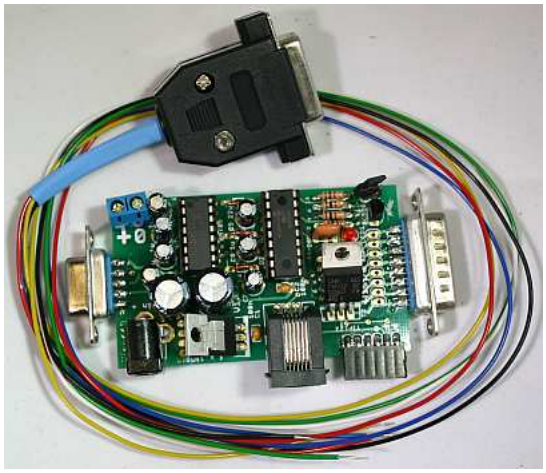
Wisp648

last modified 30-MAY-2010

Content



Introduction

Wisp648 is a serial port in-circuit programmer for Microchip FLASH PICmicro microcontrollers. The list of target PICs that are supported changes too often to include here, check the Wisp648 page at *http://www.voti.nl/wisp648* for an up-to-date list.

In-circuit programming

In-circuit programming (often abbreviated as ICP or ICSP) implies that you program your target chip while it is in its target circuit. There is no need to extract the chip from its circuit, put it in a programmer, and put it back in the circuit after programming. Instead you make a connection (a few wires) between the programmer and the target chip, and you can program it without even touching the circuit.

To make in-circuit programming possible the target circuit must comply with some rules. The details are explained in the *Programming details* section (p 3), but the summary is stated in the table below. When a resistor is mentioned the stated value is a minimum (higher is OK), and any circuit is OK that has at least the indicated impedance (for instance: an UL2803 has an input impedance of at least 2k7, so it can be connected directly to PGC, PGD or PGM pins, which require a minimum impedance of 1k). Pin 7 and 8 (RxD and TxD) are needed only when the *Serial pass-through* (p. 5) is used. The colors correspond to the wires supplied with the Wisp648 kit (but for readability grey is used in the table instead of black).

Wisp648 programming connector pin	Target PIC pin	Requirements
1	Ground (Vss)	-
2	+5V (Vdd)	When supplied by the target circuit: must be 5V +/- 5%, the Wisp648 will draw less than 50 mA.
3	PGC	Programmer connects directly to the target chip; 1k resistor between target chip and rest of the target circuit.
4	PGD	Programmer connects directly to the target chip; 1k resistor between target chip and rest of the target circuit.
5	MCLR	Programmer connects directly to the target chip; 33k resistor between target chip and rest of the target circuit. The programmer will 'load' this pin with a 1kΩ resistor connected to a 22µF capacitor. This might interfere with the applications' use of this pin.
6	PGM	For chips that have a PGM pin, either: • Target circuit pulls this pin low • Programmer connects directly to the target chip; 1k resistor between target chip and rest of the target circuit. For a target chip that does not have a PGM pin this line can be left unconnected.
7	RxD	Optional, programmer connects directly to the target chip; 1k resistor between target chip and rest of the target circuit.
8	TxD	Optional, no requirements

The previous table stated target PIC pin functions, not the pin numbers, because the numbers of the pins used vary from chip to chip. The table below works for most DIL chips, but the chip's datasheet is the final authority. Especially the UART pins (RxD and TxD) and the PGM pin vary.

Pin name	6/8-pin chips	8-pin chips	14-pin chips	18-pin chips	20-pin chips	New 28-pin chips	Old 28-pin chips	New 40-pin chips	Old 40-pin chips
Example chips	10F2xx	12Fxxx	16F630 16F688	16F6x8(A) 16F88	16F690 16F689	16F8x6(A)	16F57	16F8x7(A)	16F59
Ground (Vss)	7	8	14	5	20	8, 19	4	12, 31	5
+5V (Vdd)	2	1	1	14	1	20	2	11, 32	15,35
PGC	4	6	12	12	18	27	16	39	12
PGD	5	7	13	13	19	28	17	40	13
MCLR	8	4	4	4	4	1	28	1	14
PGM	-	-	-	10, 11*	-	24	-	36	-
RxD	-	-	-	7*	-	18*	-	26*	-
TxD	-	-	-	8*	-	17*	-	25*	-

The first column, 10F2xx, refers to the DIP version of these chips. These chips are also available in a tiny 6-pin SMD version (with a different pinout). The DIP version has two unused pins.

Microchip sells two popular programmer/debuggers, the ICD2 and the PICkit2. Beside the DB15 connector the Wisp648 also has two connectors that are compatible with these Microchip programmers. These connectors lack the PGM, RxD and TxD lines, because those functions are not present on the ICD2 and PICkit2. If you use these connectors it is probably to connect to an existing board that is ICD2 or PICkit2 compatible, so the pin assignment of these connectors is of

little interest to you. If you would want to make your own board with either of these connectors, check *Connector pin* assignments (p. 13) for the pin assignments.

Programming details

Three pins of a PIC chip are central to programming the chip: MCLR, PGC, and PGD (and of course +5V and ground). The MCLR pin is by default the reset pin of the chip. On newer chips this pin can also be configured as an input pin. The PGC and PGD functions are often on pins that are also I/O pins, on most chips RB6 and RB7.

On most chips programming mode is entered by the following sequence:

- reset the chip: +5V present, MCLR is low.
- MCLR quickly rises from low to Vpp (for most chips ~ 13V)

This sequence is called the Vdd-before-Vpp sequence. Vdd is the Microchip term for the +5V. Another sequence, required by some PIC chips, is the Vpp-before-Vdd sequence, which is described later. The Wisp648 requires (or generates) +5V. It can not be used with a lower voltage.

Note that MCLR must rise quickly. The exact timing requirement is in the programming specification of each PIC, but it is such that no capacitor on the MCLR pin can be tolerated. As a simple rule: the programmer must be connected directly to the MCLR pin, the rest of the circuit connects via a 33k resistor (10k will often be enough, but 33k is recommended). A reset-delay capacitor can still be present, but it must be on the other side of the 33k resistor.

When the chip is in programming mode PGC is used as clock line (programmer to target chip) and PGD as the data line (bidirectional) to enter programming commands and data into the chip, or to read the content of the chip. The programmer must be able to control these two pins: the rest of the circuit must not load these pins too heavily. As a simple rule: it is OK when the programmer is connected directly to the PGM and PGD pins, and the rest of the target circuit is connected via a two resistors of 1k or higher.

The programming mode used by Wisp648 is called HVP for High Voltage Programming (because a high voltage on the MCLR pin is used to enter programming mode). Some PIC chips support another, more limited programming mode: LVP for Low Voltage Programming. To use LVP it must be enabled in the configuration word in the chip. When so configured, one I/O pin (on most chips RB4 or RB5) is now PGM, for ProGraMming enable. When this pin is high immediately after a reset the chip enters the Low Voltage Programming mode. This LVP is not supported on all PICs, and it claims one I/O pin, which for the popular 18-pins chips is right in the middle of the only full 8-bits port. Hence LVP never became popular, and on newer PIC chips it is no longer available. This would be of little interest to you as Wisp648 user, except for one bug which is present in some PICs (but no-one seems to be able to tell in exactly which ones – it might even depend on the silicon revision): when the PGM pin is high during reset this can interfere with high voltage programming! This can happen even when LVP is not enabled in the image that is programmed into the chip. The solution is to make sure that the PGM pin is tied low. This can be done by adding a suitable resistor to the target circuit (maybe one is present already), or Wisp648 can take care of this via a pin in its programming interface. Note that the target circuit must allow the PGM pin to be drawn low – if you connect it to +5V with 10 Ohm resistor Wisp648 will not be able to pull it down. But 1k or more is OK. Note that this LVP problem is not specific to Wisp648: whenever you use an HVP programmer with a chip that has this problem, you must take care to pull the LVP pin of your target low.

Some PIC chips have a peculiar requirement for entering programming mode: Vpp (~ 13V) must be applied to their MCLR pin and only <u>after</u> that the power (5V) must be applied. This is called the

Vpp-before-Vdd sequence. Wisp648 can do this by generating the Vpp and then (very briefly) shorting the 5V supply. This is done by the big transistor (TIP122) on the Wisp648 board. Some target chips can not be programmed when this feature is active, and power supplies do not take this abuse well, so this feature can be disabled. To disable it you must remove the jumper on the pins J1. I suggest that you leave that jumper out by default, unless you are sure that all your power supply(s) can handle this. Your power supply must also limit the short-circuit current to a reasonable value. A few Ampere or less will be OK. All supplies based on 7805, 78L05 and similar chips have no problem with this. The Wisp648 has an on-board 7805-based power supply which can handle this shorting.

