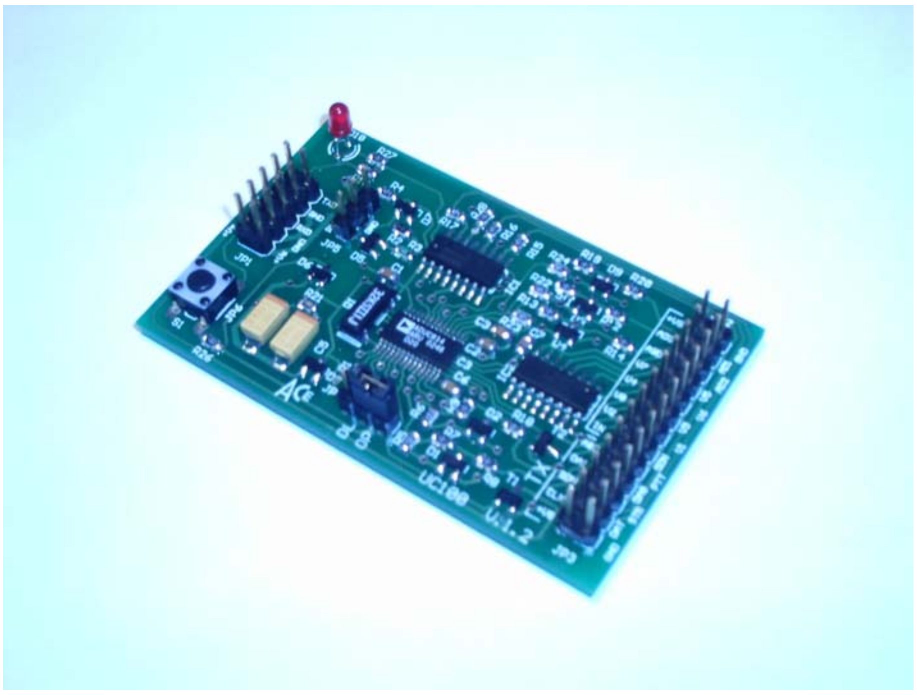
Universal Controller UC100

User`s Manual

HW Vers.1.2 SW Vers.1.01



Content

Short Description

The UC100 is a user programmable control unit optimized for transmitting information in wireless systems. It contains an 8051 compatible controller core featuring 2x 12 Bit DAC for generating modulation signals as well as 11x 12 Bit ADC Inputs for analog signal conversion. Download of the user program from the PC into the internal EEPROM is done via the RS-232 compatible interface. Only a standard Terminal Software is needed to correspond with the controller by using simple basic related instructions. After storing the user program inside the UC100 the controller is operating on it´s own. The ultra low power consumption of the module makes it perfectly suited for battery powered applications.

4 different interfaces to the environment are provided:

- Power Supply (3 … 5V DC, abt. 10mA maximum)
- Computer Interface (RS-232 compatible)
- Sensor interface (11x Analog In, Shutdown, +V)
- Connection to transmitter (Mod., PTT, ser.Bus, +V)

Programming variations:

- Output of fixed texts in morse code (A1, F1, F2)
- Output of fixed texts in PSK31
- Output of fixed texts in AX-25 Packet radio (AFSK, FSK)
- Output of numbers in DTMF Code
- Output of fixed frequencies (sinus)
- Output of current chip temperature
  in morse code, PSK31, DTMF, AX-25, LCD´s or serial (RS-232)
- Output of any applied analog voltage in modes shown above
- Input of analog voltages, mathematical conversion of them
  and output as above
- Conversion of analog signals into audio frequencies
- Output of combined fixed texts and variable values
- Generating alarm messages when breaking voltage limits
- Combinations of features mentioned above
- Operation in a program loop or activation by external DC or
  Audio signals
- Versatile usage due to individual programming

The intended transmitter can easily be adapted to the controller because all important parameters may be configured by simple instructions (for example modulation level, DC level, speed for morse transmission, baudrates for packet radio or AFSK frequencies)
When switched into Idle Mode the UC100 draws only a few Microamperes extending battery life time significantly. Additional hardware may also be switched into power down mode by using the provided "Shutdown" signal. PTT control is done by an internal FET capeable of switching currents up to 200mA.

Starting

1. Connecting the PC

Connect TXD (Pin 2), RXD (Pin 3) and GND (Pin 5) of the serial interface cable (9-pin DSub Connector) to the corresponding pins of the UC100.

2. PC – Software

If a suiting Terminal Software was shipped together with your UC100, install and use it for all programming and communication purposes between PC and UC100. All relevant information is contained in the README file.
But all other Terminal Prorams may be used unless they are able to communicate through a serial COM Port
Setting: 1200 Bps, 8 Data Bits, 1 Stop Bit, no Parity, no Echo , no Hardware Handshake
It is also very helpful to use a standard Text Editor for downloading stored User Programs.

3. Data Exchange

After linking PC and UC100 the Power Supply can be switched on. So a Reset pulse is generated starting any User Program already stored inside the internal Non Volatile Memory (Flash EEPROM).
Pressing the push button S1 on the UC100 PCB briefly switches into Program Mode. Now the Programming Menue will be displayed on the PC Monitor: **ram mode (list|run|line|ram|test|burn)**
The controller has now stopped running the user program and is ready for being programmed line by line via the keypad or for download of a user program established in the Text Editor.

4. Establishing a User Program

4.1 RAM – Mode

This mode can be used for quick download of simple User Test Programs. Data is stored inside the internal RAM. But because of the restricted RAM memory space programs are limited to about 30% of the maximum possible size. Instructions are kept inside the memory as long as the supply voltage is connected, data will be lost at switch off.

4.1.1 Input of instructions line by line

First enter the instruction **ram** or push S1 briefly. Now write the user program by typing instructions like shown in the instruction set. But take care of the correct Syntax (shown in chapter „Instruction Set"). If an instruction has been entered in a wrong way the UC100 will respond with an Error message. But after that you can continue with the correct instruction without any problems.
If memory overflow occurs you will get the information "memory full", it is not possible to continue now in RAM mode unless the number of program lines is reduced. Otherwise the user program has to be stored inside the controller by Burn mode. So the entire memory space of 640 Byte will be available.
Type **run** into the last line for starting a finished program. It may be stopped again by pressing the button S1 (INT). The Program mode menue **ram mode (list|run|line|ram|test|burn)** will appear again. Use the instruction **list** for showing the already stored program lines.

