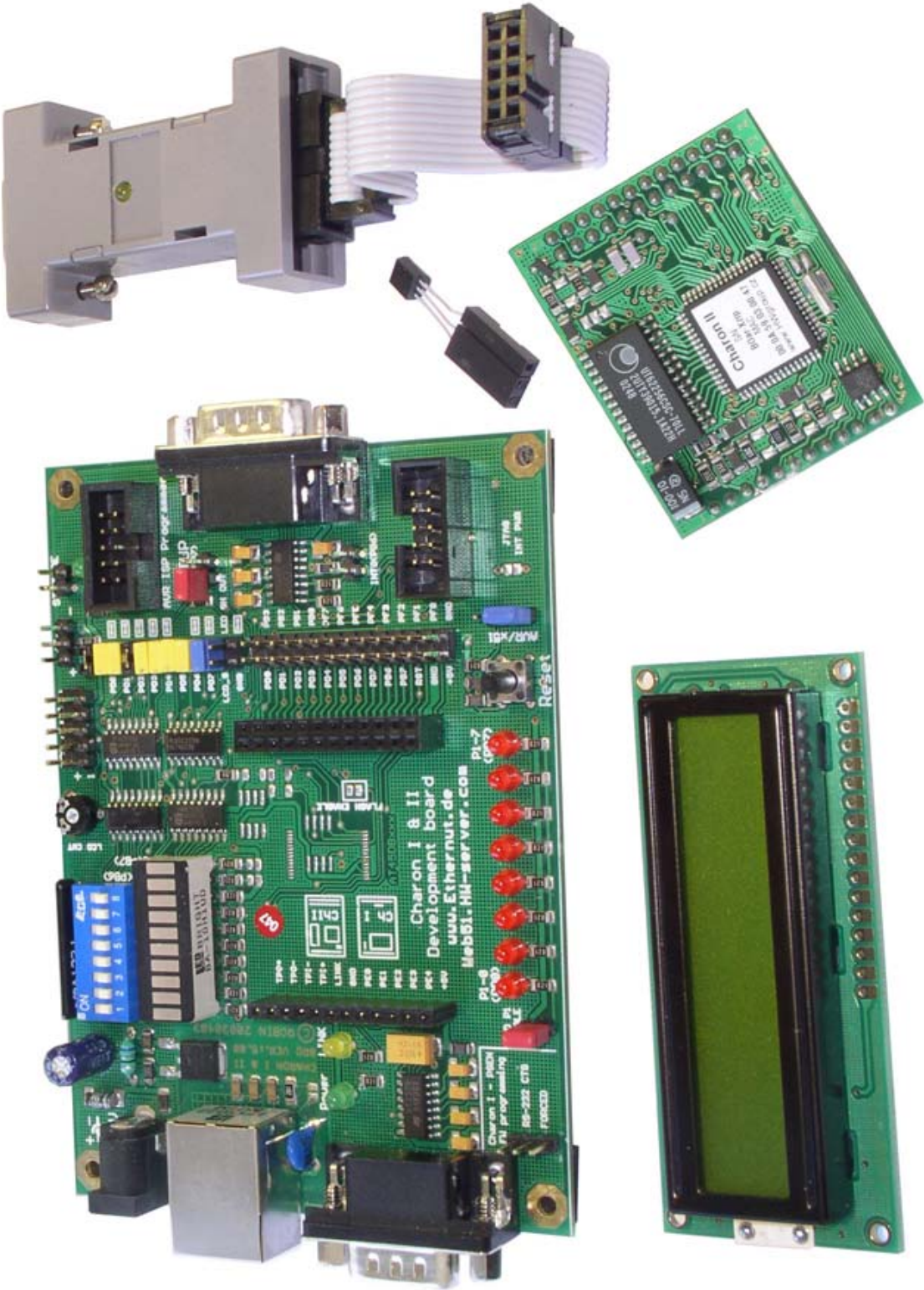
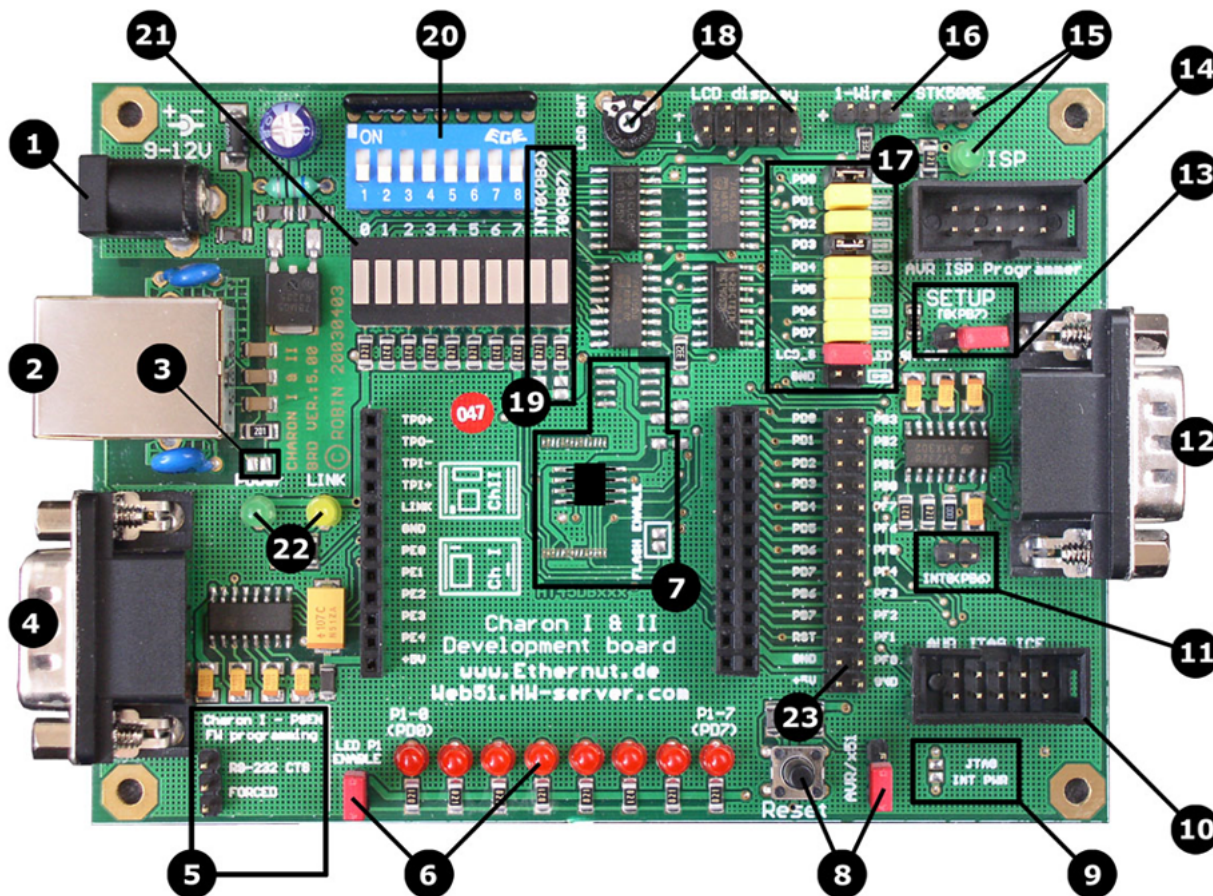


Charon II Development Kit

Getting Started guide





18 - LCD display connector

+5V	reg. sel.	Enable	D5	D7
① GND	Contrast	R/W	D4	D6

4 – RS-232 Serial port 0

2	RxD	Input	PE0
3	TxD	Output	PE1
5	GND	-	-
7	RTS	Output	PE3
8	CTS	Input	PE2

17 – Peripheral JUMPERS FIELD

PD0	1-Wire	Dallas 1-Wire pin
PD1	SH. LCD SET	RCK for LCD shift reg.
PD2	SH. IN LOAD	Load binary inputs to the input shift register (Serial port 1 input too)
PD3		Serial port 2 data input
PD4	SHIFT INPUT	Serial data input from the input shift register
PD5	SH. LED SET	RCK for LED shift reg.
PD6	SHIFT OUT	Serial data output
PD7	SHIFT CLK	Serial data clock
GND	Periph. Enable	Enable shifted periph.
SH. LCD SOUT	SH. LED SOUT	Serial data outputs from the LCD and LED shift registers. Don't short!