Power

The Wisp648 is designed for programming a target PIC which uses 5V power. The Wisp648 needs this 5V too. There are two possibilities for powering the Wisp648 and the target circuit:

- The target circuit has its own 5V supply. Do not connect a wall-wart to the Wisp648; the programmer will take its power from the target circuit.
- The target circuit does not (yet?) have its own 5V. Connect a wall-wart to the Wisp648; the programmer will supply 5V to the target circuit.

Do not mix the two situations, when both the target circuit and the Wisp648 supply 5V a small difference in the two voltages might damage one or both power supplies.

The Wisp648 has an on-board 7805-based power supply. To use this supply you must connect an external wall-wart that supplies 9 .. 18V DC. An AC wall-wart might work, but only for a target circuit that uses very little current (< 100 mA). The wall-wart connector on the Wisp648 is a 2.5mm centre-pin connector, which is commonly used for wall-warts. The centre pin is positive, which seems to be the most common polarity. Unfortunately, about 10 other connectors (and 1 other polarity) are also commonly used for wall-warts. If you prefer a more permanent and stable connection (those centre-pin connectors disconnect very easily) you can use the PCB screw connector block. The + mark on the PCB indicates where you must connect the positive wire.

The 7805 is a linear regulator, so the voltage difference between the input (from the wall-wart) and the output (5V under normal circumstances, but could be 0V when you accidentally short-circuit the power) will be dissipated as heat. The 7805 is short-circuit and thermally protected, so it should withstand all normal use. But it might get (very) hot when you use it to supply a large current, especially when combined with a high wall-wart voltage. The temperature rise can be calculated as

$$\Lambda_t = (V_{in} - 5.0V) * I * R_{th}$$

Λ_t = temperature rise in degrees Kelvin
V_{in} = Wall-Wart voltage
R_{th} = thermal resistance, for a bare 7805: 60 ^oK / W
I = total current in A

With a common 12V DC wall-wart and 100mA (= 0.1 A) total current the temperature rise will be 42 ^oK, which is reasonably safe. For a higher current and/or a higher input voltage you might want to attach a heatsink. A small heatsink might have an R_{th} of 20 ^oK / W, which rises the safe current by a factor of 3. For a heatsink with a still lower thermal resistance the internal thermal resistance of the 7805 must be taken into account (check the datasheet).

The Wisp648 contains a diode (D4, 1N5819) that will short the +5V power when it is supplied by the target circuit but accidentally connected in reverse. This diode can handle 1A, probably more for a short time. You of course will never apply power in reverse (I do), but just in case, it might be a good idea to use a power supply that is current-limited to 1A or less. In fact 100 mA will be

enough for most ‘beginners’ projects. Using a switching power unit from a PC is a very bad idea; such a PSU can deliver currents that are more appropriate for an arc welder than for an electronic project.

Serial pass-through

The Wisp648 can connect your serial port directly to the target chip. This can be very convenient: after programming your target chip you can communicate with it, using the same serial port, without re-plugging cables or other manual actions. This pass-through is done in the Wisp648 firmware, so it is limited to lower baudrates (up to 9600 baud seems to work OK). In-circuit programming requires two wires data between the programmer and the target chip, so it makes sense to re-use those two wires for serial communication. But the two PIC pins used for in-circuit programming are not the pins used by the PIC UART, so using these two pins forces you to use software (bit-banged) UART code in your target chip. The Wisp648 has two extra lines that are not used for programming the chip, but can be connected to the target chip UART pins. This requires you to make two extra connections, but now you can use the hardware UART in your target chip. (That is, if it has one. Not all PIC chips do.)

Note that the two extra lines (for connection to the PICs UART pins) are available only on the DB15 programming connector. The ICD2 and PICkit2 programmers do not have a pass-through function, so no pins are available on these connectors for such a function.

The common asynchronous serial communication (as used by the PC serial port) uses RS232 voltage levels and polarity. These PIC UART pins use TTL (0 .. 5V) levels and inverted polarity (inverted with respect to the RS232 line levels). A MAX232 or an equivalent chip is the standard way to interface between the TTL signals and the RS232 signals. But some dirt-cheap PIC circuits interface the PIC pins directly to the RS232 signals. This violates all design rules and specifications, but sometimes it works. Note that in this situation the TTL signals have the same polarity as the RS232 signals. Hence a PIC UART can not be used, because it uses inverted polarity.

The Wisp648 pass-through function interfaces between the signals on its RS232 connector and the target chip. Two choices are available:

- Use the programming lines (PGM and PGD) or use the extra lines (pin 7 and 8 of the DB15 connector)
- Use inverted polarity (the target PIC assumes that it interfaces to RS232 via a level/polarity converter – MAX232 or similar), or use true polarity (the target PIC assumes that it interfaces directly to RS232).

I recommend using inverted polarity: true polarity makes sense only for dirt-cheap circuits and its reliability is questionable. The choice between using the UART or bit-banging the serial communication on the PGC/PGD lines is up to you. The UART is easier to program, but the UART is not present on all PICs, and it requires two extra wires, which are available on the DB15 connector only. Using the PGC/PGD lines requires more work on the target chip, but needs no extra wires and allows the use of the ICD2 or PICkit2 compatible programming connectors.

When the target circuit itself has an RS232 interface and you want to ’take over’ this interface with the Wisp648 pass-through feature a resistor must be inserted to make it possible for the Wisp648 to ‘override’ the signal from the RS232 level converter. I suggest a value of 1 .. 10k. Figure 1 illustrates this trick. It also shows isolation resistors between the PGM, PGC and PGD lines and the rest of the target circuit, a 33k pull-up for the MCLR pin, and the mandatory 100nF decoupling capacitor for the power.

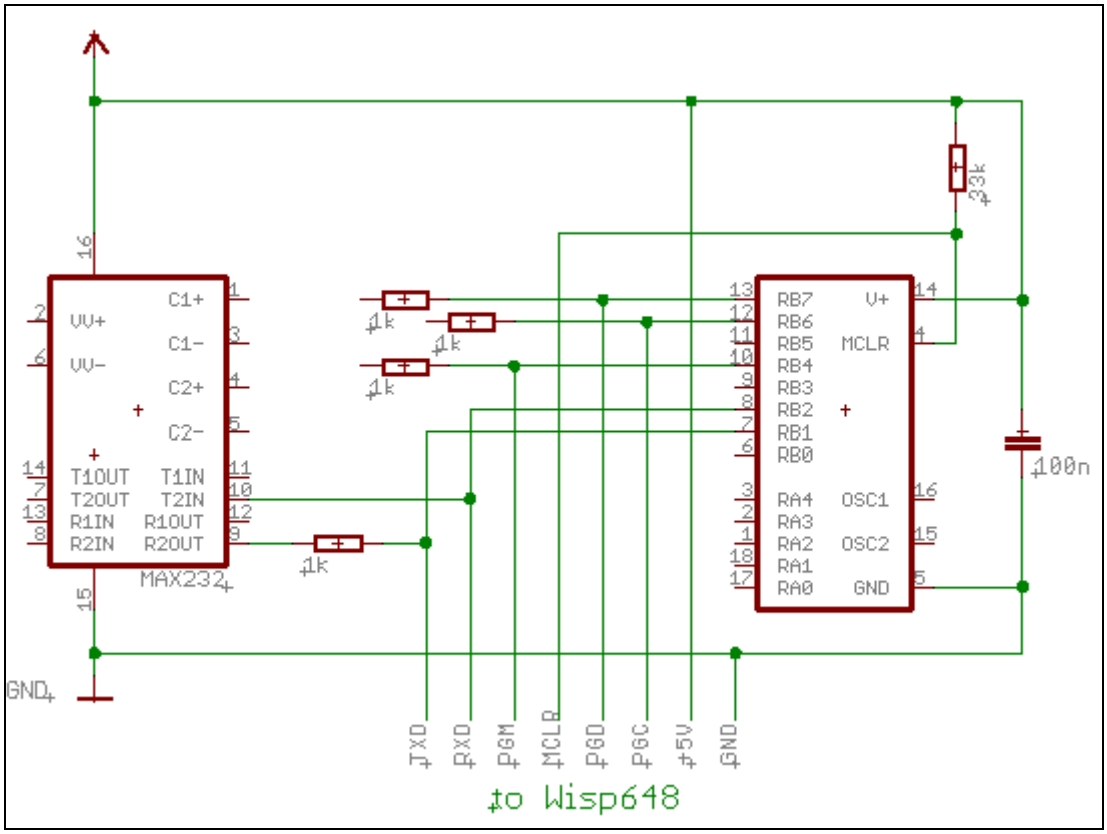


Figure 1 : combination of RS232 interface and serial pass-through

Firmware upgrading

The PIC 16F648A in the Wisp648 contains the Wisp648 firmware. This chip can not update its own FLASH, so to upgrade the firmware you will need either another 16F648A chip or a programmer that can program this chip.

