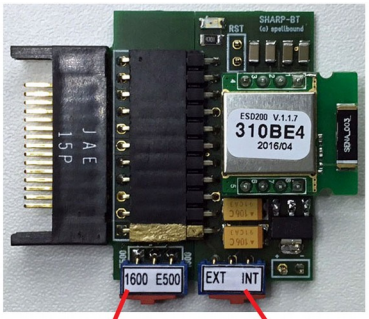
Generic Bluetooth Module for SHARP Pocket Computers



Hardware handshake switch: PC-1600 style vs 1360/E500 style

Power supply switch: Internal (Pocket) vs. External (5V-12V battery)

* Modular and generic
* Wireless data transmission
* Suitable for SHARP 15-pin and 11-pin SIO

e.g.
PC-1600
PC-1350/60
PC-E500
families

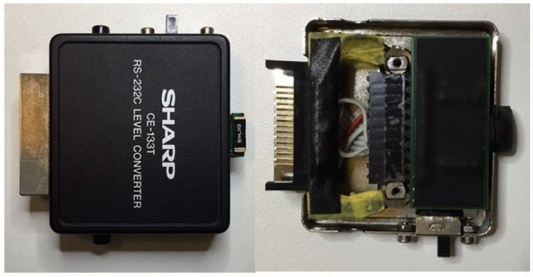
e.g.
PC-G850
PC-E220
families

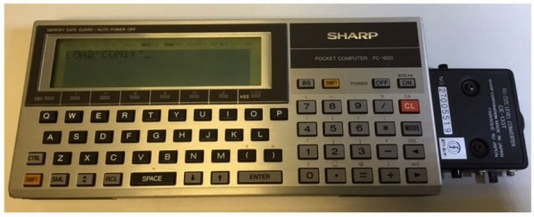
To replace all the different cables, for PC-connection with one reliable, full speed Bluetooth-module with bidirectional hardware handshake
No XON/XOFF needed

Capable of 8-bit binary program/data. transfer

UART-settings (baud-rate etc.) programmable by AT-commands directly from the Pocket Computer

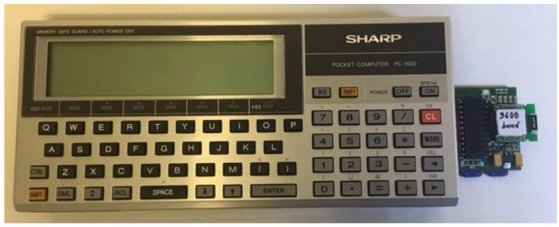
CE-133T or equivalent can be modded as a housing with 15-pins





Tested with

PC-1600



PC-1360 (with external power supply)

PC-E500S

PC-G850V

Introduction

This module is a wireless replacement for serial RS-232 cables and level converters, like the CE-160xL, CE-13xT, CE-T800/1 or custom FTDI-chip based USBtoUART cables.

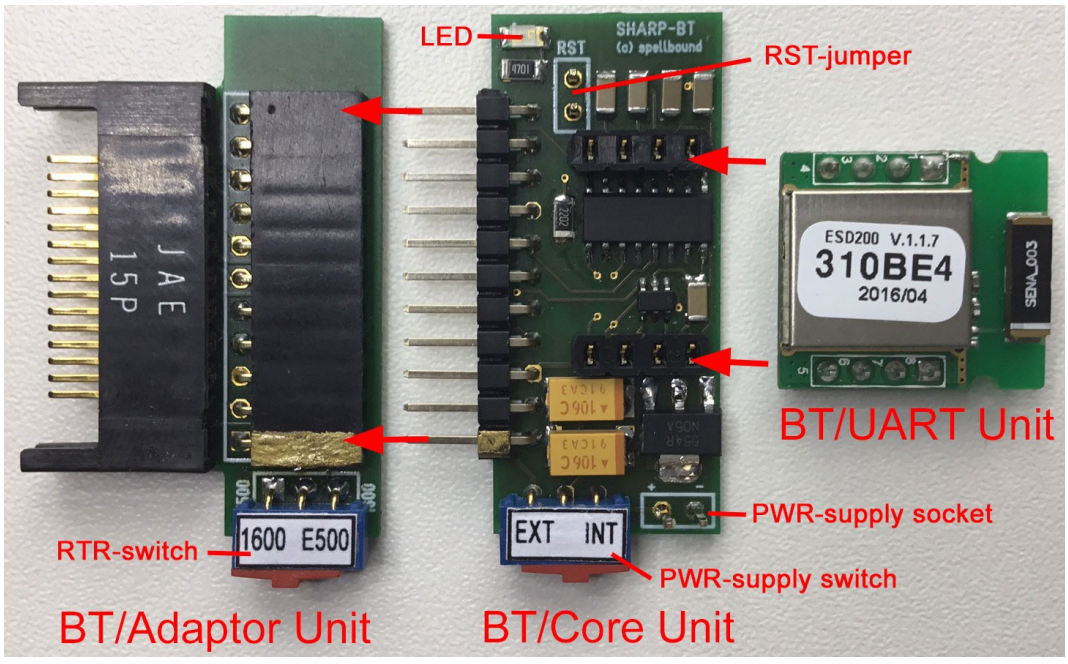
There are several advantages of the module over cable based solutions:

