors and changed output to "123=" style for better readabilty.
- Optimized several component output functions.
- Improved the BJT check to detect transistors with a built-in freewheeling diode on the same substrate (creating a parasitic BJT). The BJT output prints a '+' behind the BJT type for such transistors.
- Modified diode display to support possible BJTs with protection diode and low value B-E resistor. Those are detected as dual diodes. The B-E resistor will be shown to signal that special case.
- Modified BJT display to support base-emitter resistors. If a B-E resistor is found hFE and V_BE will be skipped since both values can't be measured in a reasonable way in that case.
- Improved diode test to detect body diodes of dep-mode FETs.
- Fixed detection problem of drain and source for depletion-mode FETs.

- Added detection of symmetrical drain and source for depletion-mode FETs.
- Vth is negative for p-channel FETs now.
- Added measurement of V_GT for SCRs and Triacs.
- Due to flash size constraints the PWM tool is now available for the ATmega328 only.

9.34. v1.10m 2013-10

- Added support for external 2.5V voltage reference (hardware option).
- Added support for relay based cap discharging (hardware option).
- Changed good-bye message into welcome message to help to detect problems with the brown-out voltage of the MCU and to mitigate a voltage drop caused by an optional DC boost converter at startup.
- Added Zener tool (hardware option).
- The main menu got an exit function in case the menu was entered by mistake.
- Support of 16MHz MCU clock.

9.35. v1.09m 2013-07

- Added IGBT detection.
- Added a sanity check for MOSFETs.
- The hFE measurement for BJTs considers the leakage current in common emitter configuration.
- For MOSFETs the direction of the intrinsic diode is shown.
- Fixed problem with swapped drain and source pins for enhancement mode MOSFETs.
- Added workaround for Makefile problems with some IDEs. Some important Values can be defined in config.h too.

9.36. v1.08m 2013-07

- Since the SmallResistor() measurement can't give correct DC resistance values for some inductors a problem detection was added to CheckResistor() to keep the original high resistance measurement result.
- Added inductance measurement (ATmega328/P only)
- Minor improvements for the display of diodes and BJTs.
- Added leakage current measurement.
- Fixed problem with germanium BJTs with a high leakage current. They were detected as p-channel JFETs.
- Renamed some functions, clarified and added some remarks.

9.37. v1.07m 2013-06

- Optimized diode display function and added display of low current Vf.
- Improved the diode detection. Caps and resistors are excluded much better. Also the cap probing is skipped for diodes to speed up probing.
- Fixed an array overflow bug in CheckResistor().
- Improved cursor display logic to tell user if another information follows or the probing loop is reentered.
- Improved UI of PWM tool to prevent exit by mistake (double key press required now).
- Added a generic menu function and adapted all menus (changed layout!).
- TestKey() provides a nice blinking cursor now.

9.38. v1.06m 2013-03

- Several minor improvements and cleanups.
- Expanded TestKey() to inform the user about an expected key press.
- Improved TestKey() function to be more responsive for short key presses.
- Added a PWM tool to generate a PWM signal with different frequencies and duty ratios.
- Implemented a sleep function to reduce power usage of the tester. On average the power usage is nearly halfed (excluding the LCD backlight).
- Improved discharge function. If the discharge of a component fails, the concerned probe pin and the remaining voltage are displayed. That will help to detect a too low value for CAP_DISCHARGED.

- Added the capability to set error types.

9.39. v1.05m 2012-11

- Moved LargeCap_table[] and SmallCap_table[] from EEPROM to flash to reduce EEPROM usage. The size for a German firmware exceeded the 512 bytes of an ATmega168s EEPROM.

9.40. v1.04m 2012-11

- Added a simple logic to the output function for diodes to prevent the measurement of capacitance for antiparallel diodes.

9.41. v1.03m 2012-11

- Fixed detection problem of power diodes with high leakage current (mistaken for resistors).
- Fixed compiler warnings about possible uninitialized variables. That increased the firmware size by about 44 bytes :-(

9.42. v1.02m 2012-11

- Added upper limit for resistance of probe leads in the self adjustment function 1.00Ω.
- Selftest and adjustment functions perform a short circuit test before running the main part and return feedback now.
- The mainmenu gives feedback about success/failure of the selected action.

9.43. v1.01m 2012-10

- Added a checksum for adjustment values stored in the EEPROM and wrote a function to validate the checksum.
- Added a measurement function for small resistors (resolution: 0.01Ω).
- Extended self adjustment to support an auto-zero for the resistance of the probe leads.
- CheckResistor() runs an extra measurement for small resistors ($<10\Omega$).
- Added a function to compare two scaled values.
- Adapted several functions to support variable scaling of values.

9.44. v1.00m 2012-09

- in the following changes were implemented:
- Simple menu for selection of self-test, self-tuning,
- saving the Adjustment values in the EEPROM and display of the adjustment values.
- hFE changed from 16 to 32 bits (no more 65k limit).

9.45. v0.99m 2012-09

- The first published version based on Karl-Heinz's has been published.

Bibliography

[1] http://www.mikrocontroller.net/articles/AVR-Transistortester *Online documentation of the Transistortester,* Online Article, 2009-2011

[2] Markus Frejek http://www.mikrocontroller.net/topic/131804 *thread of Markus Frejek, Forum, 2009.*

[3] http://www.mikrocontroller.net/articles/AVR_Transistortester *Online documentation of the Transistortester, Online Article* Online Article, 2012

[4] http://www.mikrocontroller.net/topic/248078 *Thread from Karl-Heinz K.,* 2012

[5] https://github.com/svn2github/transistortester/blob/master/Doku/trunk/pdftex/german/ttester.pdf
Online documentation of the Transistortester, Online Article

[6] - https://github.com/Mikrocontroller-net/transistortester/tree/master/Software/Markus
Complete software collection 2012-2019

[7] https://github.com/svn2github/transistortester/tree/master/Software/Markus
https://github.com/madires/Transistortester-Warehouse 2012-2019

[8] madires@theca-tabellaria.de
Author Mail

[9] http://www.mikrocontroller.net/articles/AVRDUDE *Online documentation of avrdude programmer interface,* Online Article, 2004-2011

[10] https://www.mikrocontroller.net/topic/248078
Main language is German, but English is ok too.

[11] https://www.eevblog.com/forum/testgear/($20-lcr-esr-transistor-checker-project/
Just Englisch.

[12] https://vrtp.ru/index.php?showtopic=16451
- Russian forum
- Info on various clones, like images, schematics and firmwares

[13] https://disk.yandex.ru/d/yW8xa5NJgUo5z
- (by indman@EEVblog)
- Info on various clones, like images, schematics and firmwares

[14] https://drive.google.com/file/d/1-IJA8uTcsCA_6SYHEuMydjfS2vNgmwdH/edit
- Instruction guide and files for WinAVR (by indman@EEVblog)