If you have a second programmer that can program a 16F648A chip:

- Remove the 16F648A chip from the Wisp648
- Use that second programmer to program the new firmware into the chip
- Put the 16F648A chip back into the Wisp648

If you don't have a second programmer you will (temporarily) need a second 16F648A chip:

- Use the Wisp648 to program the new firmware into the second 16F648A chip
- Remove the 16F648A chip from the Wisp648
- Put the second 16F648A chip in the Wisp648
- Verify that the Wisp648 works and has the new firmware (the PC application will report the firmware version). You can now re-use the old 16F648A.

Circuit description

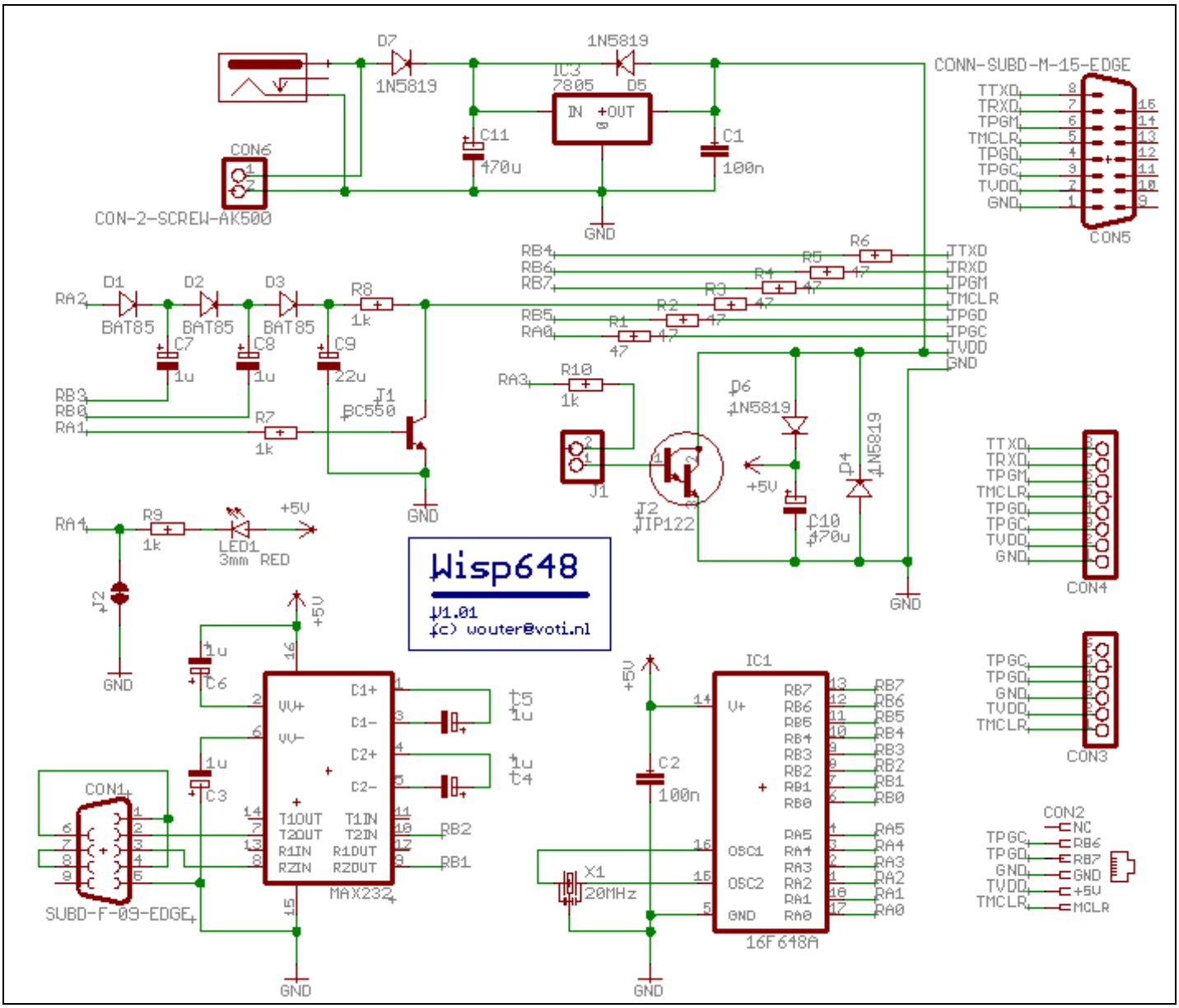


Figure 2 : Wisp648 circuit diagram

Figure 2 shows the Wisp648 circuit diagram. The MAX232 chip in the lower left corner handles the conversion between RS232 voltage levels and polarity, and the TTL levels and polarity used by the 16F648A PIC chip. The MAX232 connects to the RB1 and RB2 pins of the PIC, so the UART hardware of the PIC can be used. The LED (above the MAX232) is connected to pin RA4 of the PIC, so the PIC can switch the LED on and off. Pin RA4 is not very useful for other purposes because it is open-drain only (it can not pull itself high, only low). The LED can be activated permanently by closing solder jumper J2. This could be done when the PCB is (mis-) used as 7805 power supply only.

The 16F648A (lower right corner) chip has a 20 MHz resonator (cheap, robust, no capacitors needed) and the mandatory 100nF decoupling for its power. No MCLR pull-up is needed because the chip is configured for internal MCLR (pin RA5 is available as I/O pin, but not used here).

The three BAT85 diodes and the capacitors C7, C8 and C9 are a charge pump. When the PIC makes pin RA2 high, and then switches pins RB3 and RB0 high and low (180 degrees out of phase) it will create 3 x 5V (minus some losses) on capacitor C9. This is the source for the

programming-enable voltage Vpp. Transistor T1 is used to (briefly) pull the MCLR pin of the target chip to ground. When T1 switches off the MCLR pin will quickly rise to the ~ 13V on C9.

Transistor T2 can (under control of the PIC, and only when jumper J1 is placed) short the 5V to ground. While this is done the 16F648A PIC and the MAX232 run on the charge stored in capacitor C10.