4.1.2 Downloading User Programs at once

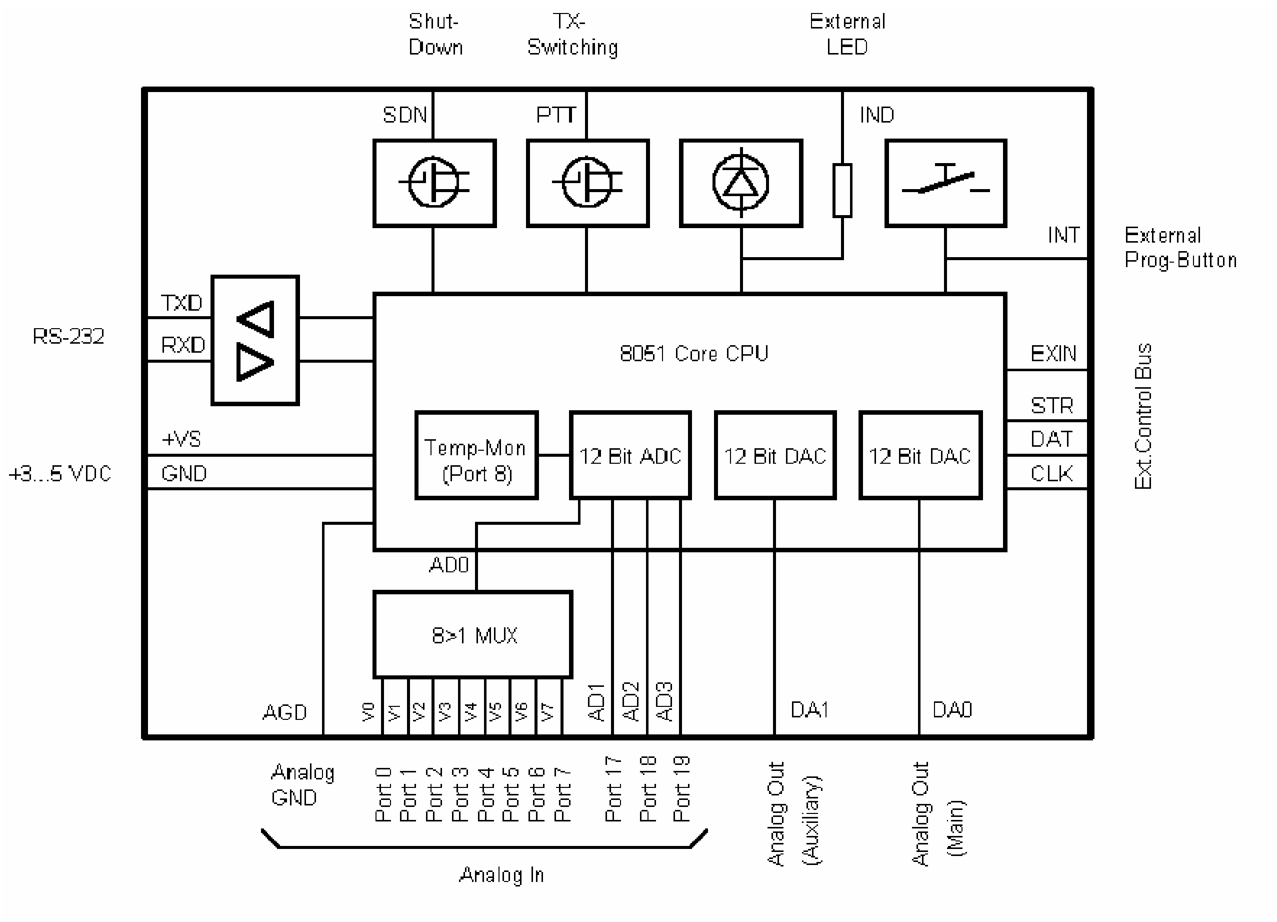
You may also write the user program by using a standard Text Editor Software and download the resulting txt-file into the UC100. It is the most comfortable way to re/program the UC100 in RAM mode. Just change the concerned lines and download the file through the Terminal program. For this kind of data transmission it is best to eliminate the echo mode by using the download instruction **ram&** instead of **ram** as the first line in the user program.