14 – AVR ISP connector

+5V	GND	GND	GNS	GND
① MOSI	ISP prog.	RST	SCK	MISO

12 – RS-232 Serial port 1

2	RxD	I	PD2
3	TxD	O	PD3
5	GND	-	-
7	RTS	O	PB6
8	CTS	I	PB7

10 – AVR JTAG ICE

TDI	Int. PWR	TMS	TD0	TCK	①
GND	n.c.	RST	+5V	GND	

Charon II module - pinout

1	TPO+	Ethernet output	INT0/SCL	PD0	13	25	PB3	SPI MISO
2	TPO-	Ethernet output	INT1/SDA	PD1	14	26	PB2	SPI MOSI
3	TPI-	Ethernet input	INT2/RxD1 serial 1	PD2	15	27	PB1	SPI SCK
4	TPI+	Ethernet input	INT3/TxD1 serial 1	PD3	16	28	PB0	SPI /SS
5	LINK	Link LED	IC1	PD4	17	29	PF7	ADC7/ TDI
6	GND	Ground	XCK1	PD5	18	30	PF6	ADC6/ TD0
7	PE0	RxD0 serial 0	T1	PD6	19	31	PF5	ADC5/ TMS
8	PE1	TxD0 serial 0	T2	PD7	20	32	PF4	ADC4/ TCK
9	PE2	AIN+/XCKO	OC1B	PB6	21	33	PF3	ADC3
10	PE3	AIN-/OC3A	OC1C	PB7	22	34	PF2	ADC2
11	PE4	INT4/OC3B	CPU reset pin	RST	23	35	PF1	ADC1
12	Vcc	+5V / cca 80 mA	Ground	GND	24	36	PF0	ADC0

Peripherals description

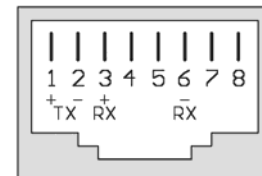
1 – Power source

DC 8 -15V power supply input, polarity protected. Use a standard 9V DC adapter with a coaxial power jack. Polarity + = center pin.



2 – Ethernet RJ45 connector

The RJ45 connector with integrated signal transformer compatible with standard TP cable 10 Mbit. You can use shielded TP cable, signal ground separated.



3 – Termination jumper

If you are using a **Charon I** (*version 6.2 or lower*), please solder this jumper. It connects the ethernet termination resistor, because older versions of the Charon I module does not have the integrated 100Ω termination resistor for the signal transformer input termination.

4 – RS-232 (Serial port 0)

First asynchronous serial port, only Rx/D, Tx/D and Handshake (RTS, CTS) pins available. Port is wired like standard PC serial port = use LapLink crossed cable for connection with the PC.

PC RS232 Port Cannon 9 – Male		
Pin		Signal
1	<-	CD
2	<-	RxD
3	->	TxD
4	->	DTR
5	--	GND
6	<-	DSR
7	->	RTS
8	<-	CTS
9	<-	RI

5 – Charon I PSEN FirmWare Programming

The jumper for Charon I programming only, for normal operation or when Charon II is installed remove this jumper.

The T89C51RD2 or RE2 chip used on the Charon I module is serial programmed by holding PSEN to ground during reset. You can control this pin from the serial port by using CTS and the reset button, if the flashing software supports it (Atmel Flip doesn't). Use the "**RS-232 CTS**" position for this case, The "**FORCED**" position selects the programming mode manually.

Charon II Bootloader extension: If you are using Charon II with Bootloader you can activate it using this pin by CTS or in the "**FORCED**" position.

6 – LED P1 enable and LEDs

8x LED indicate inverted state on the PD0 (P1) port, [Logic 0 = LED "ON", Logic 1 = LED "OFF"].

The "**LED P1 ENABLE**" jumper is used to enable the LEDs. Remove this jumper to use SHIFT peripherals.

SHIFT peripherals

Some peripherals are connected to Charon modules over PD port (or P1 for Charon I) using serial SHIFT registers. It is:

- 12 - RS-232 (**Serial port 1**)
- 16 - **1-Wire bus** connector
- 18 - **LCD display**
- 20 - **Binary input switch**
- 21 - **Parallel out LEDs**

Note : The "Serial port 1" and "1-Wire bus" use some pins of the port, so we call them *SHIFT Peripherals* too, although they are not connected over shift registers.

7 – SPI FLASH area

The ATmega128 CPU supports SPI interface, it can interface to SPI FLASH. This SPI FLASH is not originally mounted on the PCB and is not included in the standard Development Kit.

The following SPI FLASH packages are supported:

- **SOIC 8 package** - from **AT45DB011B** (1Mbit) to **AT45DB041B** (4Mbit)
- **TSOP 28 package** - from **AT45DB021B** (2Mbit) to **AT45DB161B** (16Mbit)

These additional components are required to support the SPI FLASH (not included):

- 3V linear regulator **L2951 SO8***
- Linear regulator capacitors **C11*** and **C12 * 100nF/1206**
- Resistor **R15*** for FLASH enable pin and short soldering jumper **SJ1***

* Component sources are displayed at the end of this datasheet.

8 – Reset switch & polarity jumper

Reset switch connecting RST pin from the Charon module and the JTAG and ISP connectors to the voltage polarity defined by JMP3 – reset polarity jumper.

- When using **Charon I**, jumper **position x51** – (2-3 on the board image, reset = Logical 1)
- When using **Charon II**, jumper **position AVR** - (1-2 from the scheme, reset = Logical 0).

9 – JTAG internal power solder jumper

Shorting this jumper will connect internal power of the JTAG programmer (pin 7 on the JTAG connector) with Vcc +5V on the Development Board.

10 – AVR JTAG ICE connector

When using the original AVR JTAG ICE device you can trace or download your program to the ATmega128 CPU.

TCK	1	2	GND
TD0	3	4	+5V
TMS	5	6	RST
Int PWR	7	8	n.c.
TDI	9	10	GND

Note : JTAG interface can be useful for reprogramming broken FUSES of the CPU.

11 – PB6 / OC1B jumper

Jumper for your applications, be aware that :

- The **Charon II** combines these functions on the PB6 pin:
 - RTS handshake output on the serial port 1.
 - DRDY/BUSY SPI flash output from the TSOP28 package.
- The **Charon I** module uses PB6 (INT0 for Charon I) for internal interrupts from the RTL Ethernet driver in all *Web51-C applications*.

12 – RS-232 (Serial port 1)

Second asynchronous serial port, only RxD, TxD and Handshake (RTS, CTS) pins are supported. Port is wired like standard PC serial port = use LapLink crossed cable for connection with the PC.

When using “Serial port 1”, be aware off the following combined functions on the Development Board:

- **RxD data input** – The PD2 pin is simultaneously used as “SHIFT REGISTER input data load” on the SW1 input shift register.
This means that with every incoming serial bit is a newly loaded parallel input to SHIFT register.
- **CTS serial handshake input** – The pin PB7 of the Charon II module is simultaneously used as 13 - SETUP jumper (JMP1). All our applications start in the serial port 0 SETUP mode by holding PB7 to ground during reset

***Note:** Don't use the SETUP mode together with the “Serial port 1” hardware CTS/RTS handshake, because it can occur that the CTS pin from the external application is holding PB7 to ground and if your application is reset, it will start in the SETUP mode, although SETUP jumper JMP1 is open.*
- **RTS serial handshake output** - PB6 is used as second JMP2 jumper and DRDY/BUSY SPI flash output from the TSOP28 package.

Result: Second asynchronous “Serial port 1” is supported only on the Charon II. If your application uses SPI flash in TSOP28 package or the 13 - SETUP jumper, we strictly recommend using “Serial port 1” without using HW handshake signals (CTS/RTS) in production applications.

13 – PB7 / SETUP mode JMP1 jumper

The SETUP mode jumper is used for starting SETUP mode after RESET. In the SETUP mode you can set module IP address and another parameters with using standard serial terminal **9600, 8N1, Flow control none** on the “4 - Serial port 0”.

Note: This pin is collected with CTS handshake input from the serial port 1.

14 – AVR ISP (In System Programming) Port

This port allows serial programming of the ATmega CPU (on the module) Flash ROM and EEPROM without physical removal of the microcontroller from the module's board. A multiplexer (IC2 - 4053) switches the microcontroller pins (PB0, PB1, PE0) when the ISP programming mode is selected.

MOSI	1	2	+5V
ISP Prog.	3	4	GND
RST	5	6	GND
SCK	7	8	GND
MISO	9	10	GND

15 – ISP LED & STK500 programming jumper

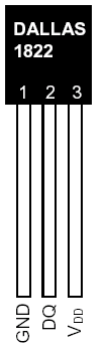
The ISP programming is available and the ISP pins are switched, when the STK 500 jumper is short. Using the original HW STK500 programmer, you don't have to manipulate with this jumper.

The ISP LED1 on the board indicates programming activity.

16 – 1-Wire bus connector

The 1-Wire interface is typically used with the DS18S22 or DS18B20 temperature sensors. It's connected to GND, +5V and DATA. DATA is connected through J1 in the jumpers field to the PD0. GND pin is marked on the PCB.