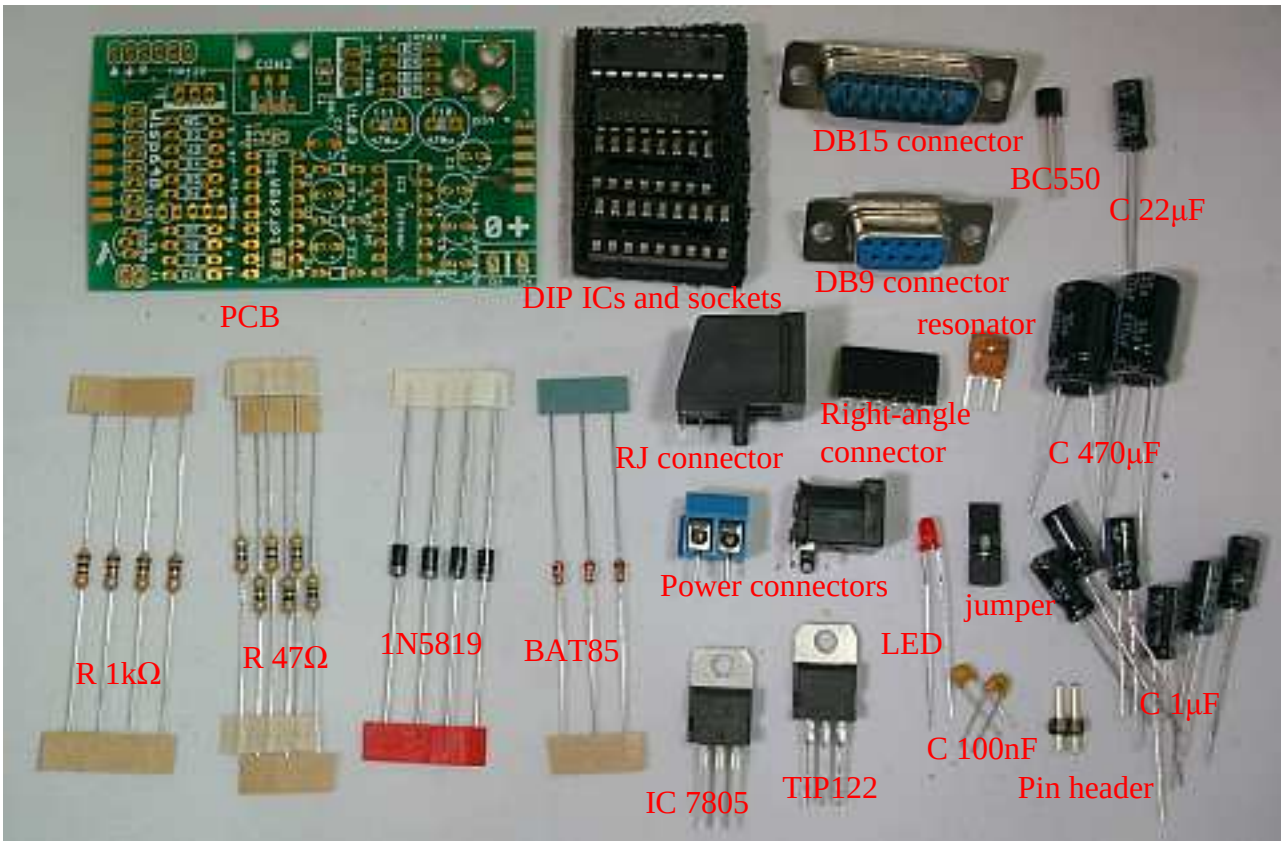
The circuit around the 7805 (top) is a standard 5V power supply. Two connectors are available to connect a wall-wart. Diode D7 prevents damage when the wall-wart is connected in reverse. Diode D5 protects the 7805 when the target circuit provides 5V power (a 7805 does not like to be reverse-powered).

The resistors R1 .. R6 provide a minimal level of protection against misuse by limiting the current that could result. They also damp ringing effects that could be caused by long wires.

Three connectors and a row of PCB pads are available for making the connection to the target circuit. The male DB15 connector is a cheap and robust connector, and provides the full Wisp648 interface. The RJ12 and pin-header connectors are compatible with the Microchip ICD2 and PICkit2 programmers / debuggers, but they lack the TPGM, TRDX and TTXD lines. The row of solder pads can be used to make a permanent connection to a target, for instance when the Wisp648 is installed permanently in a system.

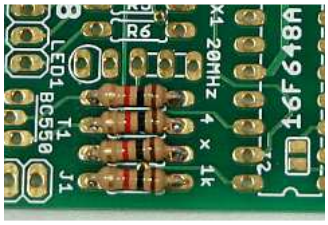
Kit construction

PCB

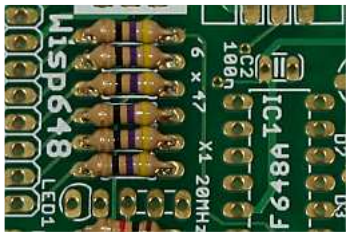


Place and solder the three BAT85 diodes. The (black) band on the diodes must be at the side of the white band in the PCB silkscreen (white print on the PCB). Trim the wires.

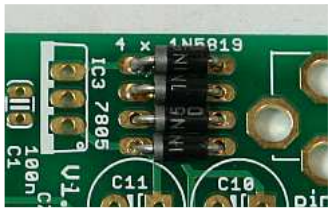
Place and solder the four 1kΩ (brown – black – red) resistors. Resistors are not polar, so it does not matter in which direction they are placed, but for aesthetic purposes it makes sense to have them all point in the same direction. Trim the wires.



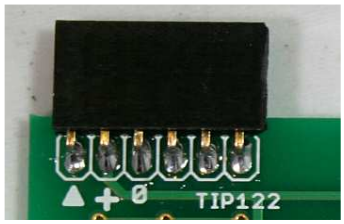
Place and solder the six 47 Ω (yellow - purple – black) resistors. Resistors are not polar, so it does not matter in which direction they are placed, but for aesthetic purposes it makes sense to have them all point in the same direction. Trim the wires.



Place and solder the four 1N5819 diodes. The white band on the diodes must be at the side of the white band in the PCB silkscreen (white print on the PCB). Trim the wires.

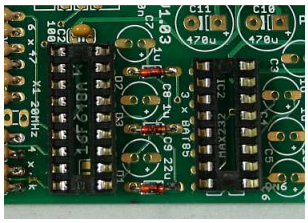


Place and solder the right-angle pin-header connector. For a good connection it is recommended to solder both top and bottom sides.

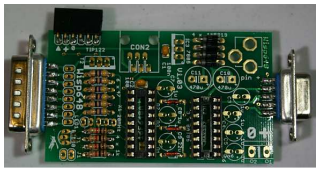


Place and solder the two 100 nF capacitors. Trim the wires.

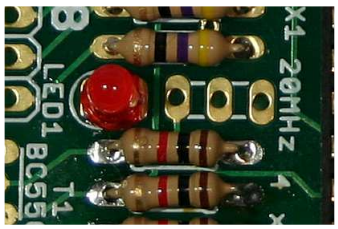
Place and solder the two PCB sockets. The sockets have a notch at one end, which should be above the notch in the silkscreen (white print).



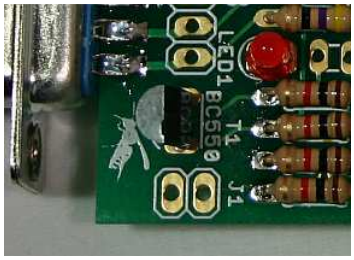
Place and solder the DB9 (female) and DB15 (male) connectors. Shifting the connectors over the PCB edge might require some force. There is only one way the two connectors can be places with each solder cup over a copper field on the PCB.



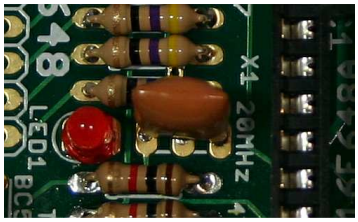
Place and solder the LED. The longer wire must be put in the hole in the middle of the silk ‘symbol’ for the LED. In other words: insert the longer wire in the hole nearest to the DB15 connector. Trim the wires.



Place and solder the BC550 transistor. The flat side of the transistor must be as show: towards the resistors.

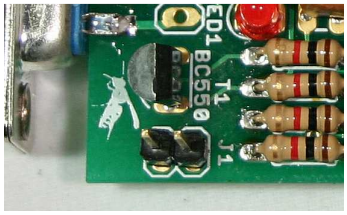


Place and solder the 20 MHz resonator. Trim the wires.

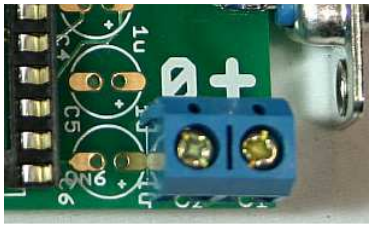


Place and solder the two-pin header.

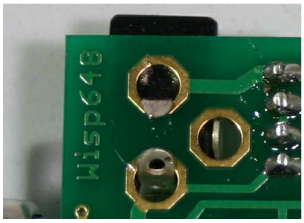
The latest version of the PCB uses a three-pin header.



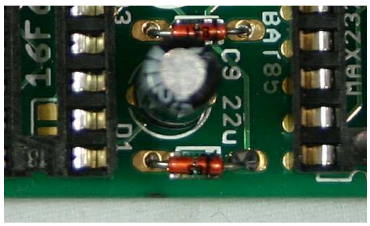
Place and solder the screw connector.



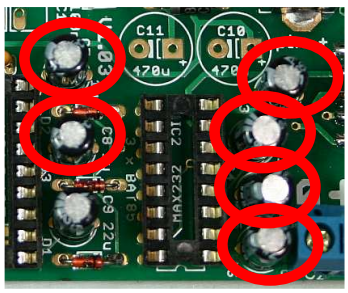
Place and solder the centre-pin power connector. I find it convenient to bend two of the pins before soldering.