4.2 BURN – Mode

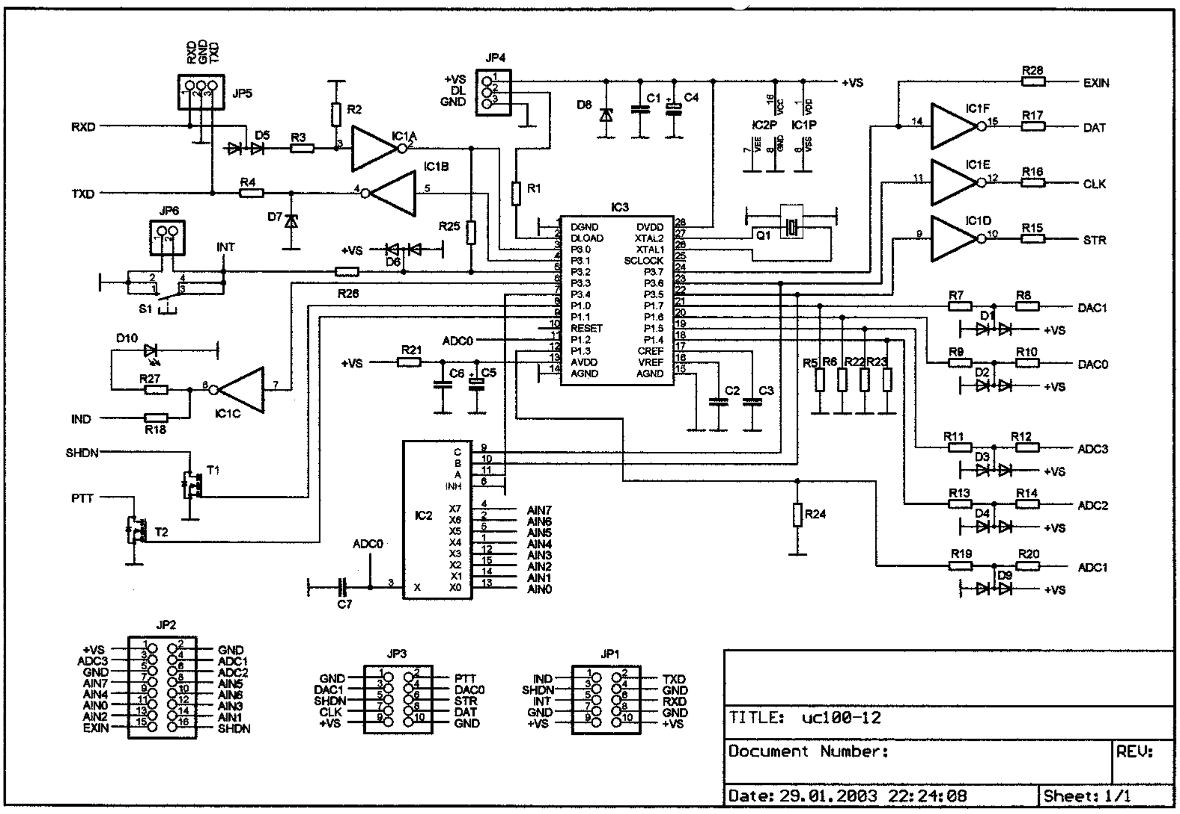
Use this mode for permanent storing of your user program inside the UC100 EEPROM. After typing **burn** as the first line all following characters will be written into the EEPROM memory. Use the instruction **end** for the last line. This kind of mode provides all available memory space for the user program and will store it permanently.
Syntax errors will result in an error message and the controller will switch into Test mode for testing the instruction code before continuing.
Entering instructions line by line is also possible in this mode but it is recommended to download the program file from a Text Editor by using **burn&** as a first line and to terminate with **end**. After that the total number of program lines will be displayed. The user program is now ready to be started by typing **run** or switching the supply voltage off and on again.

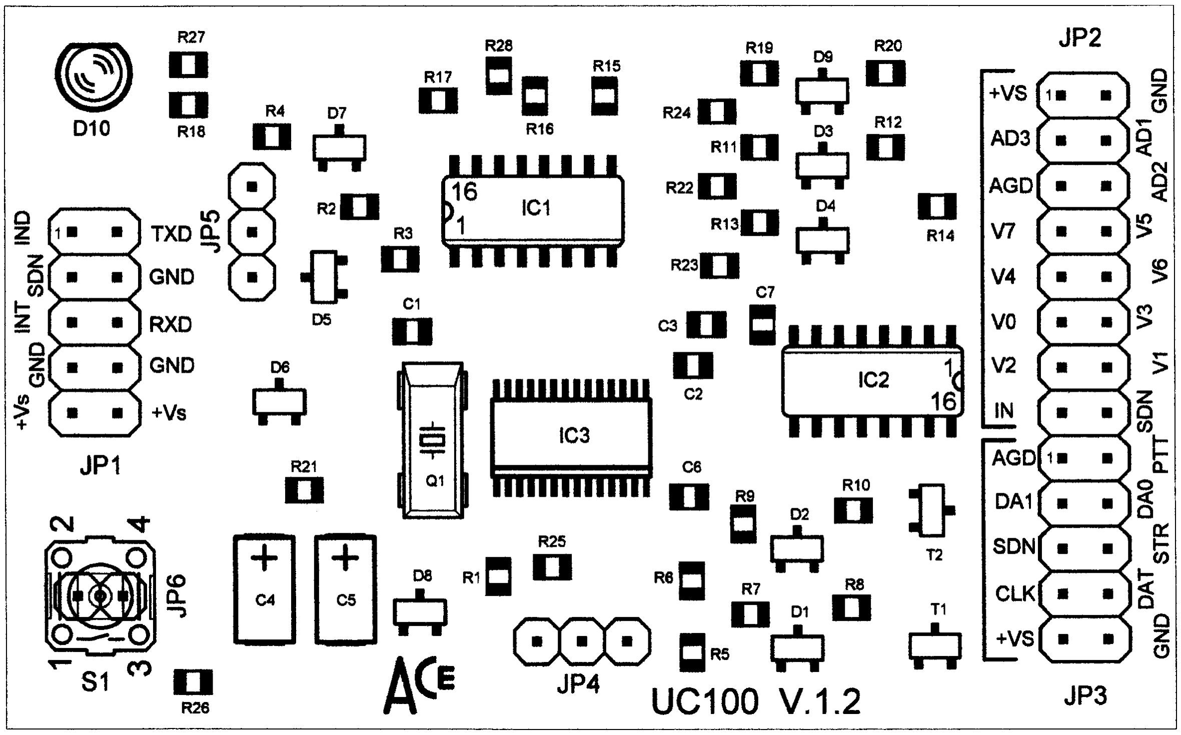
Technical Specifications

Supply Voltage	3,0 ... 5,5 VDC stabilized
Current consumption (without external loads)	abt. 10 mA
Current consumption (Sleep-Mode)	abt. 50 µA
Input Impedance of AD-Converters	abt. 1 MOhm
Maximum Vin of AD-Converters	max. +2,5 VDC
Output Impedance of DA-Converters	abt. 200 Ohm
Output Voltage of DA-Converters	max. +2,5 VDC (or +Vs, selectable)
Current handling of PTT and SHDN	max. 200 mA (against GND)
Output Impedance of TX-Bus signals	abt. 470 Ohm