Pin	Function
1	+5V
2	DATA (PD0)
3	GND



17- Peripheral JUMPERS FIELD

Jumpers on this jumper's field connect various serial peripherals to the PD port. If you want to extend on the Development Board shift registers output (LED and LCD output 74595 shift register) you can use last 2 pins "SHIFT LCD SOUT" and "SHIFT LED SOUT".

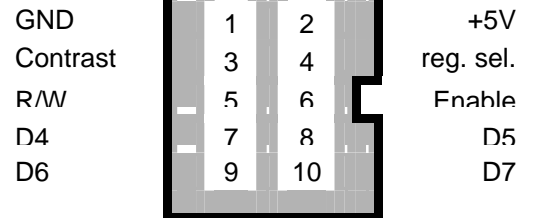


Name	Pin	Periphery	Description
J1	PD0	1-Wire	Dallas 1-Wire pin
J2	PD1	SH. LCD SET	LCD RCK (Register Clock) write shifted byte to output of the LCD register
J3	PD2	SH. IN LOAD	Load binary inputs to the input shift register (RxD data input from the Serial port 1 combined)
J4	PD3		RxD - serial port 1 asynchronous data input. Not used for SHIFT peripherals.
J5	PD4	SHIFT INPUT	Serial data input from the input shift register for reading SW1
J6	PD5	SH. LED SET	LED RCK (Register Clock) write shifted byte to output of the LED register
J7	PD6	SHIFT OUT	All SHIFT registers serial data output signal
J8	PD7	SHIFT CLK	All SHIFT registers data clock signal
J9	GND	Periph. Enable	Enable shifted peripherals – no jumper = all SHIFT registers disabled
	SH. LCD SOUT	SH. LED SOUT	Serial data outputs from the LCD and LED shift registers. Don't short!

18 – LCD display connector

Shifted output for connection an intelligent LCD display based on the HD44780 and compatible drivers.

You can connect LCD display pins directly to the connector; contrast trim is found on the board. Backlight connection depends on the specific type of LCD display.



Parallel output for the LCD display is available only if J2, J7, J8 & J9 are shorted. Pinout of common LCD 16x2 displays distributed with the Development Board shown below.



19 – PB6 & PB7 LED indicator

The last 2 LEDs of the 10 LED bar graph (the first 8 LEDs are used as binary output port indicator) can be set to indicate logic value on pins PB6 (JMP2) and PB7 (JMP1 – SETUP) .

For enable indication of these port pins, solder the “LED_INT1” and “LED_T0” jumpers.

20 – Shift register parallel input switch

8.bit binary input switch for testing purposes, controlled by SHIFT register IC1 – 74165. D0 input is on the LEFT side, defined as 1, under D0. Parallel input is available only if J3, J5, J7,J8 & J9 are shorted

21– Shift register parallel output LEDs

8.bit binary output port for testing purposes, displayed by first 8 LEDs of the 10 LED bar graph. Output is controlled by SHIFT register IC4 – 74595. D0 output is on the LEFT side, defined as 0. Parallel output LEDs are available only if J6, J7,J8 & J9 are shorted.

22 – LED indicators

Power LED (LED2) is directly connected to the power supply. It is lit when power is applied to the board.

LINK LED is used to indicate activity on the Ethernet port. The yellow LED (LED3) indicates the 10BASE-T link status and should be lit, if the link status is OK.

If the 93C46 eeprom is not soldered on the Charon II module, this LED indicates receive and transmit activity from and to the network.

Getting Started

The Charon II modules in the "Development Kit" pack are distributed with the Development Board, 2 x 16 LCD display, DS1822 Thermometer and Charon II module is programmed to the "**Charon II DB demo**" firmware. This chapter shows you how to test the Development Board and all peripherals.

The Development Board contains frequently used peripherals: Ethernet connection (10Mbit twisted pair), LCD display connector, RS232 line, thermometer(s) connection, LED diodes as outputs, DIP switches as inputs, I/O pins on connectors, setup jumpers. The Described developed application "**Charon II DB demo**" shows an example of how to use this module in embedded TCP/IP devices in a simple way - to control Charon II peripherals located on Development Board from an HTML page in a browser

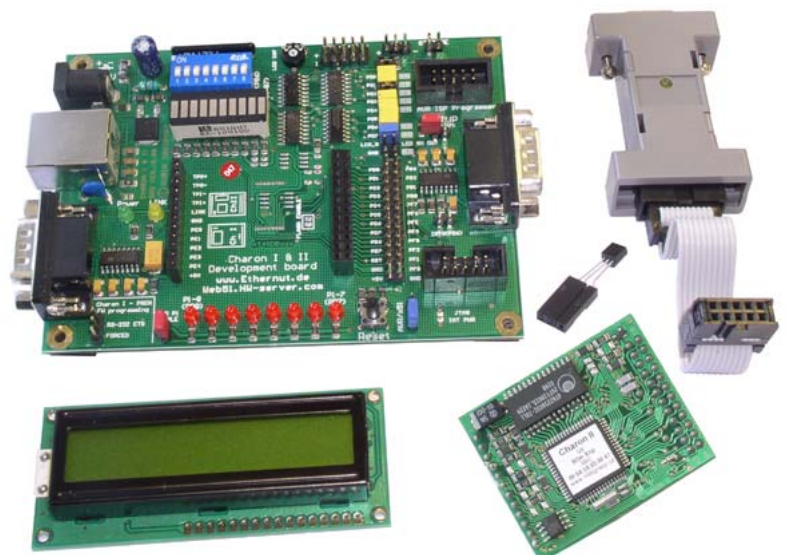
Prerequisites for Operation

The following hardware items are necessary to run the Charon II Development Kit:

- A standard PC equipped with Linux or Windows 95/98/NT/2000, an available serial COM port and a twisted pair Ethernet adapter card.
- Terminal emulation software, such as Hyperterminal (Heracles SETUP on the CD).
- An unregulated power supply matching your local mains. It should supply DC 8-16V, 150 mA minimum, on a standard 2.1 mm barrel connector.
- Two straight through twisted pair cables together with 10 Base-T hub or switch or a twisted pair cross cable, if you don't have a hub or switch.

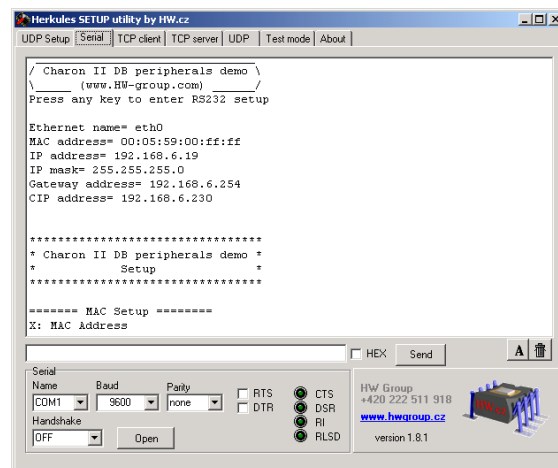
The following items are included in the Charon II Development Kit

- The **Charon II** module with MAC address and serial number on the label.
- The **Charon I&II Development Board** (Shortly Development Board only).
- An **HW STK-500** compatible programming adapter for serial RS-232 port.
- A **LapLink serial communication cable** with a DB-9 female socket on both ends.
- The **DS1822** 1-Wire thermometer sensor
- A **LCD display 2x16**
- **CD** with all necessary software in the **/Charon2/** directory



You will need a serial terminal and TELNET program for setting up the module. You can download our latest **Hercules setup utility** version from the link below. (<http://www.hw-group.com/download.html#setup>).

You can also use Hyperterminal, but some problems can arise, so we recommend the use of **Tera Term Pro**, which you can use as a RS-232 link terminal and for Telnet relations. (You can download it from our web pages www.hw-group.com).



Power supply



The development kit is powered by **DC 9-15V** current using a standard 2.1 mm power connector. The maximum consumption is up to 150mA.

Connecting the RS232 and Ethernet

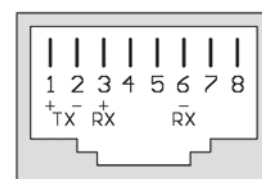
Use a LapLink serial cable with Cannon 9F Connectors for the RS232 link connection with the PC. If there is no free Cannon 9 serial port on your PC, you will need to use a 9/25 reduction.

The pin wiring on the board is in the left table. The second table describes the PC serial port wiring.

PC RS232 Port Cannon 9 - Male		Development Kit Cannon 9 - Male	
Pin	Signal	Pin	Signal
1 <-	CD	1	
2 <-	RxD	2 <-	RxD
3 ->	TxD	3 ->	TxD
4 ->	DTR	4	
5 --	GND	5 --	GND
6 <-	DSR	6	
7 ->	RTS	7 ->	RTS
8 <-	CTS	8 <-	CTS
9 <-	RI	9	

The Ethernet connection

- **HUB, Switch, Bridge** : Direct TP Twist pair cable (called PATCH cable)
- **PC or any other end device**: Twisted cable (The connectors have different wire colors and switched TX and RX wires.)