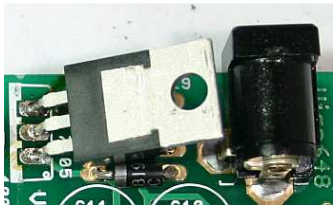
Place and solder the one 22 µF electrolytic capacitor. The longer wire must be inserted in the hole marked with a +. Or: all electrolytic capacitors must have their white bands towards the DB15 connector.



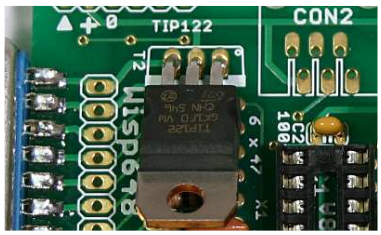
Place and solder the six 1 µF electrolytic capacitor. The longer wire must be inserted in the hole marked with a +. Or: all electrolytic capacitors must have their white bands towards the DB15 connector.



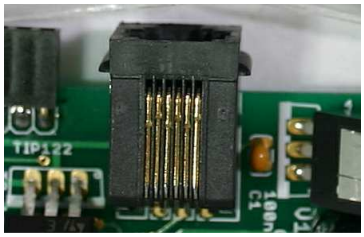
Place and solder the 7805 regulator IC. Be careful not to use the TIP122 transistor, which has the same TO220 shape! The metal tab of the regulator must be oriented away from the black power connector. The metal tab corresponds to the thick wide edge on the silkscreen drawing. I prefer to limit the component height of the Wisp648 programmer, so I bend the regulator towards the power connector. This requires a little slack in the wires, so it is easiest to do this before soldering. Trim the wires.



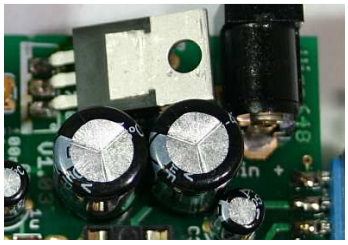
Place and solder the TIP122 transistor. Check again that it is indeed the transistor, not the 7805 regulator! The metal tab of the transistor must be oriented towards the resistors. Like the regulator, I prefer to bent the TIP122 slightly over the resistors.



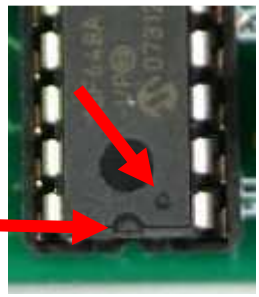
Place and solder the big black RJ11/12 connector.

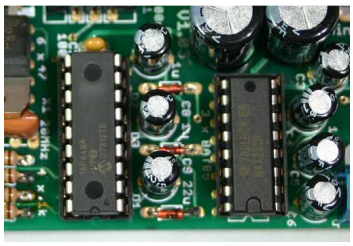


Place and solder the two 470 µF electrolytic capacitors. The recent versions of the PCB have one 470 µF capacitor and one of 2200 µF, check the PCB. The longer wire must be inserted in the hole with the square copper pad and marked with a +. Or: all electrolytic capacitors must have their white bands towards the DB15 connector. The capacitor nearest to the four diodes has a very tight fit, but with some pressure I could always push it in place,

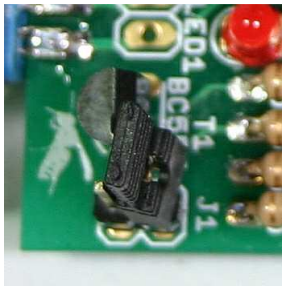


Place the two chips in their sockets. This will be easier if you first bent the two rows of legs a little bit towards each other. Note that each chip has either a notch at one small side, and/or a marking near a pin. This notch or marking must be nearest to the edge of the PCB.





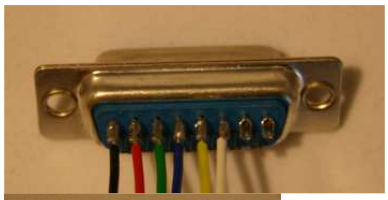
Place the pin jumper on one pin of the pin header. The jumper activates the TIP122 power-short option. As explained elsewhere in this document, it is advised to have this option disabled unless you are certain that it should be enabled.



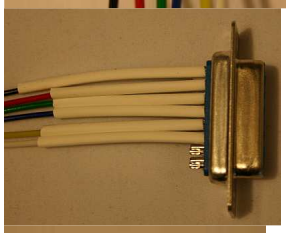
Recent versions of the PCB have a 3-pin header. The power-short option is activated by placing the jumper nearest to the resistors. The power-short is disabled by removing the jumper, or (at a request from a user who kept losing the jumper) by placing it in the position away from the resistors. On the most recent version of the PCB this position is marked ‘dis’, the active position is marked ‘ena’.

Cable

Strip a few millimeters of the six colored wires and solder them to the female DB015 connector. The solder cups are numbered from left to right: 1 … 8. You use only the top row, the bottom row has only 7 cups. From left (1) to right (6) the wire colors are black, red, green, blue, yellow and white.

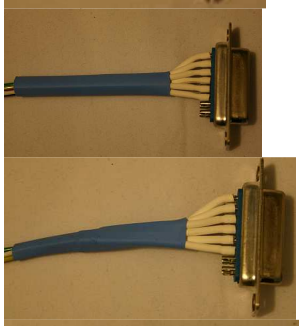


Shift the six 1.6 mm diameter heat-shrink tubes over the wires and the solder cups. The picture shows white tubes, but the color can vary.



Apply heat to shrink the heat-shrink. The fumes from the heat-shrink are not healthy, so do this in the open air. An electrical paint stripper is an ideal heat source, but a cigarette lighter or even a match could be used as alternative.

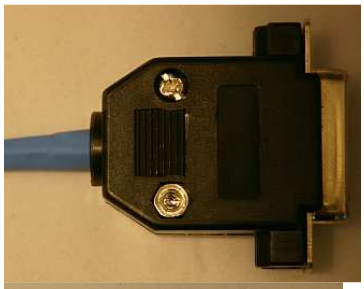
Shift the one 6.4 mm diameter heat-shrink tube over the six wires. Bend the wires to form a bundle centered behind the connector.



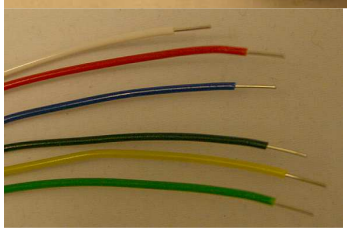
Shrink this tube too.

Fasten the strain relief at the appropriate place over the wire bundle.

Close the hood. I prefer not to use the two locking screws because the Wisp648 connector has no mating nuts, but feel free to use them if you want.



Strip a few millimeters from the other ends of the wires so they can be inserted in a solderless breadboard.



Troubleshooting

Does the LED blink?

When power is applied to the Wisp648 (either by the target circuit, via the in-circuit programming wires, or by a wall-wart), the red LED should blink three times. When the LED does not blink you should check

- the power supply, and
- the Wisp648 circuit around the 16F648A chip in the programmer.

Is your power OK?

If you use a wall-wart to power the Wisp648, it must supply 9..18 V DC. An AC wall-wart might work, but only for a target circuit that uses very little current (< 100 mA). If you use the target's 5V to power the Wisp648, make sure that it is really 5.0 V, certainly not 4.5Vor 5.5V. In either case you must put a 100nF capacitor over the power pins of the target PIC. A larger capacitor (22 uF or more, electrolytic is OK) must also be present, not more than 20 cm (sum of ground and 5V wire or trace lengths) away from the target PIC.

Connector pin assignments

DB9 female: RS232

Note: if you use a cable between your PC and Wisp648 it must be a straight cable (not crossed). Pins 2, 3 and 5 (ground, receive, transmit) need to be wired, the other wires can be wired or not. Only three pins (GND, RxD, TxD) are used by the Wisp648, the other pins are wired to provide the usual handshake feedback.

Wisp648 is designed to work with USB-to-serial converters. The performance (the time it takes to program a chip) might be somewhat slower than with a 'real' serial port, but the functionality should not be affected. It has been tested OK with converters based on Prolific and FTDI chips. If you find a converter that does not work with a Wisp648 I am very interested.