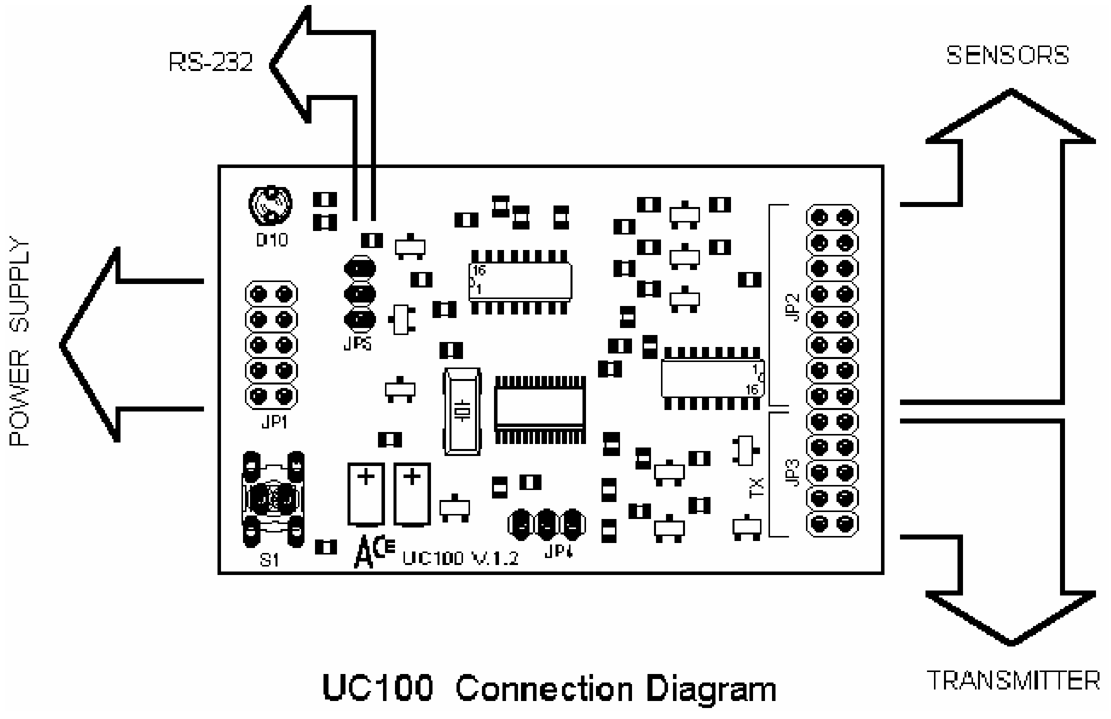


UC100 Block Diagram


JP5
RXD
GND
TXD
JP4
+VS
DL
GND
IC1A
IC1B
IC1C
IC1D
IC1E
IC1F
IC2P
IC1P
IC3
Q1
JP6
INT
S1
D10
IND
SHDN
PTT
T1
T2
IC2
ADC0
AIN7
AIN6
AIN5
AIN4
AIN3
AIN2
AIN1
AIN0
EXIN
DAT
CLK
STR
DAC1
DAC0
ADC3
ADC2
ADC1
JP2
JP3
JP1
TITLE: uc100-12
Document Number:
REV:
Date: 29.01.2003 22:24:08
Sheet: 1/1


D10
JP1
TXD
GND
RXD
GND
+Vs
IND
SDN
INT
GND
+Vs
JP5
IC1
IC3
IC2
Q1
S1
JP6
JP4
UC100 V.1.2
JP2
+VS
AD3
AGD
V7
V4
V0
V2
IN
GND
AD1
AD2
V5
V6
V3
V1
SDN
JP3
AGD
DA1
SDN
CLK
+VS
PTT
DA0
STR
DAT
GND

Pinning



UC100 Connection Diagram

JP1 (Supply Connector):

+VS:
Voltage Supply for UC100, stabilized DC between 3.0V and 5.0V.
Applying voltage generates a Reset Pulse. Protected against over voltage by a 5,6V Zener Diode (Caution ! No current limiting resistor, diode can be destroyed at high current levels)

GND:
General Supply Ground, to be connected to Minus Pin of Power Supply

SDN:
Shutdown Pin for additional hardware. Open Drain Output for reducing current consumption during „Sleep" – mode (switches to GND when unit is ON), +24VDC / 200mA maximum.

INT:
Switching signal from standard operation into Programming Mode, pushbutton against GND, in parallel to internal button S1

IND:
LED drive pin for external indication. A 2mA LED can be directly connected against GND (1k resistor on PCB)

TXD:
TX-Data for RS-232 interface, PC can be connected directly, default value of Baudrate 1200

RXD:
RX-Data for RS-232 interface, PC can be connected directly, default value of Baudrate 1200

JP2 (Sensor-Connector):

+VS:
Supply Voltage of UC100 provided for use as supply for external Sensor circuitry (direct connection to +VS at JP1)
GND:
Supply Ground for external Sensor circuitry (direct connection to GND at JP1)
AGD:
Analog GND for applied voltages to be measured (direct connection to GND at JP1)
AD1:
Input into A/D-Converter (Port 17), static Input impedance abt. 1M // 30pF, avoid Vin > +VS and < 0V
AD2:
Input into A/D-Converter (Port 18), static Input impedance abt. 1M // 30pF, avoid Vin > +VS and < 0V
AD3:
Input into A/D-Converter (Port 19), static Input impedance abt. 1M // 30pF, avoid Vin > +VS and < 0V
V0:
Input into A/D-Converter via internal Multiplexer (Port 0) , avoid Vin > +VS und < 0V
V1:
Input into A/D-Converter via internal Multiplexer (Port 1) , avoid Vin > +VS und < 0V
V2:
Input into A/D-Converter via internal Multiplexer (Port 2) , avoid Vin > +VS und < 0V
V3:
Input into A/D-Converter via internal Multiplexer (Port 3) , avoid Vin > +VS und < 0V
V4:
Input into A/D-Converter via internal Multiplexer (Port 4) , avoid Vin > +VS und < 0V
V5:
Input into A/D-Converter via internal Multiplexer (Port 5) , avoid Vin > +VS und < 0V
V6:
Input into A/D-Converter via internal Multiplexer (Port 6) , avoid Vin > +VS und < 0V
V7:
Input into A/D-Converter via internal Multiplexer (Port 7) , avoid Vin > +VS und < 0V
SDN:
Shutdown Pin for additional circuitry. Open drain output for reducing current consumption during „Sleep" Phase
(switches to GND), +24VDC / 200mA maximum
IN:
Digital input signal for specific applications (not in Vers.1.00)