- Wireless communication to a PC/MAC (obvious)
- Reliable, full speed data transfer with bidirectional hardware handshake:
  The most common cables (e.g. CE-133T) do not support RS-232 hardware handshake at all, so the XON/XOFF protocol (software handshake) and typically a slower baud rate than the pockets maximum must be used instead.
  However, if you have a cable that supports bidirectional RTS/CTS hardware handshake for the PC-1600 this cannot support hardware handshake for the PC-E500(S) and PC-1350/60, and vice versa, because of incompatible signal interpretations (RTS vs. RTR) of these models - although they use the same physical 15-pin plug.
  This BT-module is able to compensate that via a switch.
- 8-bit binary data/program transfer capability
  If your pocket computer supports serial, binary data transfer, like the PC-1600 (e.g. by BSAVE/BLOAD) – the BT-module can handle that as well.
- One-fits-all:
  The BT-module is designed to support SHARP pockets with a 15-pin serial port as well as those with a 11-pin serial port.

Structure

The BT-module consists of three sub-modules or units:

- BT/UART Unit
  Namely the Sena Parani ESD200 Bluetooth to UART module.
- BT/Core Unit
  Custom level-converter and adaptor for the SHARP 11-pin serial interface
- BT/Adaptor Unit
  Custom adaptor from SHARP 11-pin to SHARP 15-pin serial interface including a switch for the hardware handshake style (i.e. RTS vs. RTR)



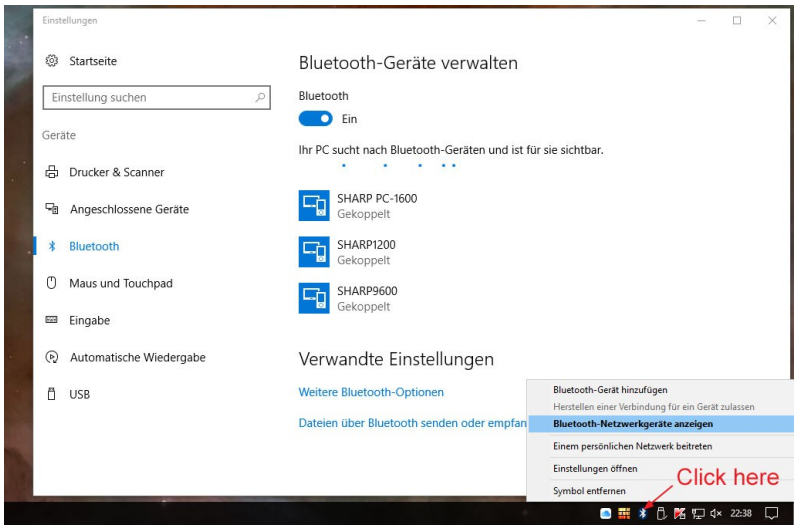
Hardware Functions

The picture above is also showing the switches, connectors, LED and jumper positions:

- RTR-switch
  Toggles the hardware handshake pinout for the 15-pin interface between PC-1600 style and PC-E500 style. Another SHARP pocket with PC-E500 style pinout (tested) is the PC-1360. In general these are pockets that provide an RTR (or RR) signal at pin-11 of the 15-pin serial interface. In contrast, PC-1600 style interfaces are those that only expose a RTS (or RS) signal at pin-4 and none at pin-11.
- PWR-supply switch
  Toggles the power supply source for the BT-module between internal (i.e. batteries of the pocket) vs. external (i.e. power supply connected to the PWR-supply socket).
- PWR-supply socket
  An external power supply of 5V to 12V can be connected here to unburden the batteries of the pocket. When the BT-module is used together with pockets that are driven by coin cells, like the PC-1350/60, the external power supply must be used.
  ATTENTION:
  Take care of correct polarity (+/-), otherwise the module may be destroyed.
- RST-jumper
  By setting a standard jumper here for about 3-4sec (or more) while the module is internally or externally powered the module performs a hard reset, which restores the factory default settings of the BT/UART (see doc of Sena Parani ESD200 and BT-Module Defaults, Hard Reset and Factory Defaults). The jumper has to be set after the module was powered on, otherwise no hard reset will be executed.
  This should only be done in "emergency cases", e.g. when you have set a baud rate higher than the max of your fastest pocket computer – so that the module does not "speak" to your pockets anymore ;-)
  The factory default UART setting is 9600baud, No parity, 1 stop-bit.
- LED
  The LED lights up while the DTR signal of the connected pocket computer is at HIGH level. This is the case, when the pocket opens its serial interface for communication (read or write). So the LED does not indicate actual data transfer but the state "ready for data transfer".