Pin	name	Wisp648 function
1	CD	not used, but connected to 4 and 6
2	RxD	Serial data: Wisp648-to-PC
3	TxD	Serial data: PC-to-Wisp648
4	DTR	not used, but connected to 1 and 6
5	GND	Ground
6	DSR	not used, but connected to 1 and 4
7	RTS	not used, but connected to 8
8	CTS	not used, but connected to 7
9	RTI	Not used

DB15 male: connection to target

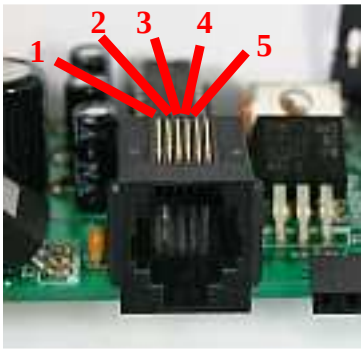
The colors correspond to the colors of the wires supplied with the kit. No wires are supplied for pins 7 and 8, so no colors are shown for these lines.

Pin	Wisp648 function
1	Ground
2	+5V
3	PGC
4	PGD
5	MCLR
6	PGM (LVP)
7	Asynch Wisp648 ➔ target
8	Asynch Wisp648 ➔ target
9..15	Not used

Black centre-pin connector and screw connector: wall-wart

The Wisp648 has two power connectors; you can use the one you prefer. The black 2.5 mm centre-pin connector is convenient and matches the connector often found on wall-warts. The screw connector is more reliable and should be used when a more permanent connection is to be made. The crew connector is clearly marked ‘0’ and ‘+’. The centre-pin of the black connector must be positive.

RJ11/12 connector: connection to target, ICD2 compatible



Pin	use
1	Not connected
2	PGC
3	PGD
4	Ground
5	+5V
6	Vpp / MCLR

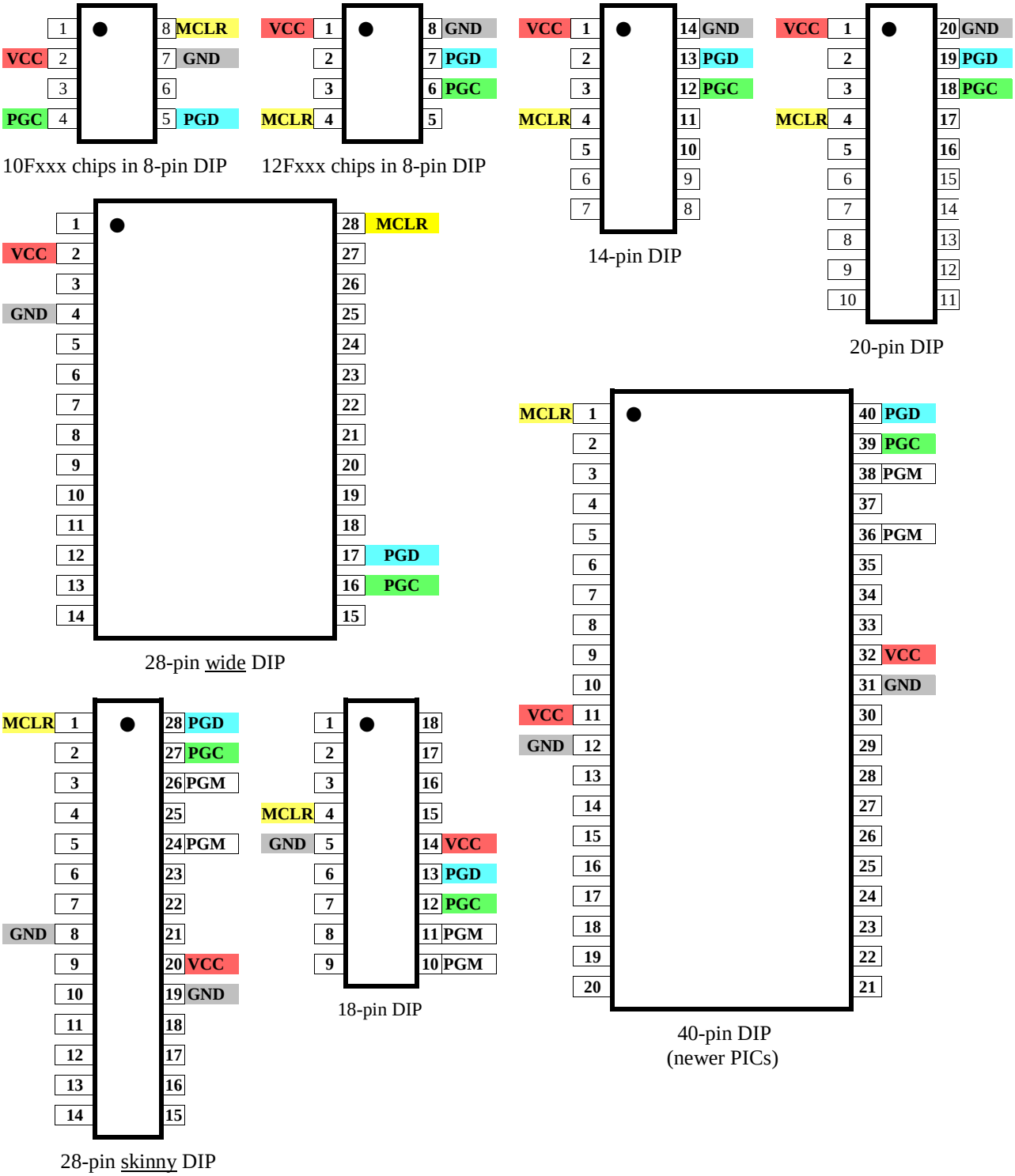
The RJ11/12 connector is compatible with the connector on a Microchip ICD2. Note that the Microchip ICD2 cable reverses the pins, so the pin assignment at the target side is the mirror of the pin assignment at the programmer side. The table shows the pin assignment at the programmer side.

6-pin header: connection to target, PICkit2 compatible

Pin	use
1	Vpp / MCLR
2	+5V
3	Ground
4	PGD
5	PGC
6	Not connected

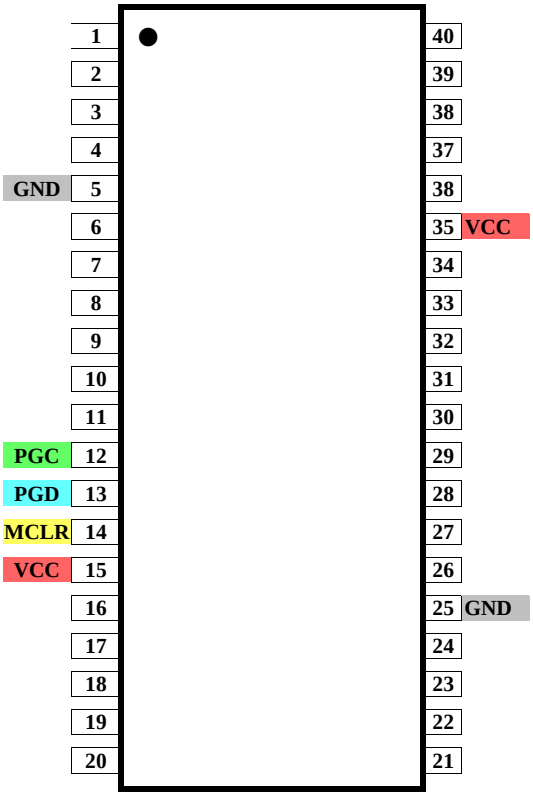
The Wisp648 has a PICkit2 compatible pin-header connector. You can use this connector connect to a target board that has a PICkit2-compatible pin-header connector. The table shows the use of the pins. Pin one is the one marked with the triangle.

Connecting a target chip



The pictures show how the colored wires of the Wisp648 should be connected to various target chips. For readability the black wire is shown here as gray, and the white wire (PGM) is shown in a box on a white background.. Note the difference in pinout between the 10F and 12F series, which both use an 8-pin package, and between the small and wide 28-pins packages. Also note that, as far as in-circuit programming is concerned, the 8 pin (12Fxxx only), 14 pin, and 20 pin DIP packages are sufficiently similar to program these chips in one (20 pin) socket. The PGM pin varies between chips, check the chips datasheet for the correct pin.

The FLASH versions of some older 40-pins chips, like the 16F59, use a different pinout, shown below.



(older PICs)

FAQ

Can Wisp648 be used with Linux

Yes, but you will have to use either the Python source version of the XWisp tool, or check whether one of the third-party PC programs will run on your particular Linux version.