How to set up the Development Board before first power up?

Before you run the Charon module for the first time, please check the jumpers on the Development Board. You can see the default state in the photo on the first page.

- **5 – Charon I PSEN** – no jumper
- **8 – polarity jumper** – AVR position
- **13 – PB7 / SETUP mode JMP1** jumper – shorted
- **4 – RS-232 (Serial port 0)** – connect to the PC COM port with serial LapLink crossed cable.

Run the RS-232 terminal program (**Heracles** or **Tera Term** for example) with port parameters: 9600Bd, no parity, 8 data bits, 1 stop bit (**9600 8N1**). Switch the handshake control off (**Flow : NONE**).

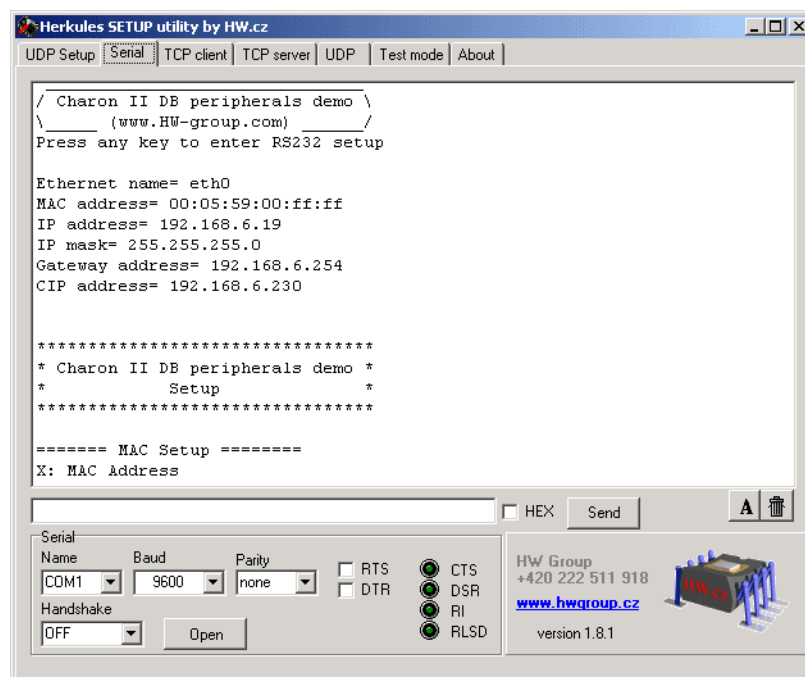
Connecting the supply – RS-232 SETUP

Connect the power supply connector – The green LED is lite. The first page of the KIT's setup should be displayed in the opened serial terminal.

RS-232 SETUP is intended for use in setting Charon II network parameters. Before you connect the Charon II Development Kit into the network, the parameters should be set to the correct values for proper operation on the network (accessing the Charon II Development board from network).

You can see this page on the picture, follow the instructions and "Press any key to enter RS232 setup".

The actual configuration is printed, and can be seen in the easy menu allowing change of basic network parameters.



When the *Charon II DB peripherals demo firmware* is loaded into the microcontroller, these network parameters are set to default values: MAC= 00-0A-59-03-00-5F , IP=192.168.1.100 , IP mask=255.255.255.0 , IP gateway=192.168.1.1 .

Just press "I" or "i" (non case sensitive) to assign an IP address for the device.

„IP [xxx.xxx.xxx.xxx]=“ dot separators are inserted automatically, if value is 100 or greater. After value entry, the value is stored into EEPROM immediately. Set your IP address, Gateway and Mask.

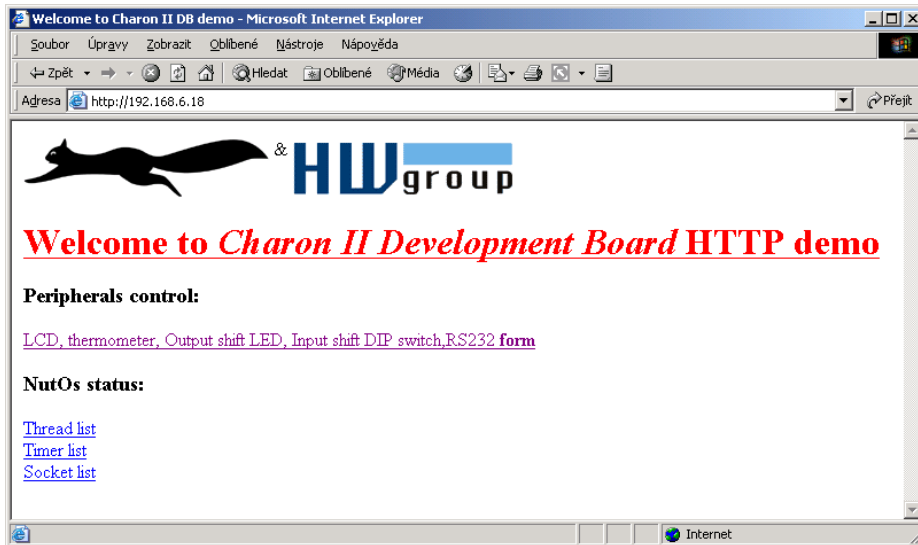
Remove the "13 – PB7 / SETUP JMP1 jumper" and press "R" for "R: Reboot (exit setup)".

Note: During the RS-232 Setup mode there isn't running the TCP/IP connectivity. You have to reset device without shorted the "13 – PB7 / SETUP JMP1 jumper" or exit from the RS-232 Setup after saving changed parameters.

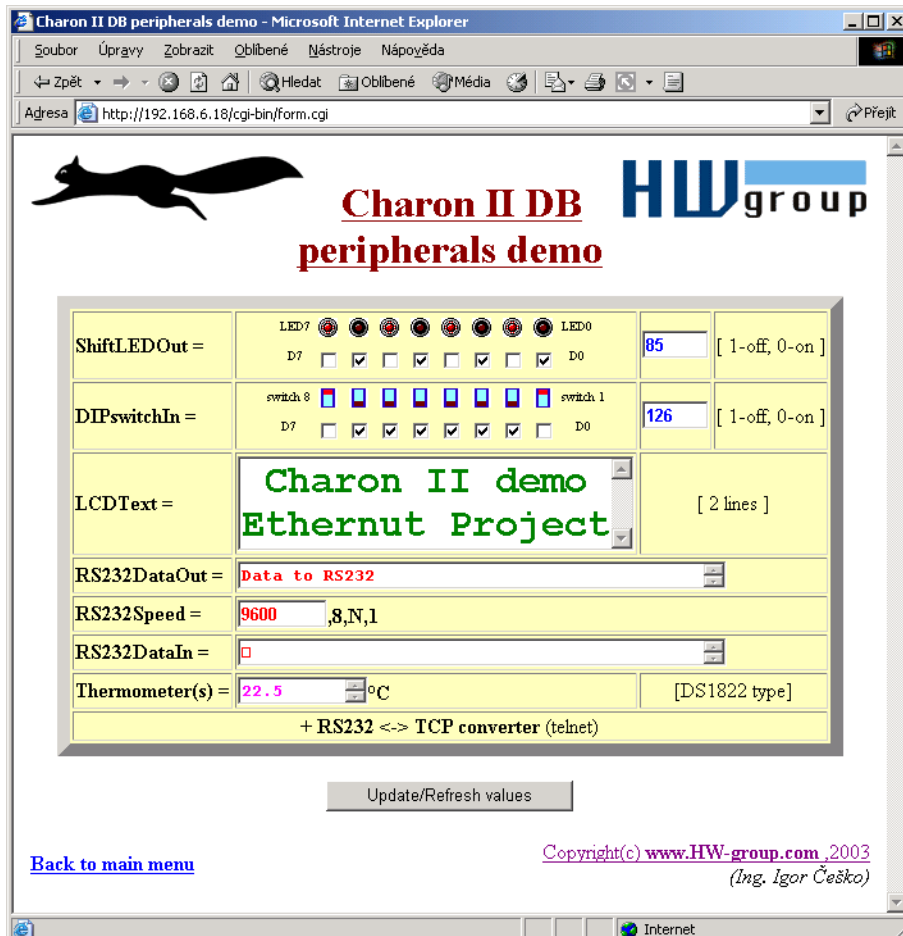
Testing HTTP demo

The user can access *Charon II DB peripherals demo* HTML page by an internet browser. Simply write the current Charon II IP address into the browser and the following page appears. The default IP address =192.168.1.100 , in browser: <http://192.168.1.100>.