BT-Pairing

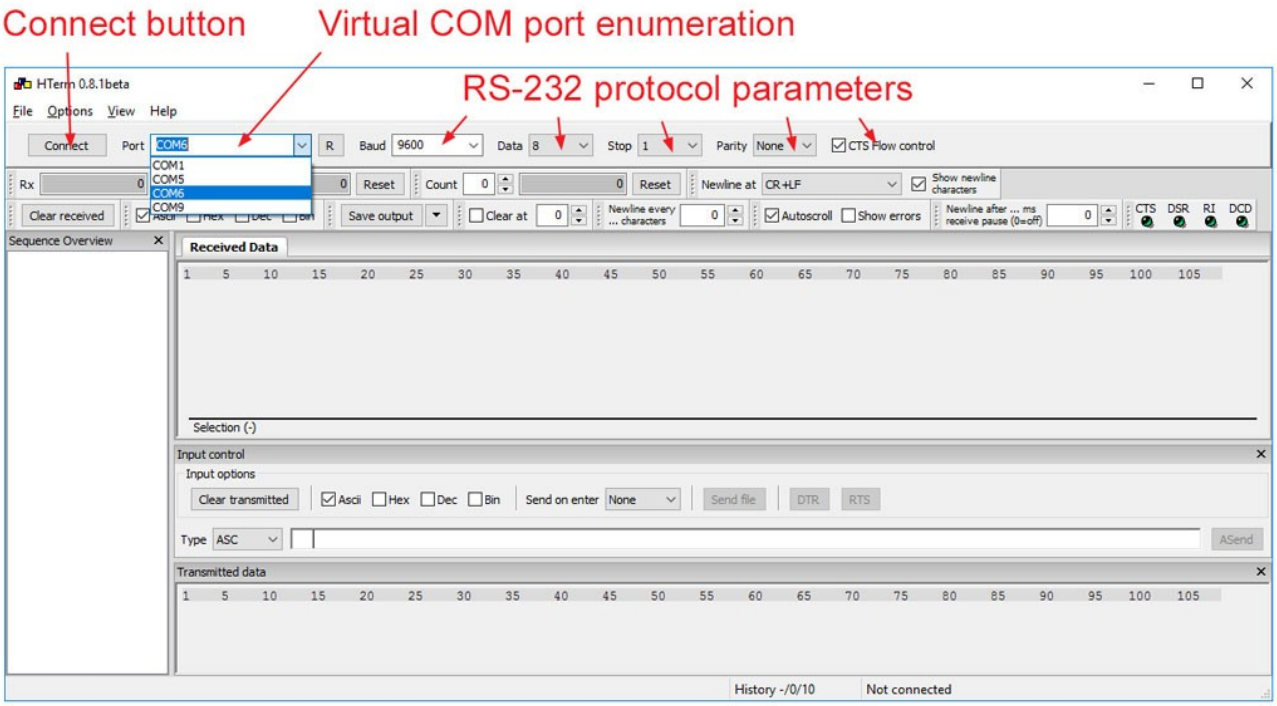
The Bluetooth pairing process is just like with any other Bluetooth device. When powered the BT-module is visible to your PC/Laptop/MAC (unless you change the modules operating mode) and the pairing can be initiated from those peers. The passcode is 1234, but can be changed. For Windows 10 the dialog is this:



BT-Connection

As a result of the BT-pairing process the host operating system (Windows, OSX, Linux) creates one ore more virtual COM-ports. Windows (10) creates two virtual COM ports per Bluetooth-device, which are enumerated by the Windows device manager and by the terminal programs as well.

The following screenshot shows hterm on Windows 10



One COM-port is for outgoing connections and one for incoming. Unfortunately you cannot easily decide upfront which is which – so you have to try connecting to the module by your terminal program and see which of the two is establishing a connection. A successful connection should at least result in activation of the DCD line. Terminal programs typically indicate that:

I recommend to deactivate the unused virtual COM-port in the Windows device manager to avoid further confusion.