JP3 (TX-Connector):

DA0:
Analog output for TX modulation (main signal), output impedance 200 Ohm, Vmax = Vref (2,5V) or +VS (selectable)
DA1:
Analog output for TX modulation (auxiliary signal), output impedance 200 Ohm, Vmax = Vref (2,5V) or +VS (selectable)
AGD:
Analog Ground for TX modulation (direct connection to GND at JP1)
+VS:
Supply Voltage of UC100 provided for use as supply for a low power transmitter unit (direct connection to +VS at JP1)
GND:
Supply Ground for a low power transmitter unit (direct connection to GND at JP1)
PTT:
TX-ON Signal (Open – Drain), +24VDC / 200mA maximum

SDN:
Shutdown Pin for additional circuitry. Open drain output for reducing current consumption during „Sleep" Phase (switches to GND) or user defined, +24VDC / 200mA maximum

DAT:
Data – Signal of the serial TX – Control Bus (may also be used as a standard output pin)

CLK:
Clock - Signal of the serial TX – Control Bus (may also be used as a standard output pin)

STR:
Strobe - Signal of the serial TX – Control Bus (may also be used as a standard output pin)

JP4 (Jumper for Prog-Download):

OP:
Place Jumper at OP for normal operation

DL:
Jumper at DL for updating the Firmware

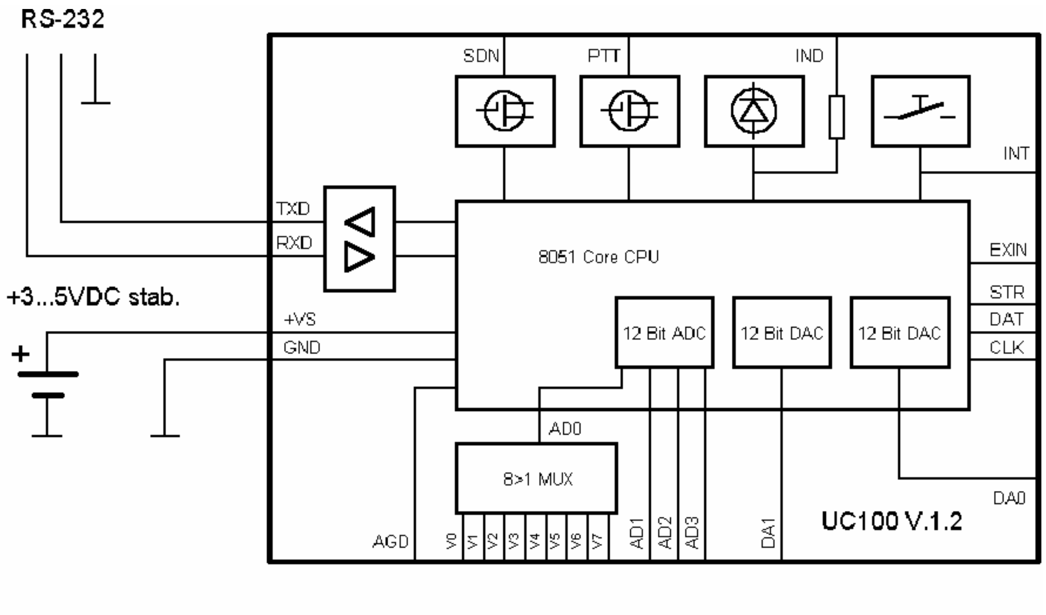
JP5 (additional RS-232 connector):

TXD:
TX-Data for RS-232 interface, PC can be connected directly, default value of Baudrate 1200

RXD:
RX-Data for RS-232 interface, PC can be connected directly, default value of Baudrate 1200

Connection Examples

1. Connecting the PC

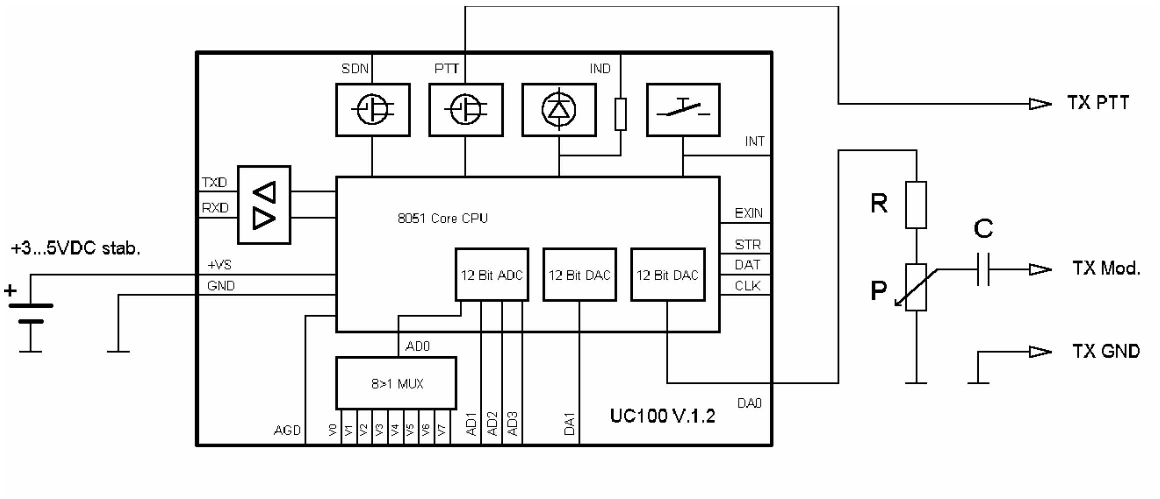


UC100 Connecting the PC

For programming and operating the UC100 on the PC connect the Serial Interface to the TXD, RXD und GND Pins on the UC100 board. Try to keep the cable as short as possible. Further Informations are to be found in the chapter „Starting" of this manual.

When using longer supply lines for the board it is recommended to place some LC T or Π shaped filter close to the UC100 supply connector in order to keep emission of unwanted controller generated signals low.