There is a 'welcome' HTML page and from this page the user can enter to other pages (links on page):

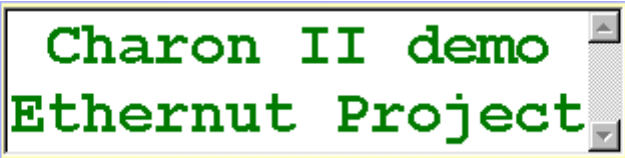


On *NutOs status* HTML pages we can see some important NutOs operating system information. The main purpose of the implemented application on Charon II is peripheral control. Therefore the most important link is *Peripheral control* to show the following HTML page:




Charon II DB peripherals demo

LCD display control

LCDText =		[2 lines]
-----------	--	-------------

On the webpage is an edit box to control displayed text on the external LCD display connected to the Development Board. This edit box has dimensions of 2x16 chars - the same as the connected LCD display to ensure similar visualisation

LED diodes - output pins control

ShiftLEDOut =		<input type="text" value="254"/> [1-off, 0-on]
---------------	--	--

LED diodes present on the Development Board (as LED-bar graph) are controlled with 8 checkboxes. Each checkbox has a corresponding picture which shows the required (or present) LED state. Checkbox state represents the bit value of the output shift register - if it is checked then given bit/pin is in log.1 state and if unchecked then given bit/pin is in log.0 state. Because LED diodes are connected to VCC then LED states are opposite the checkbox states. But the pictures display the real LED states.

Output shift register port state is displayed in one edit box as a decimal value.

DIP switches - input pins reading

DIPswitchIn =		<input type="text" value="8"/> [1-off, 0-on]
---------------	--	--

DIP switches present on the Development Board are monitored within 8 checkboxes and 8 corresponding pictures. Each checkbox has a corresponding picture, which shows the present DIP switch state. The checkbox state represents the bit value of the input shift register - if it is checked then the given bit/pin is in log.1 state and if it is unchecked then the given bit/pin is in log.0 state. Because the DIP switch is connected to GND (when it is switched on the input pin is connected to GND) then DIP switch states are negated checkbox states. But picture displays are the real DIP switch states.

Input shift register port state is displayed in one edit box as a decimal value.

RS232 data output

RS232DataOut =	<input type="text" value="Data to RS232"/>
----------------	--

The user can send some data (text) from this HTML page to the RS232 port located on the Development Board. The user writes the required text in the edit box and in the next update the text is transmitted to the RS232 (at baud rate given in RS232 baud rate box setting).

Because this RS232 data is sent to Charon II with the GET method and Ethernut supports a maximum of 256 bytes received by the GET or POST methods, length of this RS232 data output is limited to cca 70 bytes (because the whole 256 byte GET message also includes other settings: LED, DIP switch,...).

RS232 data input

RS232DataIn =

When Charon II receives some data from RS232 the user can display this data (text) on the HTML page in the supplied edit box. This data is updated with every refresh or update of the HTML page. Charon II buffers a maximum of 255 bytes of received chars. If no data was received from the last refresh/update then the previously received data remains displayed.

RS232 baudrate control

RS232Speed = ,8,N,1

Charon II RS232 line baudrate is controlled through the given edit box on the HTML page. When we transmit some data from the HTML page to RS232 line, data is transmitted at the given baud rate. After refresh/update of the HTML page this box displays the current baud rate (therefore speed may be slightly different as required due to limited step in Charon II UART baud rate setting).

Thermometer(s) reading

Thermometer(s) = °C [DS1822 type]

The Charon II Development Board has a connector for connecting 1-wire thermometers. The Charon II measures temperature from all thermometers every 1 second. The number of connected thermometers is automatically detected. The user can read the thermometers values on the HTML page. The values are displayed in the edit box on the HTML page - in the case of multiple connected thermometers, the temperatures are displayed on multiple lines.

In the present firmware, the maximum number of thermometers that can be connected is 5 (but may be set to different value in Charon II firmware).

RS232 to TCP port 23 (telnet) link

+ RS232 <-> TCP converter (telnet)

On the HTML page is displayed information that Charon II works also as a converter between the RS232 line and TCP (port 23 = telnet). This converter works independently from controlling other peripherals (except baud rate of RS232 whose changes are reflected immediately).

Update and refresh values on HTML page

Update/Refresh values

All required values are sent to the Charon II Development Boards peripherals by the **Update/Refresh values** button. Then the required data is submitted by the GET method to the Charon II which works as a server. The Charon II sets required values, collects required information and displays the modified HTML page with current peripheral values.

Additional pictures and links



**Charon II DB
peripherals demo**



The HTML page shows 2 pictures - logos of the main firmware/software developer companies. These pictures are links to their homepages where the user can obtain more information about embedded Ethernet (products, examples, solutions, ..).

On the top left corner of the page is a link to the *Charon II Developer Board demo* homepage. Use this link to see additional information about Charon II current operating system state.

Bottom right is the link to www.HW-group.com, author of this page and developers names.

Description of RS232-telnet converter

RS232 to TCP port 23 (telnet) link

Charon II DB peripherals demo firmware works in the background as RS232 to telnet (TCP port 23) converter. To display functionality of this converter simply run *telnet* on a computer on the network and connect to a given IP address (default IP=192.168.1.100). Then connect a computer with a running RS232 terminal to the Charon II Development Board serial port (default serial port parameters are **9600,8,N,1**, but the baud rate can be changed from the HTML page). After this action the telnet client is connected with the terminal program through the network. The user can write something into the terminal and Charon II sends this data to the telnet client (data appears in telnet window). This also works in reverse (from telnet to RS232).

Description of firmware

Basic principle of firmware is to perform some actions as a result of user requirements and some background services. The background is running in threads:

- Thermometers reading: *Thermo thread*
- RS232 receiving and RS232 to TCP converter: *Receiver thread*
- TCP to RS232 converter: *Main program loop*
- HTML pages processing: Service threads *httpd1, httpd2, httpd3, httpd4*

When firmware starts it firstly opens the network and UART device. It initializes UART to default speed 9600 baudrate and loads network parameters from EEPROM. Then the *Setup jumper is checked* and if present runs *RS232 setup*. Starts *Receiver thread* to receiving data from RS232 line and starts

Thermo thread reads temperatures from thermosensors. Initializes network to parameters stored in EEPROM. Then register CGI samples to display web pages (*NutOs status* pages and *Charon II DB peripherals demo* page). Next the 4 threads are created for processing server HTTP responses (threads are named *httpd1, httpd2, httpd3, httpd4*). The LCD display is then initialized to default value: startup text is displayed. At the end of main routine is endless loop for processing TCP to RS232 transfer (this can be made in independent thread, but in this case the main program routine will have nothing to do).

LCD display control

Output to LCD display is performed during HTML page response in *ShowForm* routine. Required text to LCD is sent as parameter "LCDtext" from HTML page. The string value of this parameter is firstly filtered by the *ReplaceHTMLStr* routine to correct conversions to true text chars. Then actions for "intelligent" separation of given text to 2 lines on the LCD display is performed. Finally initialization of LCD display (*LCD_Init* function) and writing required text to this LCD (*LCD_Puts* function) takes place. Initialization is necessary because in the case of the LCD being removed, no firmware restart is required.

LED diodes - output pins control

LED diode (in bargraph) control is performed during HTML page response in *ShowForm* routine. Required values of the LEDs is sent as parameter "ShiftLEDOut" from HTML page. According to this parameter, value simply sets the states of LED diodes with function *DevBoardShiftLedOut* (because LEDs are connected to outputs of shift register on Development Board).

DIP switches - input pins reading

DIP switch state is captured during HTML page response in *ShowForm* routine. Capture is performed by function *DevBoardShiftByteIn* because DIP switches are connected to inputs of the shift register on the Development Board. State is finally inserted into HTML page.

RS232 data output

RS232 data output

Sending data to the RS232 line is performed during HTML page response in *ShowForm* routine. Required data is sent as parameter "RS232DataOut" from HTML page. These parameter values are firstly filtered by function *ReplaceHTMLStr* and then sent to Charon II UART (*_write* function).

RS232 data input

RS232 receiving is processed in the background in thread *Receiver*. This thread simply reads data from RS232 line and if something is received, sends received data to TCP port (if client is connected) and also stores data to buffer. This buffer *rxbuff* can be read by user into HTML page response. Buffer *rxbuff* has limited size - 255 bytes.

Reading RS232 data is performed during HTML page response in *ShowForm* routine. Content of *rxbuff* buffer is inserted to HTML page.

RS232 baudrate control

Change of baud rate on RS232 line is performed during HTML page response in *ShowForm* routine. Required baud rate is sent as parameter "RS232Speed" from HTML page. According to this parameter value is set the UART baud rate (*_ioctl* function).