ATTENTION

When the BT-module is powered but not connected it performs an active scan for BT-peers, which is the most power consuming mode of the module (about 50mA). Of course this is not critical, but you shouldn't leave the module powered but unconnected for a longer period of time to safe battery life.

If you plan a longer re-configuration session, you can put the module into main mode 0 (i.e. standby main mode) to save batteries - see BT-Module Settings.

Serial Settings

The fundamental principle of RS-232 communication in general is the alignment of the protocol parameters. They have to be identical for the pocket computer and the terminal program used at the peer. Since the BT-module can be thought of an 'intelligent' virtual RS-232 cable, its settings have to be identical too, in order to establish a functioning data transfer. The essential RS-232 protocol parameters and the default settings of the BT-module are:

1. Baud rate: 9600 – changeable in range of 1200 to 115200, each step doubles
2. Data bits: 8 – not changeable
3. Parity bit: None – changeable to Even, Odd
4. Stop bits: 1 – changeable to 2
5. RTS/CTS hardware handshake: On – must not be changed
6. DTR/DSR hardware handshake: Loopback – not changeable
   (i.e. DTR output of pocket is routed back to DSR and CD inputs
   => DTR/DSR handshake setting/requirement of the pocket is irrelevant)
7. XON/XOFF software handshake: not supported
   (i.e. the BT-module does not generate or interprete XON- and XOFF-chars but treats them as normal characters and just transfers them)

So the pocket computer and terminal prog as well ideally matches the BT-modules defaults. If this is not possible or desired, then select a setup that is as close as possible. E.g. the PC-1360's highest supported native baud rate is 1200, which means you have to lower the BT-modules baud rate to 1200 in order to use it with the PC-1360. But all other default settings of the module are/can be matched by that pocket.

The PC-1600, PC-E500 and PC-G850V however are fully compatible with the BT-modules default settings, so it comes "ready to run" for those pockets (surely others too, but not tested).

Pocket Computer Serial Settings

The concrete possibilities, commands and syntax for setting the RS-232 protocol parameters varies quite heavily between the different SHARP pocket computer models.

Here are some templates to match the BT-modules defaults:

PC-1600: // RTS/CTS hardware handshake setup via SNDSTAT and RCVSTAT

```
SETCOM "COM1:",9600,8,N,1,N,N
INIT "COM1:",512
SNDSTAT "COM1:",59:RCVSTAT "COM1:",61,0
```

PC-1360: // 1200 baud max, fixed RTR/CTS and DTR/DSR/CD hardware handshake

```
OPEN "1200,N,8,1,A,C,&1A"
CLOSE
```

PC-E500(S): // fixed RTR/CTS and DTR/DSR/CD hardware handshake

```
OPEN "9600,N,8,1,A,C,&H1A,N,N"
CLOSE
```

PC-G850(V/S): // config via menu, no DTR/DSR hardware handshake

TEXT->Sio->Format:

baud rate	= 9600
data bit	= 8
stop bit	= 1
flow	= RS/CS

Serial setup for other SHARP pockets should be similar to one of these.

REMARK

The end-of-line and end-of-file character-settings are not exactly part of the serial protocol settings. They are not interpreted by the BT-module, but by the terminal-app and the pocket computer. E.g. if you have a BASIC source file containing CR+LF eol codes that you want to LOAD, you have to adopt the settings (e.g. 'L' instead of 'C' in the OPEN-statement). But this is a general rule and has nothing to do with the BT-module.

Terminal Program Serial Settings

As stated above the serial settings of the terminal program have to be identical to that of the pocket and the BT-module. Recommended freely available terminal programs are:

Windows: hterm

MAC-OSX: CoolTerm

Once you have set the pocket and the terminal program to the same serial settings as the BT-module and have established a connection to the correct virtual COM-port on your Bluetooth peer computer, you can use the BT-module just like an RS-232 cable. You can perform the respective BASIC commands like SAVE and LOAD or even BSAVE or BLOAD for binary transfer, if your pocket computer supports that.

BT-Module Settings

This section is only relevant, if you want to (or have to) change the BT-modules defaults.

I recommend to change these settings only if mandatory – e.g. for adopting the baud rate to the pocket computers maximum.

The settings of the BT-module are the settings of the BT/UART sub-module (Sena Parani ESD200) that is hooked by the BT/Core sub-module (see Structure).