2. Connecting a Transmitter (FM or AFSK)



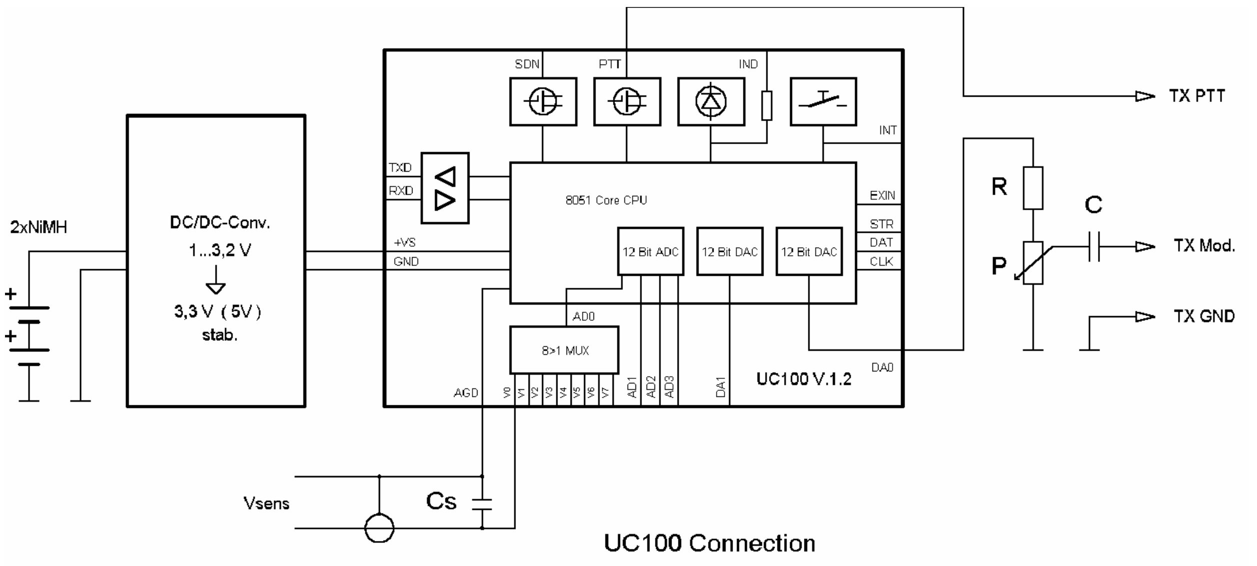
UC100 Connecting a Transmitter (FM or AFSK)

Almost every Transmitter or Transceiver can be controlled by the UC100 using no or just a simple interface circuit. A MOS FET is used for PTT by switching this signal down to GND (200mA max.)

As the UC100 is offering the feature to adapt the modulation level by software, transmitters can be directly controlled by the D/A-Converter. A more versatile way is shown in the picture above. So it is possible to match also very sensitive inputs at full audio bandwidth as well as DC decoupling between UC100 and transmitter (Caution! Not suitable for DC controlled modulation types like FSK).
Keep the resulting resistance of R+P between 500 Ohms and 10 kOhms. The coupling capacitor C has to be matched to the input impedance of the modulator. In practice values between 100 nF and 1 µF should work fine.

At connection lines between UC100 and transmitter longer than abt. 5cm it is recommended to use shielded cables.

3. Connecting a Transmitter (FM/AFSK) and Sensor, Battery Supply



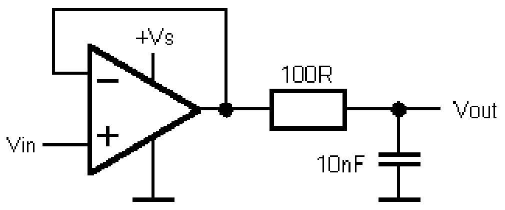
UC100 Connection

Battery Operation, TX-Mod. FM or AFSK, 1 ext. Sensor

Try to avoid supplying the UC100 directly from batteries. Best accuracy is achieved at supply voltages within +-5 % of a value within the specified range (3...5 VDC). So a DC/DC-Converter with stabilized output voltage should be used at battery operation.

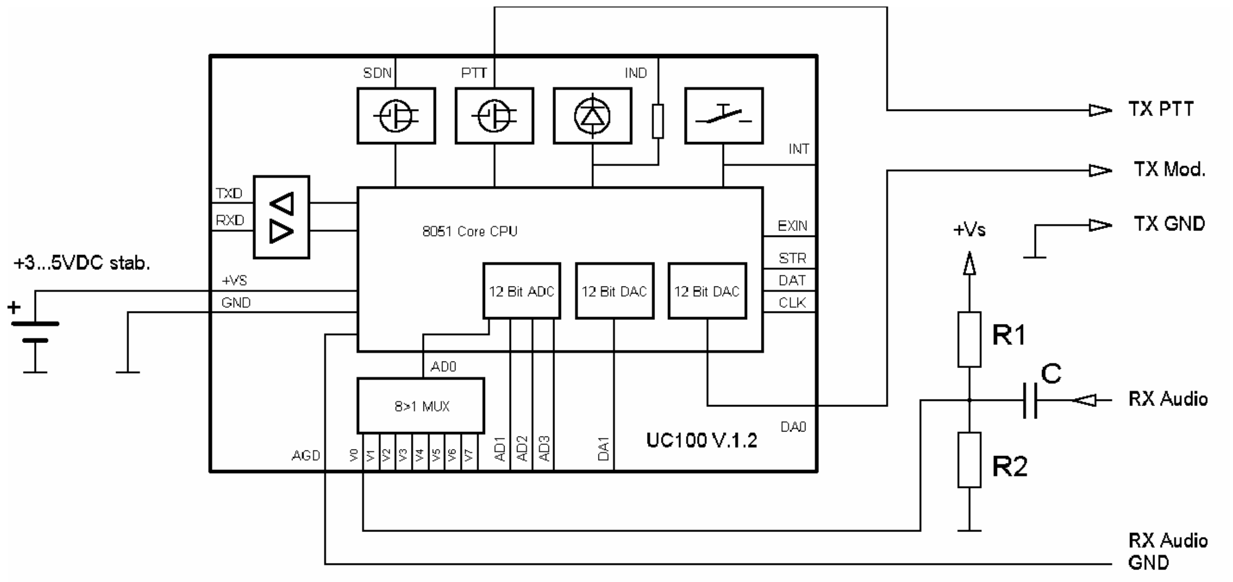
When connecting sensors you have to observe 2 important things to get the best results. The sensor´s output impedance should be as low as possible and longer connection lines have to be shielded.
It is recommended to place a buffer amplifier (rail to rail OP Amp) between sensor output and A/D-Converter input (see below)
For getting stable measuring values a capacitor (Cs) may be placed right at the input pin of the UC100 board. It´s value has to be matched to the sensor´s output impedance as well as the maximum signal frequency. If too low Cs will have no significant effect on the stabilization, very large values are reducing the maximum measuring frequency.