Thermometer(s) reading

1-wire thermometers (Dallas DS1822 or DS1820 type) capturing is processed in the background in thread *Thermo*. This thread works in an endless loop: Firstly detecting all connected thermometers (*TM_Init* function) and then reads all detected thermometer values (*TM_Read* function). The temperatures are stored into buffer *Temperatures*. Then the thread sleeps for 1 second and repeats this sequence. Therefore new connected thermometers are detected automatically (the present firmware supports connection of up to 5 temperature sensors). From the buffer, the user reads temperature values during HTML page response.

Reading temperatures is performed during HTML page response in *ShowForm* routine. *Temperatures* buffer is read and its values are converted to temperatures and inserted in to HTML page.

RS232 - TCP port 23 (telnet) converter

This conversion is performed in two independent threads.

- RS232 to TCP conversion is processed as part of the RS232 receive in thread *Receiver*. This thread simply reads from RS232 line and if something is received, sends received data to TCP port (if client is connected) and then stores data into buffer *rxbuff* for HTML page response.
- TCP to RS232 conversion is made in main program. The main program finally performs a loop where it opens socket (*NutTcpCreateSocket* function) and then waits for client connections (*NutTcpAccept* function). When connection is established the stream for reading data from TCP port is opened. During connection, all received data from TCP port is transmitted to UART (in function *StreamCopy*). When disconnection occurs, socket is closed and this loop is restarted.

RS232 setup: Ethernet, TCP/IP parameters

RS232 setup routine is performed in the independent function *RS232Setup*. Here the RS232 menu is displayed and is where user input/output is processed.. User changes and changed network parameters and stored into EEPROM. To Exit from this routine and continue in firmware is possible using the "R" option in *RS232 setup*. To enter into this routine, the user must short the setup jumper and reset firmware (by reset button) on the development board.

HTML pages and pictures

All HTML pages present in firmware are stored in project *HTMLdir* directory. These pages and pictures are used as templates for generating HTML pages. To simplify HTML page generation we choose the following principle:

Charon II DB peripherals demo firmware only insert to this template HTML page a couple of variables in form : **variable = value** . Charon finds the position of insertion of this small number of variables according to the first presence of char "@". Therefore insertion place must be near the top of the page (to ensure removing other "@" char collisions). Position place is set into JavaScript function which according to this value, builds the whole HTML page visualisation: set checkboxes as bits in byte value, loads corresponding pictures to checked/unchecked checkbox, displays corresponding text to edit boxes (including some testing procedures and conversions), JavaScript functions present in HTML page performs some additional functions: Edit boxes height settings according to text length, removing unsupported chars from edit box (e.g. only numbers are valid in baud rate settings) and formatting of submitted parameters.

Using this technique we transfer complicated page layout processing/testing to the client side, which usually has more performance than embedded Ethernet server. Disadvantage of this method is that the client must support JavaScript language - but currently JavaScript is present in all commonly used commercial browsers.

Part of HTML template page for inserting variables:

```

...
ShiftLEDOut      = "<#ShiftLEDOut> ";
ShiftLEDIn       = "<#ShiftLEDIn> ";
LCDText          = "<#LCDText> ";
RS232DataOut     = "<#RS232DataOut>";
RS232Speed       = "<#RS232Speed> ";
RS232DataIn      = "<#RS232DataIn> ";
Thermometer      = "<#Thermometer> ";

//not remove next line - this char is mark for Charon insert data command
//@
if ((ShiftLEDOut == "<#ShiftLEDOut> ") || (ShiftLEDOut==null)) ShiftLEDOut = 0xAA;
if ((ShiftLEDIn == "<#ShiftLEDIn> ") || (ShiftLEDIn==null)) ShiftLEDIn = 0x55;
if ((LCDText == "<#LCDText> ") || (LCDText==null))LCDText = " LCD Text"+"\\n"+
"Ethernut project";

if ((RS232DataOut=="<#RS232DataOut>") || (RS232DataOut==null))RS232DataOut = "RS232
Data Out" + "\\n" + "0123456789";

if ((RS232Speed == "<#RS232Speed>") || (RS232Speed==null))RS232Speed = 9600;
if ((RS232DataIn == "<#RS232DataIn> ") || (RS232DataIn ==null))RS232DataIn = "RS232
Data In" + "\\n" + "0123456789";

if ((Thermometer == "<#Thermometer> ") || (Thermometer==null))Thermometer = 0.0001 +
"\\n" + 0.0002 + "\\n" + 0.0003;
...

```

And part of firmware source code to insert variables into template:

```

...
while (NutRomFileRead(DemoFile1,&DemoData,1)>0){
  if ((DemoData!='@') || (!EnableParse)){
    fprintf_P(stream, PSTR("%c"), DemoData);
  }
  else{
    u_int i,j;
    EnableParse=0;
    fprintf_P(stream, PSTR("\\r\\n"));
    /*ShiftLEDOut*/
    fprintf_P(stream, PSTR("ShiftLEDOut= %d;\\r\\n") , ShiftLEDOut);
    /*ShiftLEDIn*/
    fprintf_P(stream, PSTR("ShiftLEDIn = %d;\\r\\n") , ShiftLEDIn);
    /*LCDtext*/
    j=strlen(LCDText);
    fprintf_P(stream, PSTR("LCDText =unescape(\\"));
    for (i=0;i<j;i++){
      fprintf_P(stream, PSTR("%%02x"),LCDText[i]);
    }
    fprintf_P(stream, PSTR("\\");\\r\\n"));
    /*RS232DataOut*/
    j=strlen(RS232DataOut);
    fprintf_P(stream, PSTR("RS232DataOut =unescape(\\"));
    for (i=0;i<j;i++){
      fprintf_P(stream, PSTR("%%02x"),RS232DataOut[i]);
    }
    fprintf_P(stream, PSTR("\\");\\r\\n"));
    /*RS232Speed*/
    fprintf_P(stream, PSTR("RS232Speed =%lu;\\r\\n") , RS232Speed);
    ...
  }
}

```

And after processing the template page (variables are modified):

```

...
ShiftLEDOut      = "<#ShiftLEDOut> ";
ShiftLEDIn       = "<#ShiftLEDIn> ";
LCDText          = "<#LCDText> ";
RS232DataOut     = "<#RS232DataOut>";
RS232Speed       = "<#RS232Speed> ";
RS232DataIn      = "<#RS232DataIn> ";
Thermometer      = "<#Thermometer> ";
//not remove next line - this char is mark for Charon insert data command
//
ShiftLEDOut= 254;
ShiftLEDIn = 8;
LCDText=unescape( "%20%43%68%61%72%6f%6e%20%49%49%20%64%65%6d%6f%20%0a%45%74%68%65%
72%6e%75%74%20%50%72%6f%6a%65%63%74" );
RS232DataOut =unescape( "%44%61%74%61%20%74%6f%20%52%53%32%33%32" );
RS232Speed =9601;
RS232DataIn =unescape( "" );
Thermometer = 26.6250 ;
;
if ((ShiftLEDOut == "<#ShiftLEDOut>") || (ShiftLEDOut==null)) ShiftLEDOut = 0xAA;
if ((ShiftLEDIn == "<#ShiftLEDIn> ") || (ShiftLEDIn ==null)) ShiftLEDIn = 0x55;
if ((LCDText == "<#LCDText> ") || (LCDText ==null)) LCDText = " LCD
Text"+"\\n"+"Ethernut project";

if ((RS232DataOut=="<#RS232DataOut>") || (RS232DataOut==null)) RS232DataOut= "RS232
Data Out" + "\\n" + "0123456789";

if ((RS232Speed == "<#RS232Speed> ") || (RS232Speed ==null)) RS232Speed= 9600;
if ((RS232DataIn == "<#RS232DataIn> ") || (RS232DataIn ==null)) RS232DataIn= "RS232
Data In" + "\\n" + "0123456789";

if ((Thermometer == "<#Thermometer> ") || (Thermometer ==null)) Thermometer= 0.0001 +
"\\n" + 0.0002 + "\\n" + 0.0003;
...

```

Additional information

All firmware is written for use with compiler **WinAVR** and **NutOs version 3.3.0**. For use with another compiler it is necessary to rewrite some compiler unsupported functions (e.g. as *strtok_r* function).

Project files must be located in directory `\\ethernut\\nutlapp<ProjectDirectory>` to ensure correct inclusion of some headers.

Project was created by *Ing. Igor Cesko* for the [HW-group](http://www.HW-group.com) company. Additional routines for LCD display control, shift registers control, 1-wire thermometers reading, threads processing, UART control, TCP control was collected from www.HW-group.com resources and examples included in **NutOS operating system** www.ethernut.de.