The settings of the ESD200 can be changed by so called AT-commands. These are special character strings beginning with "AT.." which are interpreted by the module when sent to it in standby mode (e.g. unconnected).

These settings are persistent (i.e. stored in an internal flash memory), so they "survive" the scope of a session.

In principle there are three different ways how the BT/UART sub-module can by accessed hardware-wise for a settings change:

1. Sena Parani JigBoard for ESD200
   Sena commercially offers a shield (JigBoard) for the ESD200 that has a DB9 port and can be connected via an DB9/USB Adaptor to a Windows PC. For convenience there is a Windows-based frontend application (called ParaniWin), that encapsulates the most importand (but not all) AT-commands behind a friendly UI.
2. The ESD200 doc includes a detailed circuit diagram for building a custom JigBoard. Rest as in 1.
3. The AT-commands can be entered and sent from your pocket computer directly to the BT-module without any shield/JigBoard.

The rest of this section is focused on 3. For the complete list and detailed description of all available AT-commands, see the ESD200 doc.

Technically, sending an AT-command from your pocket to the BT-module is nothing else but sending a character string over RS-232 (e.g. by OPEN, LPRINT). Reading the answer (e.g. 'OK' or 'ERROR') is nothing else but reading one ore more character strings (i.e. lines of the answer) from the RS-232 interface (e.g. by INPUT#)..

The only difference is that the module is accepting AT-commands only in standby mode and not in online mode (i.e. connected / ready for data transfer to a Bluetooth peer).

The ESD200 provides 4 different main modes:

Mode	Meaning	Remarks
0	Standby main mode, module accepts AT-commands, but cannot be connected	Factory default mode after reset
1	Try connect to the last connected device	Used for point-to-point communication with another ESD200 as slave
2	Wait for connection from last connected device	Used for point-to-point communication with another ESD200 as master
3	Allow other BT-devices to pair and connect. Normal operating mode.	The BT-module is in this mode when delivered

The main modes 1-3 provide two submodes each: Online and standby. In online-submode the module does not interprete or accept AT-commands but transfers data. In standby-submode it accepts AT-commands but does not send data via Bluetooth.

You can make an explicit transition from online-submode to standby-submode by sending a special escape character string (which is changeable and deactivatable). Default escape string is "+++". To turn back to online-submode use the AT-command `ATO`.

To change the main mode use the AT-command `AT+BTMODE,n` where n is the mode number from the table above.

As an example configuration session lets assume the module is in main mode 3, online (connected) and we want to change the modules baud rate to 1200. The module can respond to AT-commands with an "answer" (typically `'OK'`, `'ERROR'` or specific requested infos). The command response can be switched on by `ATS10=1` and off by `ATS10=0`. For an AT-command session the command response should be on, but for normal operating with your pocket PC it must be off. So here is a sequence, to do the job:

```
+++                          // escape to standby-submode => no repsonse
ATS10=1                      // command response on => response: OK
AT                           // 'ping' the module just for check => response: OK
AT+UARTCONFIG,1200,N,1       // set the desired UART parameters => response: OK
ATS10=0                      // command response off again => no response
ATO                          // return to online submode => no response
```

The UART parameter change becomes effective after the next power-on boot of the module.

The concrete BASIC-commands and syntax for RS-232 handling vary between the different SHARP pocket computers. Here are some templates (assuming serial setting done – see Pocket Computer Serial Settings):

PC-1600:

```
SETDEV"COM1:",KI,PO      // redirect input and printer output to COM1
LPRINT"AT"               // send a string to COM1
INPUT I$: PRINT I$       // read one line from COM1 and show
```

PC-1360, PC-E500:

```
OPEN                     // open the COM interface with last settings
LPRINT"AT"               // send a string to COM
INPUT#1,I$:PRINT I$      // read one line from COM and show
```

PC-G850(V/S):

```
OPEN"COM:"               // open the COM interface with Sio/Format settings
PRINT#1,"AT"             // send a string to COM
INPUT#1,I$:PRINT I$      // read one line from COM and show
```

Serial character string transfer for other SHARP pockets should be similar to one of these.