I have a Wisp628

The Wisp628 is an older version of the Wisp648. It lacked a number of hardware features that are present on the Wisp648:

- No build-in 7805 power supply
- No build-in TIP122 circuit (could be added externally as ‘dongle’)
- No LED
- No ICD2 and PICkit2 compatible connectors
- MCLR (Vpp) was ‘forced’ to +5V (via diodes and 1k resistor)

The Wip628 used the pin 7 of the DB15 connector for the ‘dongle’ enable function. When the dingle circuit was connected, this pin could not also be used to pass serial data from the target to the PC. The Wisp648 firmware does drive this pin, to be compatible with Wisp628 hardware, but its on-board ‘dongle’ circuit is connected to a different pin of the 16F648A, so it can not be activated inadvertently by serial data.

Wisp628 board can be used with the Wisp648 firmware. It will behave as a Wisp648, except that the hardware features that are not present on a Wisp628 will of course not work. Note that you will

need a 16F648A chip. Recent Wisp628's will have this chip, older Wisp628's can have a 16F628 or 16F628A chip. The Wisp648 firmware does not fit in a 16F628 or 16F628A.

I don't have a serial port

Wisp648 is designed to work with USB-to-serial converters. The performance (the time it takes to program a chip) might be somewhat slower than with a 'real' serial port, but the functionality should not be affected. It has been tested OK with converters based on Prolific and FTDI chips. If you find a converter that does not work with a Wisp648 I am very interested.

I want to use a ZIF socket

Are you sure? In-circuit programming is so much easier. But assuming you know what you are doing: you can of course connect a ZIF socket to a Wisp648. Just take care that you connect a 100nF capacitor over the power pins. And for target chips that have multiple ground and/or power pins: connect them all. Note that the actual pins you must connect depend on the chip. Most chips with the same number of pins use the same pins, but you might want to check the datasheet to be sure.

Can I use a Wisp648 at 3.3V?

No, you can't. At 3.3V the charge pump that creates the ~ 13V Vpp does not function correctly, the MAX232 chip is out of spec, and the 16F648A can not run at 20 MHz. The Wisp648 is designed for 5V only, if you want to program at another voltage you will need another programmer.

I want to copy one PIC chip to another PIC chip

Wisp648 and its PC software can be used to read the content of a supported PIC, save this content into a .hex file, and write this file to another PIC chip, but only when the source PIC chip is not read-protected. Read-protection is a feature that the author of a program can enable if he chooses to do so. If he did, you are out of luck (and you probably have no business copying that chip). There are companies that will read a read-protected chip for you (for some fee), so don't rely on read-protection as an infallible protection mechanism for your own product. But it will prevent copying by anyone with only 'normal' tools.

Can I use Wisp648 for LVP programming?

No. But why would you want to? Wisp648 uses HVP programming, which can do everything that can be done with LVP programming, plus a bit more (like changing the LVP enable fuse bit). Note that after programming a chip, it makes no difference whether it was programmed using LVP or HVP, except that you can not use LVP to change the LVP-enable bit in the configuration fuses.

Should I enable the power short?

(The power short is enabled by placing the jumper on pins J1.) Some PICs always require this feature, some require it only with certain fuse settings (internal oscillator and internal MCLR), some can't be programmed when this feature is enabled, and some don't care. To complicate matters, some power supplies (especially expensive lab equipment) restart slowly after a short, so they won't work with this feature enabled. The most simple choice is to disable the power short, until you need it. You will probably only need it when you program a PIC with internal MCLR (MCLR pin configured as input pin).

Can I build my own Wisp648?

Yes you can. I explicitly allow anyone to build a Wisp648 for his own use. Of course I prefer you to buy a kit or build Wisp648 from my webshop, but for some people even the price of the kit might be too high. If that is the case for you: go ahead, build your own. The firmware and PC software can be downloaded from my site. But don't sell Wisp648 programmers, kits, PCBs or

programmed PICs for such programmers, and don't build them except for your own use. Some borderline cases:

- It is allowed to design and make a Wisp648 PCB, but it is not allowed to have it made for you by a PCB company, nor is it allowed to sell such a PCB, or to provide the PCB design to the general public.
- It is allowed to program a 16F648A with the Wisp648 firmware, create a PCB, or even build a complete Wisp648 programmer for a friend, provided that it is a single programmer (don't start a neighborhood Wisp648 building service), and you do not make any money doing so.

Note that there is no restriction on the *use* of a Wisp648 programmer. Why these restrictions at all? Because I must make a living, and I would like my PICmicro activities to contribute to that, so I can justify spending more time on those activities. That includes updating the Wisp648 design and PC software for it, so in the end you will benefit too.

Can you send me the Wisp648 circuit and PCB files?

Sorry, I won't. I sell a kit containing all Wisp648 parts, including a programmed 16F648A and a PCB. I want to encourage Do It Yourself activities, but you will have to use a breadboard, or design your own PCB. If you want the comfort of an existing PCB: buy the kit!

Why does my Wisp648 identify itself as Wisp628?

Most PC firmware for Wisp628 checks the programmer name, and refuses to work when it is anything but Wisp628. Wisp648 is compatible with such firmware, except that it would not work when it revealed itself as Wisp648. So I decided to let the Wisp648 identify itself as Wisp628, at least until all third-party PC software is modified to accept the identification "Wisp648".

Programming 18F fuses does not work

Some PIC fuses settings, in particular the enabling of the PLL in 18F chips, require a power-up to take effect. Just a reset is not sufficient. By default the Wisp648 will reset the target chip after programming, but it will not automatically cycle the power, so when you (for instance) enable the PLL in an 18F chip *for the first time* you will need to remove and re-apply the power.

Your PC software sucks. I want to make my own.

No problem. But you are not the first with this idea, so you might check on the internet for third-party PC software for the Wisp648, maybe you like it better than mine. Rob Hamerling is one source. His website is currently at http://www.robh.nl, otherwise google will be able to find him. But if you insist on making your own software, by all means do so. I would appreciate it if you make your software available on the internet and drop me an email so I can mention it on the Wisp648 page. The communication protocol used by the Wisp648 is described in other documents, check the Wisp648 page. If these documents are not clear feel free to ask me for an explanation. Or read the Wisp648 firmware, this too is available from the Wisp648 page.

PCB 1.05 bug

Version 1.05 of the Wisp648 PCB has a bug: the PGM line is permanently shorted to ground. The effect is that if you use the white wire to pull the PGM/LVP pin low during programming, this pin will permanently be pulled low, thus preventing normal use of this pin. If you don't use this pin you won't notice this problem (which is why I did not discover it – thanks Johan Van Hecke!)

Now what can be done about this?

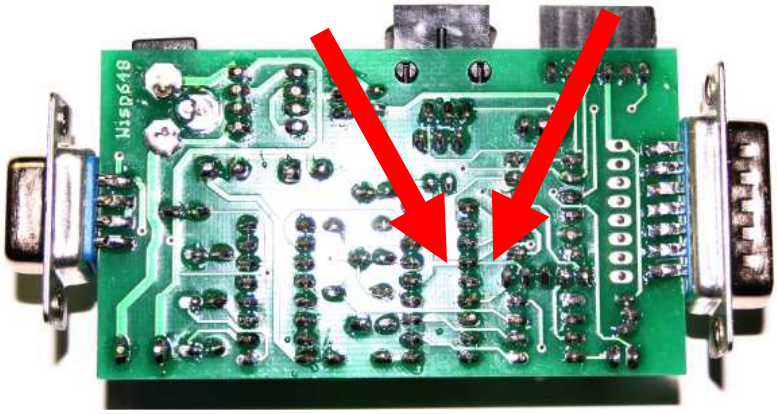
1. When you use a target chip that does not have the LVP feature you have no problem. Read no further (at least not for now).

2. Instead of using the white wire you could put a pull-down resistor between the PGM/LVP pin of your target chip and ground. A value of 10 .. 100 kΩ should work fine.

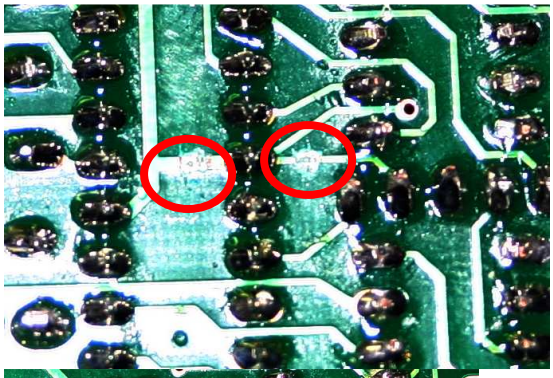
3. If your Wisp648 was bought ready-made you can contact me to have it replaced by a corrected version.