How to compile Charon II DB peripherals demo

For your own Charon II Development Board peripherals demo code (shortly **db_demo**), this compile and development applications list can be useful.

Following version you can find on the CD, they are related to the "Charon II DB peripherals demo" version 1.2.0. There is important to understand, that it's not automatic, you can compile this code with higher SW version without any troubles, so it's strictly recommended to install older versions from CD first and when you are oriented, upgrade your system...

What you have to install

1. WinAVR GNU C compiler

[/Charon2/Prog_and_Utils/WinAVR-20040404-bin-install.exe](#)

It's a free compiler (you can download it from the <http://winavr.sourceforge.net/>), with various language support. During the installation process leave the checkbox for automatic set environment variables checked. We recommend to check if the installation process extended paths: SET PATH=C:\WinAVR\bin;C:\WinAVR\utils\bin;%PATH%

2. Ethernut system: Nut/OS for Win32 Environment

[/Charon2/Prog_and_Utils/nut342c.exe](#)

It's the core of the Ethernut OS and TCP/IP stack. You can find actual version on the <http://www.ethernut.de/> original website. We recommend to install it to the default directory [C:\ethernut\nut\](#)

NOTE: Charon 2 is HW compatible with the **Ethernut 1.3** board, use it in the configuration!

3. Ethernut System Documentation

[/Charon2/Prog_and_Utils/nut342d.exe](#)

It's the Documentation core of the Ethernut OS and TCP/IP stack. We recommend to install it to the default directory [C:\ethernut\nut\doc](#)

4. Programmer: AVR Studio 4.0 - stk500.exe

[/Charon2/Prog_and_Utils/AVRStudio4_7_240.exe](#)

WARNING: Install the AVR Studio 3.56 or 4.07, both working fine with the attached **HW STK 500 dongle**. With higher AVR Studio versions (higher than 4.07) there can be troubles with STK 500 FW version!

5. Create the \db_demo\ directory in the Ethernut application directory

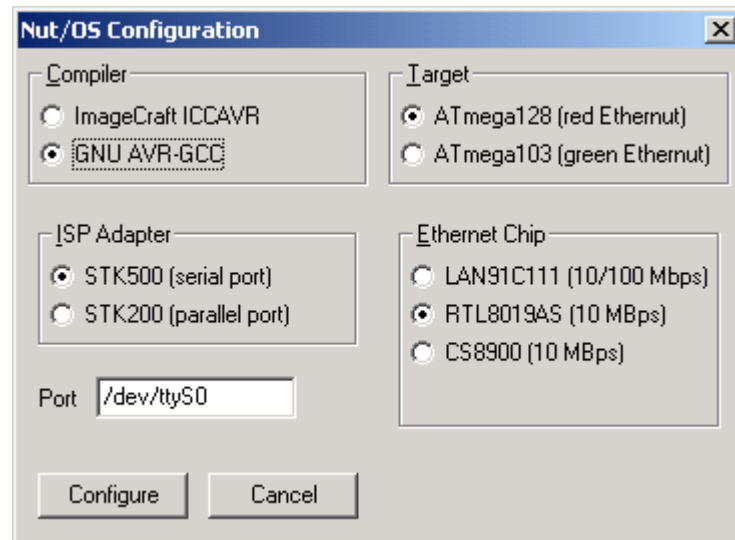
(C:\ethernut\nut\app\db_demo\) and copy the files from the \app\ directory on the CD (\Charon2\Charon2_DB_Demo\app\db_demo\) there.

6. If you prefer to use the AVRICC ImageCrat compiler, don't forget to copy the files from the \appicc\ db_demo's directory on the CD to the Ethernut separated directory for the AVRICC project management (C:\ethernut\nut\appicc\db_demo\)

7. If you want to use USB JTAG dongle with cooperation of AVR Studio for debugging, you'll need to install FTDI virtual serial port drivers (\Charon2\Prog_and_Utils\USB_JTAG\).

How to setup and use the installed system

- **Configure your installed Ethernut** - run the file `C:\ethernut\nut\nutconf.exe` and configure system (Charon II = GNU AVR-GCC, ATmega 128, RTL8019AS, STK500 Port `"/dev/ttyS0" = COM1`, `"/dev/ttyS1" = COM2`).
The system is recompiled, if not, please check the PATH (see the `"C:\ethernut\nut\doc\api\html\compiler.html"` for more information).



- **Try to compile SIMPLE** application to the `"C:\ethernut\nut\app\simple\"` directory, open command line window (run "CMD" in this directory). Try to write "make" and this simple example should be compiled and **simple.hex** should be created in this directory. If you have troubles with it, check if your Ethernut is configured to the GNU C compiler.
- **Compile and burn**
You can specify where is your dongle HW STK500 programmer installed (`"/dev/ttyS0" = COM1`), if you call "make burn", the created .hex file will be programmed with using HW STK500 over ISP interface to the MCU.

Using ImageCraft C (ICCAVR)

In opposite to the Open Source Compiler AVR-GCC, ImageCraft's commercial compiler ICCAVR comes with its own IDE (Integrated Development Environment). It can be more easy, to use the ICCAVR, especially for beginners.

Nut/OS libraries for AVR-GCC and ICCAVR are based on the same source code. A full featured demo version of ICCAVR can be downloaded from the ImageCraft (www.imagecraft.com).

You can find detailed description how to use the ICCAVR on the website (www.ethernut.de/en/iccavr/), but in general, you have to configure installed Ethernut to ICCAVR, set rights paths, copy some libraries from original nut directory to the ICCAVR, Configure Compiler Options and Target Options (ATmega128..). Now you can compile your own project with using ICCAVR.

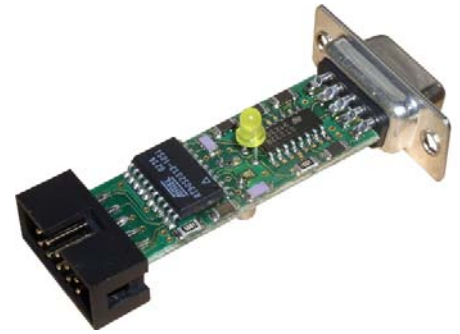
Any troubles?

Check the original installed Ethernut documentations (`C:\ethernut\nut\doc\`) or our website (www.HW-group.com).

Using the HW STK500 Programmer

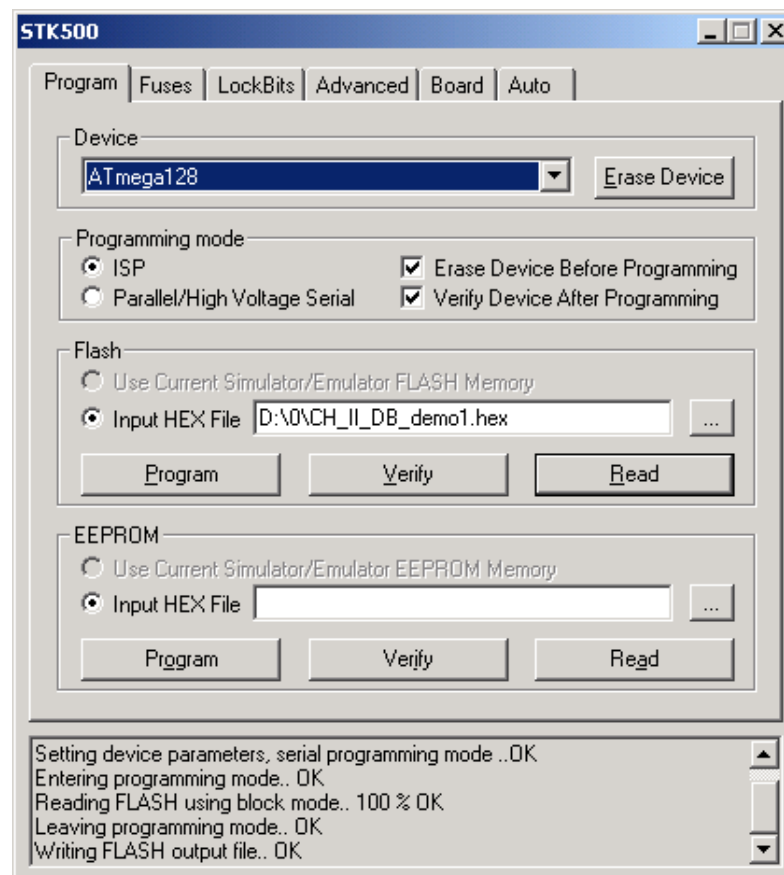
Introduction

This chapter presents some information about how to upload your software into the Charon II - Ethernet or any other AVR microcontroller board. A not-so-simple programming adapter is presented, which enables you to program the target board using the RS232 interface (aka. COM port) of your PC.