Here is an elegant and short dialog style program for entering AT-commands and displaying the respective results for the PC-1600:

```
10 CLS :WAIT 0
20 DIM C$(1)*80:DIM R$(1)*80: REM some cmd and resp are too long for std-vars
30 PRINT "COMMAND-INTERFACE"
40 SETCOM "COM1:",9600,8,N,1,N,N: REM must match active BT-module settings
50 INIT "COM1:",512
60 SNDSTAT "COM1:",59:RCVSTAT "COM1:",61,2: REM set handshake and rcv-timeout
70 OUTSTAT "COM1:"
80 SETDEV "COM1:",PO
90 C$(0)="":INPUT "SND:";C$(0)
100 IF LEN (C$(0))=0THEN END
110 LPRINT C$(0)
120 SETDEV "COM1:",KI
130 B=0:ON ERROR GOTO 190: REM install timout handler for COM1 receive
140 R$(0)="":INPUT R$(0): REM timeout, if no (more) respopnse line available
150 IF B=1GOTO 180: REM if there was a timeout, stop reading lines from COM1
160 IF R$(0)<>""THEN PRINT "RCV:";R$(0): REM print non-empty response line
170 GOTO 140: REM try read another line of the response
180 GOTO 80: REM ready to enter next command
190 B=1:IF ERN =143THEN RESUME NEXT: REM receive timeout on INPUT in line 140
200 PRINT "ERROR:";ERN :END
```

The baud rate in line 40 has to be adopted to the actual setting of the BT-module.
Be aware, that a response typically has multiple lines, some of which are empty. So the total response is consumed, when INPUT is blocking (resp. timed out, if your pocket supports that).

I leave it as an "exercise" to code equivalents for other SHARP pockets ;-)

BT-Module Defaults, Hard Reset and Factory Defaults

The BT-module comes pre-configured "ready to run" and these defaults are not equal to the factory defaults you get after a hard reset (see Hardware Functions) and `AT&F` command).

ATTENTION

After a hard reset the factory defaults are restored and the module is in main mode 0, not in main mode 3. So it can't be paired nor connected in that state!

After a hard reset or change of the BT-modules 'name' it may be necessary to perform a new pairing process from your host OS. Delete the old pairing (BT-device) first in that case.

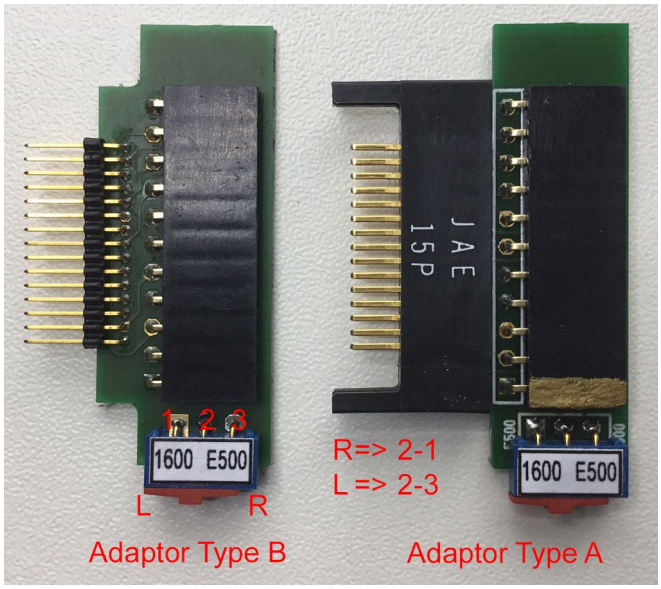
Here is a list of commands you can apply, in order to restore the delivery pre-configuration after a hard reset:

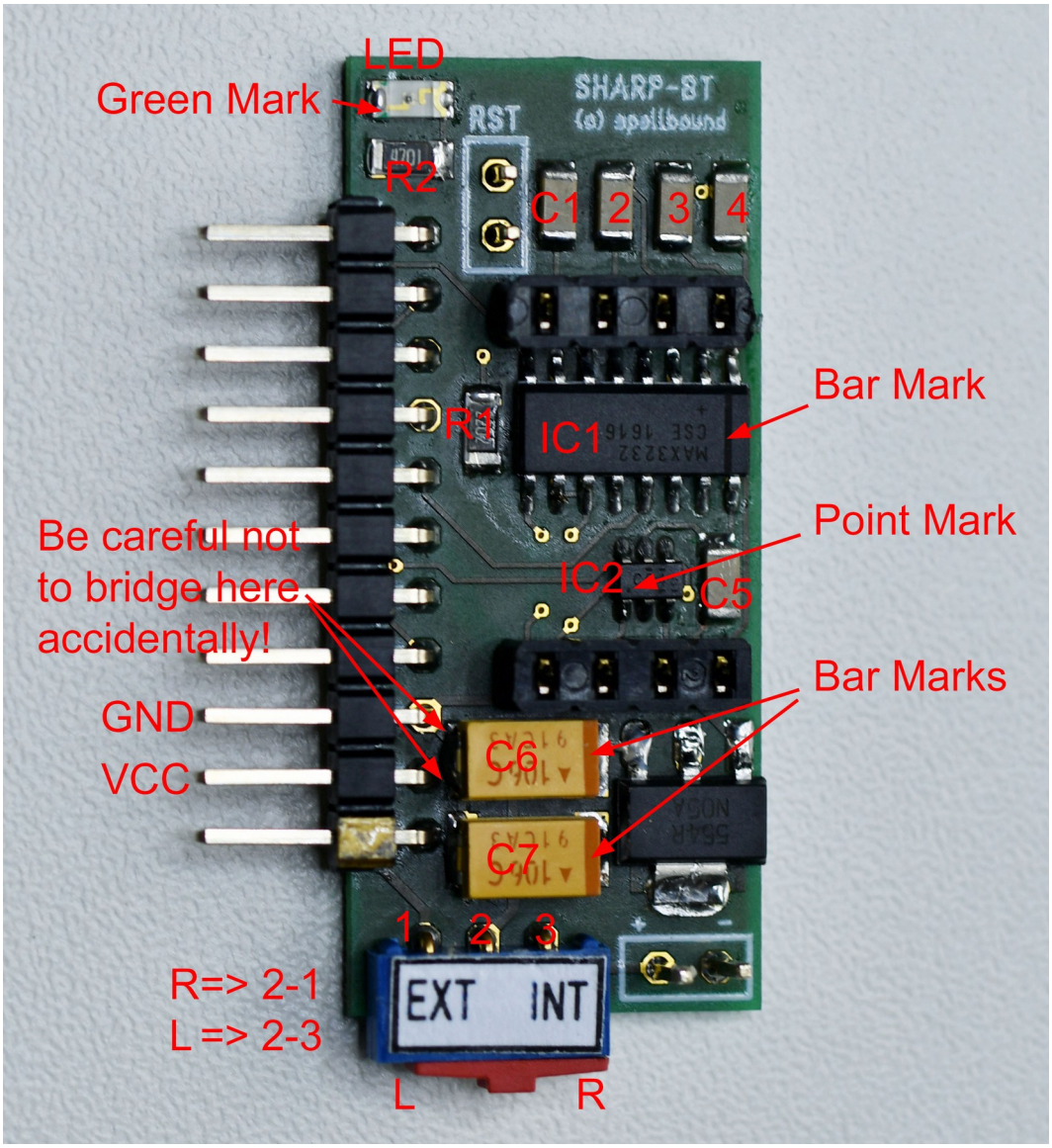
```
AT                          // 'ping' the module just for check (=> response: OK)
AT+BTNAME=SHARPBT1          // set a user-readable name for this Bluetooth device
AT+BTSEC,1,0                // set authentication=on (i.e. passcode), encryption=off
ATS12=1                     // IMPORTANT: set empty internal cache on disconnect
AT+BTMODE,3                 // IMPORTANT: change to main mode 3
ATS10=0                     // IMPORTANT: disable command response
```

(Some of) the changes become effective after the next power-on boot of the module.

Kit Assembly

The assembly of the kit is straight forward:





Orientation-critical parts are the ICs, LED and tantal electrolytic capacitors (C6,C7).

Recommended assembly order for BT/Core unit:

1. IC2
2. IC1
3. Small capacitors (C1-C5 = 100nF), resistors (R1=22kOhm, R2=4,7kOhm), LED
4. Large capacitors (C6, C7 = 10µF)
5. Voltage regulator
6. Plugs, pins, switch

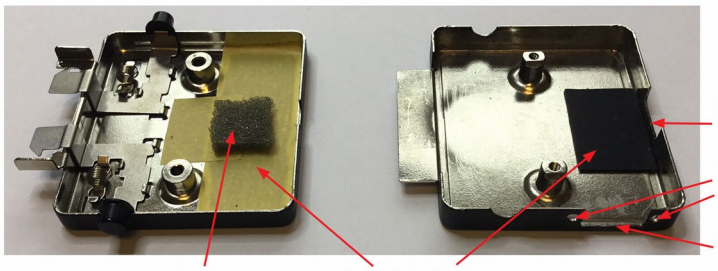
Be aware of the switching scheme of the provided toggle switches (see pics) in case you exchange them by different switches.