As far as the modulation level is concerned informations as in chapter 2. can be applied.



Proposal for Buffer Amplifier

4. Connecting a Transmitter (FSK) and Audio Detection

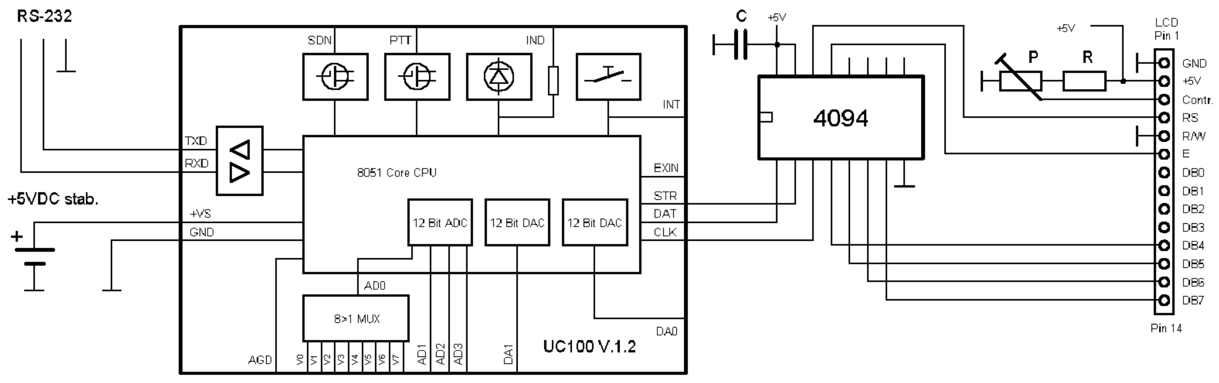


UC100 Connection to a Transmitter (FSK) and Receiver

Using for example the mode **h96** (FSK, 9600 Bd.) the modulation signal is available at DA0 as a voltage between 0 and 2,5VDC. That square wave signal can directly be used to control the modulator.

The UC100 may also be controlled by a tone burst (instruction **wait <frequ>**). In order to decode it´s frequency the audio signal has to be connected to one of the input ports (picture above: port 0). Best level handling performance can be achieved by placing a voltage divider right at the A/D-C input providing a DC level of about 1 to 1,5V. So audio input levels in the range of 30 to 500 mV RMS will be detected without problems.
The overall serial resistance of the divider R1/R2 should be fixed within a range of 10 kOhm to 500 kOhm. But try to keep the calculated parallel resistance of R1//R2 as large as possible in order to reduce the load for the receiver audio output stage. R1//R2 should never get below 1 kOhm. So also the coupling capacitor (C) can be kept at lower values. At R1//R2 values around 10kOhm capacitor C may have 100nF, between 1kOhm and 10kOhm C may be fixed to 1µF.
Large audio input levels (above 800mV RMS) will be limited by the UC100 protection circuitry and result in distorted input signals. That may lead to possible decoding failure.

5. Driving an LC Display



UC100 Connecting an LCD

It is possible to drive standard LCDs by connecting an additional shift register (4094). Using the **lcd** instruction will activate the UC100´s external serial bus to write data via the shift register to the display.
Preferred LCDs are dual line types ranging from 16 to 40 characters.

For this application any CMOS 4094 circuit may be used, no special performance will be needed. Bypass capacitor C (100nF) should be implemented for correct operation. In order to control the display contrast R (2k7) and P (1k) have to be connected to LCD Pin 3.

The external shift register can also be controlled directly for different purposes by using the command **shft** (see Instruction Set).

Instruction Set

System Configuration

cfg This instruction may be placed at the start of the user program for changing certain default values. It is not needed if all default settings are applicable.

Default values:

- Signal SDN is on at Sleep Mode
- LED follows PTT
- Access to program mode also by terminal instruction
- Maximum output voltage of DACs = Vref (2,5V)

SYNTAX: **cfg 0000000000b** (10 Bits, 5 of them used, remaining 5 Bits = 0)

Bit 0 (**000000000Xb**) :	0: SDN Signal in Sleep Mode	1: SDN determined by user program
Bit 1 (**00000000X0b**) :	0: LED on with PTT on	1: LED determined by user program
Bit 2 (**0000000X00b**) :	0: Prog.-Mode also by terminal	1: Prog.-Mode only by switch S1
Bit 3…7 (**00XXXXX000b**) :	not used, set to 0	
Bit 8 (**0X00000000b**) :	0: DAC0 max. up to Vref (2,5V)	1: DAC0 max. up to +Vs
Bit 9 (**X000000000b**) :	0: DAC1 max. up to Vref (2,5V)	1: DAC1 max. up to +Vs