4. You could repair the PCB. This requires cutting two traces and soldering one wire. The pictures below show the steps. This is probably best done after the board is assembled.

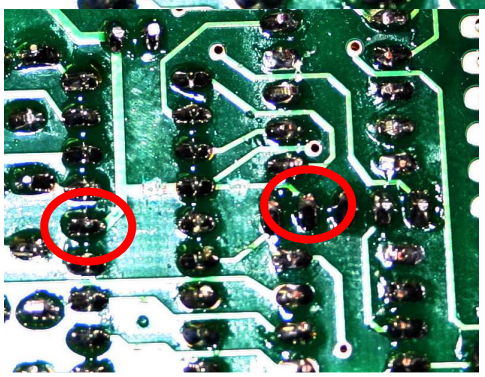
Cut the two PCB traces at the indicated places.



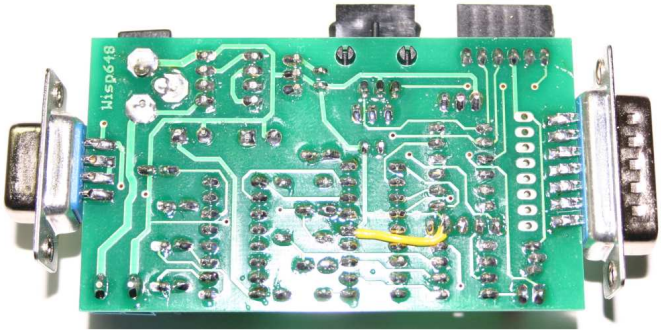
Now only the diagonal trace connects to the PCB pad.



Use a short piece of wire to connect the two pads that were previously connected.



This is what the result should look like.



Abbreviations and acronyms

CTS	Clear To Send : An RS232 signal that was used by the modem to indicate to the connected computer that it could send data. To avoid any waiting by legacy applications or drivers, it is often connected to the RTS line.
DSR	Data Set Ready : An RS232 signal that was used by the modem to signal that it was switched on and ready for use. To avoid any waiting by legacy applications or drivers, it is often connected to the DTR line.
DTR	Request To Send : An RS232 signal that was used by the computer to signal to the modem that it (the computer) was switched on and ready for use.
GND	GrouND : The neutral and return signal.
ICD2	In-Circuit Debugger 2 : A Microchip programming and debugging tool. It is a bit expensive, but supports a very wide range of PIC chips. When you switch target PIC chips very often it can be a bit annoying that it needs to download new firmware when you switch to a different target PIC type. A PICkit2 has both USB and serial connectors. It can use an external power supply (wall wart), it can draw power from the USB bus, or it can be powered from the target circuit. When powered from a wall wart the ICD2 can vary the target Vdd voltage.
LVP	Low Voltage Programming : Using its PGM pin to switch a target PIC chip to programming mode.
HVP	High Voltage Programming : Using a high voltage its MCLR pin to switch a target PIC chip to programming mode. This high voltage (see Vpp) is needed only to enable the programming mode, it does not supply any significant current.
MCLR	Master CLear and Reset : The reset pin available on most PIC chips. This pin also serves to enable the PIC programming mode, when the pin is (quickly) raised from 0V to the Vpp level. On most PIC types the Vpp level is ~ 13V.
PICkit2	A Microchip programming and debugging tool. Much cheaper than an ICD2, but USB only, a more limited range of supported target PIC types (especially when used as debugger), and it can not use a wall wart. But it is a lovely small gadget, and it might one day support standalone programming.
PCB	Printed Circuit Board
PGC	ProGram Clock : PGC and PGD are the programming interface pins of PIC microcontroller chips. When the target chip is in programming mode (see Vpp and PGM) the PGC line is used by the programmer as clock line to shift data bits into or out of the target chip. The protocol is documented in the “Programming Specifications” document for the particular chip.
PGD	ProGram Data : see PGC.
PGM	ProGraM : PIC pin that can be configured (in the fuses word(s)) to act as a logic-level programming enable pin (sampled once when the chip leaves reset). The advantage of using the PGM pin (LVP) over the use of MCLR as programming enable (HVP) is that no separate high voltage (see Vpp) is needed. Disadvantage is that the PGM pin is no longer available as normal I/O pin. Not all PICs have a PGM pin. On some PICs a floating or high PGM pin can hinder HVP programming. This can be solved by pulling the PGM pin low. This can be done by a simple pull-down resistor in the target circuit. Alternately the Wisp648 has a dedicated line (white wire) that is pulled low at the appropriate moments.
PIC	General Instruments once produced a chip called the PIC1650, described as a

Programmable Intelligent Computer. This chip is the mother of all PIC chips, functionally close to the current 16C54. General Instruments sold its product to Microchip. As far as I know Microchip has never used PIC as an abbreviation, just as PIC. Recently Microchip has started calling its PICs microcontrollers PICmicro MCU's.

RTS — Request To Send : An RS232 signal that was used by the computer to signal to the modem that it (the computer) wanted to send data.

RxD — Receive Data : RS232 data line. As seen from a DTE (Data Terminal Equipent) this is the line on which serial data is received.

TxD — Transmit Data : RS232 data line. As seen from a DTE (Data Terminal Equipent) this is the line on which serial data is received.

Vcc — Name for the positive power supply connection of an IC. The cc part might be derived from 'common collector', from the times when an IC could contain a lot of NPN transistors that all had their collectors connected to the positive supply rail.

Typical supply pin labeling				
bipolar transistor	FET			
V_{CC}	V_{DD}	V+	V_{S+}	Positive supply voltage
V_{EE}	V_{SS}	V−	V_{S-}	Negative supply voltage or ground

Vdd — Name for the negative or ground (0V) power supply connection of an IC. The dd part might (in analogy with Vcc) be derived from 'common drain'.

Vpp — Microchip's name for the voltage level on the MCLR pin that will enable the programming mode of a PIC chip.

Vss — Name for the negative or ground (0V) power supply connection of an IC. The dd part might (in analogy with Vcc) be derived from 'common source'.

WISP — Wouter's In-System (PIC) Programmer : My first PIC programmer design, with some unusual features: serial interface without a MAX232 or similar, but multiple WISPs could be connected to one serial port; it could vary the target Vdd voltage using a D/A converter that had to be calibrated; used pin RA4 to switch the ~ 13V Vpp directly (not possible with later PIC types). It used a 16c84 or 16F84(A). The design was published on the web, but I never sold this programmer.

"Tom Poes en de Wispen" is a comic story by the late Dutch author Maarten Toonder. In this story wisps are bees or wasps that cause the victim of their strings to predict the future. I liked this as analogy for a microcontroller programmer, so I choose a wasp as symbol for Wisp series of programmers.

Wisp628 — Successor to the WISP programmer. Used a 16F628(A) or 16F648A and a MAX232. No daisy-chaining, no variable Vdd. Sold as programmed PIC, as kit, and build. Needed an external circuit ('dongle') to program chips that require the Vpp-before-Vdd sequence.

Wisp648 — Successor to the Wisp628 programmer. Adds a build-in 'dongle-circuit', 7805 regulator, activity LED, and connectors compatible with ICD2 and Pickit2.

Document history

- 2010-05-30 : jumper text corrected; one elco value changed
- 2009-07-18 : the LED blinks three times, not two [Henk de Vries and many others]
- 2009-05-08 : the power-short option now has a 3-pin header instead of the 2-pin header

- 2009-01-13 : 10F pinout at p2 [Bert Oudshoorn]
- 2008-05-18 : Acronyms and abbreviations added; small textual improvements; PCB 1.05 bug documented;
- 2007-12-17 : color code for 47R corrected [Aat Koch, Peter Melin, Mikael Halvarsson], one elco is a tight fit [Aat Koch]
- 2007-12-15 : FAQ added: Wisp648 identifies itself as Wisp628 [Fris Kieftenbelt]; DB15 connector on the PCB must be male [Rikard Bosnjakovic]; 18F PLL fuse FAQ added; some textual refinements
- 2007-12-08 : kit instructions corrected: ”six 47 Ω” [Jan-Erik Soderholm]
- 2007-12-04 : 10Fxxx and 18-pin chips ICSP connections corrected [Lukas Lybaert]
- 2007-11-24 : PICkit2 connector added; layout improved; some minor changes
- 2007-11-20 : first version