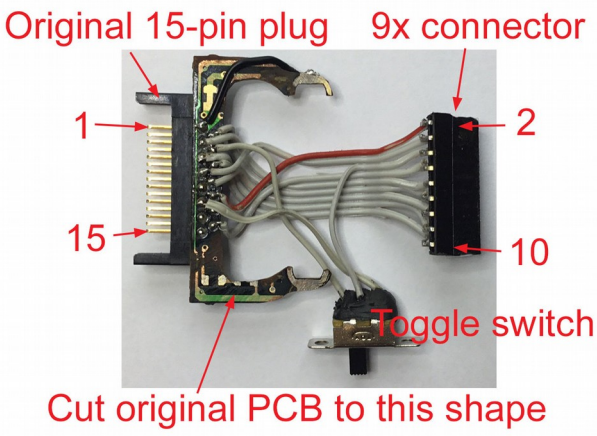
ATTENTION

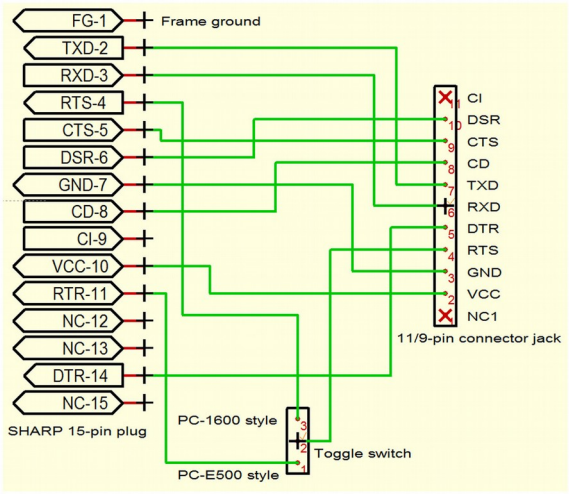
Before you make any operational tests with the module after assembly, please assure that there is no short circuit between VCC and GND (see pic).

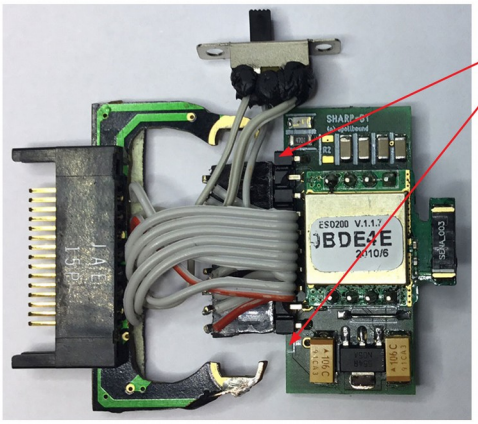
CE-133T Mod for Housing

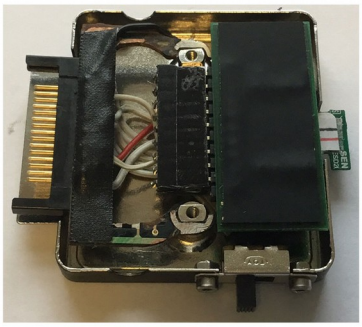
If you have an abandoned CE-133T or equivalent, you can modify it with a micro-tool to build a nice BT-module housing for 15-pin SHARP interfaces. To fit into the housing you have to remove the plastic and shorten the pins of the ESD200, so that the total assembly is as flat as possible.











Final assembly
and
usage

Symptoms, Causes and Solutions

Symptom	Possible Causes	Solutions
Protocol ERROR or invalid line number ERROR when LOADing a program	RS-232 protocol parameters of pocket, terminal-app or BT-module do not match	Check the settings (baud rate, parity, stop-bit, RTS/CTS-handshake) at pocket and terminal-app. If necessary change the BT-module settings. All three must match.
	The internal cache of the BT-module contains data (junk) from the last data transfer or AT-command responses from the module, that have not been read.	Disconnect from and power-off the module/pocket to empty the internal cache.Then reconnect again. If a hard reset was performed, assure that ATS12=1 is set.
	The program source file contains other end-of-line markers than you have set at the pocket (e.g. CR vs. CR+LF)	Change the end-of-line setting at the pocket computer or change the source file respectively.
The LOADing of a program does not terminate (stuck)	The hardware handshake is not set correctly	Check RTS/CTS settings in terminal-app and pocket. Check the handshake style switch of the BT-module (at the 15-pin adaptor).
	The source file does not contain the end-of-file character the pocket expects	Add the proper end-of-file character the pocket expects (e.g. 1A(hex)) or send it from the terminal app.

ATTENTION

Be aware that if there was an incomplete/interrupted data transfer from the host/terminal-app to the pocket, it is very likely that the internal cache of the BT-module contains some rest of that preceeding data stream (junk) that would be consumed by the next reading operation, leading to an error. So in that case, disconnect from and power-off the module/pocket to empty the internal cache.Then switch on and reconnect again.