Note, that the hardware of this adapter should work for almost any AVR target, but the software is quite limited. It has been tested with ATmega103 and ATmega128 chips only. While programming flash memory, fuses and lock bits of these MCUs works fine, the current version also fails programming the on-chip EEPROM.

Another word of caution. Playing around with the software inside your programming adapter or with the ATmega fuse settings is risky if you don't take special care. Always make sure, that you have a second programming alternative or a Charon module with a working bootloader, if you change the adapter software. Also take care not to disable the ATmega oscillator, when reprogramming the fuse registers.



HW STK500 limitations

- It programmes ATmega103 and **ATmega128 only!**
- It works with the **AVR studio 3.56** and **4.06** or higher only!
- You can't program internal **EEPROM** with this dongle.
- You can use **ImageCraft**, because it calls STK500.EXE from your AVR Studio.

Atmel AVR ISP Programming generally

There are several methods to upload your software to a target device like an Ethernet Board. One of the most advanced is using an [Ethernet bootloader](http://www.ethernut.de/en/eboot/) (<http://www.ethernut.de/en/eboot/>) based on the DHCP, BOOTP and TFTP protocols. But how do we get the bootloader software into the target system?

This requires special hardware called a programming adapter, also called programming **dongle**. Programming AVR devices like the ATmega128 is fairly simple. There is no need for special programming voltage supply and timing is not very strict as long as you don't violate the speed limit. Note however, that AVR chips typically support two programming modes, parallel and serial programming. Parallel programming is very fast and excellent for volume production, but it requires an additional 12 Volt supply and is most often done before soldering the chip onto the PCB. The simple serial programming method is the first choice for low volume production and during development. Just three pins are used in this mode, an input line, an output line and a clock line. In addition, the RESET line of the chip must be held low during programming. Because serial programming is done while the chip is already soldered onto the target board, it is also called **In-System Programming** or ISP. Many AVR boards are equipped with a 6 or 10-pin connector, where you can plug-in the programming adapter. The 6-pin version had been used initially by Atmel, but they later switched to the 10-pin version which provides additional ground lines. Originally - Ethernet system uses the 10-pin version.

On the ATmega128 the ISP input and output lines are shared with the transmit and receive lines of the first on-chip USART. This adds a minor problem. As long as the adapter is connected, the output line of the adapter shares the same MCU input line as the RS-232 receiver output, which is included on almost any ATmega128 board. To overcome this, Atmel used an additional line called programming enable or programming LED. The programming software on the PC will set this line low before starting the programming cycle. This line can be used to switch the pins on the ATmega from the RS-232 driver to the ISP connector. On Ethernet version 1.3 or Charon I&II Development Board this is done by a multiplexer chip and the line will also lite the red programming LED. But not all programming adapters provide this signal, so on Ethernet 1.3 (**15 – ISP LED & STK500 programming jumper** on the Development Board) a jumper has been added to manually pull this line low.

JTAG Programming

JTAG is completely different from ISP. It can not only program the target device, but adds additional hardware and software debugging support. And it requires a more advanced programming adapter which costs much more than a simple ISP adapter. Atmel's "low cost" JTAG adapter, called **AT JTAGICE**, comes with an adapter cable, which can be used to connect it to an Ethernet 1.3 board with some trouble, but directly to the Charon I&II Development Board. On the Ethernet 2.0, the 10-pin connector can be switched between JTAG and ISP mode by a 20-pin jumper field. Changing the jumper isn't convenient, but typically done once only.

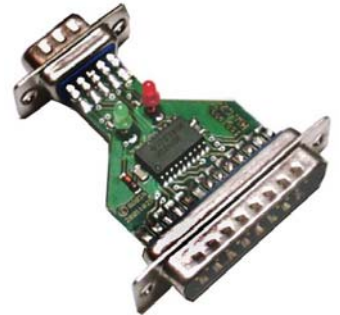
Due to their high price when compared to ISP adapters, JTAG adapters are mainly used for debugging.

ISP Adapters

The often used terms "serial or parallel programming adapter" may be misleading, because some ISP adapters use the PC's parallel port while others are connected to one of the PC's serial COM ports. But both of them use the AVR's serial programming mode.

Parallel port (LPT) ISP Adapters

ISP adapters for the PC parallel printer port are most simple. Some of them do not even need any active components. But you can imagine, that directly connecting your AVR board to the PC parallel port may cause problems. Therefore, most adapters use a line driver. But still this isn't very safe. The parallel port on the PC is not very well protected against shorts or overload and several people have killed this port by plugging the adapter in or out without switching off the Ethernet board power supply first. Very bad, when you do this to your notebook computer. Let's call these devices "**LPT adapters**"



But it is possible to order Charon II Development Kit with the LPT interface. You can use PonyProg (www.PonyProg.com) or UISP software for programming; it's slower but sometime useful.

Serial port (RS-232) ISP Adapters

In comparison to the parallel port, the serial port on the PC is very well protected. But again, there are two types of programming adapters for the COM port. The simpler type uses the port lines directly to generate the programming pulses for the AVR chip. Let's call these devices "**RS-232 adapters**"



The second type sends programming commands and program data to the adapter, which interprets them and accordingly generates the pulses. This adapter type needs some kind of processing capabilities, typically provided by a microcontroller built into the adapter. Let's call these devices "**RS-232 dongles**"

ISP Software

What ever the type of adapter you intend to use, you should check the available programming software too. There is no solution, that fits everything and a large variety of tools are available on the Internet. Luckily, almost all of them are free.

- Atmel offers **AVR Studio**. This isn't just simple programming software, but an integrated development environment with assembler, debugger, simulator, source code editor and other goodies. It's free, but only available for the Windows operating system. Two additional command line tools for ISP are included as well, STK500.EXE and AVRPROG.EXE.
[STK 500 – RS-232 dongle compatible only]
- Originally written by Uros Platise, **uisp** is a command line tool for the Linux operating system. It's currently maintained by Ted Roth and Marek Michalkiewicz and available at savannah.nongnu.org. A port to the Windows operating system is included in the [WinAVR distribution](#). Before Ted and Marek took over the uisp support, several programmers tried to enhance it and found it a hard nut to crack. The source code is a mixture of C and C++, handles many kinds of different hardware and is not really easy to follow.
[original STK 500 – RS-232 dongle and LPT adapters]

- The commercially supported **ICCAVR compiler IDE** offered by [ImageCraft](#) comes with it's own programmer GUI. However, it uses an external call to STK500.EXE to deal with STK500 compatible programmers until version 6.28. Therefore, AVR Studio had to be installed too. Since version 6.29, ICCAVR handles it internally.
[STK 500 – RS-232 dongle compatible only]
- Last not least, Claudio Lanconelli developed **PonyProg**, a powerful programming software for the Windows operating system. His company [LancOS](#) does not only offer free downloads of this software, but also low cost ISP adapters.
[Various LPT and RS-232 adapters]

HW STK 500 dongle

HW STK500 dongle contained in the Development Kit is a copy of the original **SISP** from the Ethernut Project. Please respect some non-compatibility troubles with original STK500 adapter. We are working on this, but it's only a tool for Charon II module programming..

For more detail about this, please check Ethernut project article:

ISP Adapter (<http://www.ethernut.de/en/isp/>)

Contacts and detailed information

Czech Republic	HW group Rumunská 26 / 122, Praha 2, Czech Republic Tel. +420 222 511 918, Fax. +420 222 513 833	http://www.HW-group.com/
Africa	TELEPAC Technology Morocco 21, Place Charles Nicole 20100 - Casablanca - Morocco Phone +212 22 889 889, Fax. +212 22 472 733	www.telepactechnology.com
Denmark	IT-Tronic Holmblædsvaenge 4, DK 2300 Copenhagen, Denmark. Tel.: +45 70 22 69 70, Fax: +45 70 22 69 80	www.IT-Tronic.com
France	LEXTRONIC 36/40 rue du général de Gaulle - 94510 LA QUEUE EN BRIE Tél.: +33 145 768 388, Fax: +33 145 768 141	www.lextronic.fr
Germany	egnite Software GmbH Westring 303, 44629 Herne, Germany Phone +49 232 392 53 75, Fax. +49 232 392 53 74	www.egnite.de
Hungary	PRIME Computer Ltd. 3530 Szemere str. 20., 3530 Miskolc Hungary Phone: +36 46 501-330, Fax: +36 46 501-333	www.prime.hu
United Kingdom	TR Control Solutions Global House, Ashley Avenue, Epsom, Surrey, KT18 5AD UK Phone: +44 208 823 9230, Fax: +44 208 823 9240	www.TRcontrolsolutions.com
USA	Capitol Automation 500 Main Street, Clinton, Ma. 01510, U.S.A. Phone: 800 550 9672 or +1 978-368-0116, Fax: 800-550-9672	www.CapitolAutomation.com