Read/Write Variables

a … **j**	Universal 16 bit signed variable SYNTAX : **a=<x>** or connection of 2 other 16 bit variables or constants (e.g. **a=b+c** or **e=e+1**)
p0, **p1**, **p2**, **p3**	Port address as 8051 – Standard, here p1 and p3 accessable SYNTAX: just like with universal variables **Caution:** Changing port status may result in undefined state or data exchange problems. Should be used only by persons owning fundamental knowledge about 8051 core controllers.
x, **xl**	String variable, length of string variable, also see instructions **rdln, rs** and **pr**, read - variable **xm** and Program Examples respectively
lev	Changing modulation amplitude SYNTAX: **lev=0** to **lev=255** for 0…2,5V
spd	Changing speed at CW or duration of beep - signal (see also **cwa1, cwf1, cwf2** and **beep**) , SYNTAX: **spd=0** to **spd=255**
dc	DC level on modulation signal SYNTAX: **dc=0** to **dc=255** for 0..2,5V
f0	Determines an audio frequency for output at DA0, stepwidth=0,2Hz, e.g. **f0=5000** for an audio frequency of 1 kHz, use instruction **tone** for output(not for AFSK)
f1	Defines second frequency for AFSK
txd	TX – Delay for the modes PR (AX-25), PSK and DTMF, but different units for each mode SYNTAX: **txd=0** to **txd=255**
tail	Defines the tail – time at PR (AX-25) SYNTAX: **tail=0** to **tail=255**
devi	Deviation at analogue FSK (starting at 0 Volts up) SYNTAX: **devi=0** to **devi=255**

port	Selects an analog port, 0…7 = ADC0 via Multiplexer, 8=Chip-Temp-Sensor, 17…19=ADC1..3 direct. SYNTAX: **port=0** to **port=8** and **port=17** to **port=19** respectively
rs	Last data Byte on serial interface. Can also be used to transmit Bytes
th0	Sampling rate for modulation, changes all speeds and frequencies
shft	Sends data to an external shift register via the serial bus (DAT, CLK, STR) SYNTAX: **shft=0** to **shft=255** or **shft=00000000b** to **shft=11111111b** How to connect the shift register see Connection Examples, 5. Driving an LC Display

Read – Variables

adc	Voltage level at the selected port. Value (decimal) is between 0 and 4095 for voltages from 0 to 2,5V DC.
temp	Chip temperature at 3 digits with 0.1 Deg C resolution, without decimal point (the internally calculated value on port 8) so the decimal point has to be set for correct output (e.g. **pr %d31:temp** puts out the chip temperature via the serial interface in the following way: **21.3**). Tolerance range abt. +- 1,5°C, depending on current consumption of the CPU.
xm	Maximum Size of the String Variable (Size of FIFO Buffer)

Write – Variables

dac0	Puts out a DC voltage between 0 and 2,5V on DA0 SYNTAX: **dac0=0** to **dac0=4095** for 0 to 2,5V
dac1	Puts out a DC voltage between 0 and 2,5V on DA1 SYNTAX: **dac1=0** to **dac1=4095** for 0 to 2,5V

Procedures

pr	Puts out a data line via the RS-232 interface SYNTAX for fixed data: **pr <x>** (x=information to be sent) SYNTAX for variables: **pr %d<y>:<x>** (d=dec, y=output format, x=variable to be sent) Output formats : Always starting with the formatting sign **%** Followed by **d** (decimal), **h** (hex), **$** (string variable x) or **n** (new line) Decimal output **d** requires definition of used digits and if necessary the position of the decimal point e.g. **%d31:temp** writes the chip temperature at 3 digits and sets comma to 0,1 Deg (xx,x) at **%d51:temp** indication is the same, but 2 additional blanks at the beginning **%<constant>** and **%<variable>** used for sending ASCII Bytes
rdln 0	Starts non blocking read from RS-232 interface (FIFO), no EOL, stopped by **rdln -1**
rdln 1	Starts read from RS-232 interface into String **x** with length **xl**, terminated by CR (13)
rdln -1	Stops non blocking read, started with **rdln 0**

rdln 2 .. 65535	Like **rdln 1** but terminated by CR (13) or timeout in ms Caution: EOL (0Dh) also stored in string. Delete by **xl=xl-1** In case of timeout a String **xl=0** will be returned
rsbd	Sets the RS-232 Baud rate. Default value is 1200Bd SYNTAX: **rsbd=<x>** (x=Baud rate)
cwa1	Transmits fixed texts or variables in Morse code (Keying of PTT – Signal). Speed is about 60 lpm, can be changed by instruction **spd**. (**spd=70** for abt. 45 lpm, 60 for abt. 55 lpm, 50 for abt 65 lpm, 40 for abt. 80 lpm, 30 for abt. 105 lpm) SYNTAX for fixed text: **cwa1 <x>** (x may contain letters, numbers or numerous special signs) SYNTAX for variables: **cwa1 %d<y>:<x>** (d=dec, y=output format, x=variable or port) See more detailed informations concerning output format at instruction **pr,** also check instructions **spd** and **init**
cwf1	Transmits fixed texts or variables in Morse code (keying of the DC – voltage on DA1). Speed is abt. 60 lpm, can be changed by instruction **spd**. For other features see **cwa1**.
cwf2	Transmits fixed texts or variables in Morse code (audio keying with frequency **f0**). Speed is abt. 60 lpm, can be changed by instruction **spd**. For other features see **cwa1**.
h12	Transmits fixed text or variables in AX-25 protocol (Packet Radio), AFSK, 1200 Bps SYNTAX for fixed text: **h12 :<X> <Y>:<z>** (X=RX-Callsign, Y=TX-Callsign, z=any text) SYNTAX for variables: **h12 :<X> <Y>:%d<z>:<a>** (X,Y as above, d=dec, z=output format, a=variable) Example : Output of chip temperature: **h12 :RXCALL TXCALL:%d31:temp** Up to 8 digipeater calls can be used, e.g. **h12 :RXCALL TXCALL DIGI1 DIGI2:test** (use capital letters for the callsigns!).
h24	Transmits fixed text or variables in AX-25 protocol, AFSK, 2400 Bps, SYNTAX like **h12**
h96	Transmits fixed text or variables in AX-25 protocol, FSK, 9600 Bps, SYNTAX like **h12**
dtmf	Transmits fixed numbers or variable numbers in DTMF coded tones SYNTAX for fixed numbers: **dtmf <x>** (x = any group of numbers) SYNTAX for variables: **dtmf %<x>** (x = defined variable, like instruction **pr**)
psk	Transmits fixed texts or variables in PSK31 SYNTAX for Fixtext: **psk <x>** (x =any text or numbers) SYNTAX for variables: **psk %<x>** (x = defined variable, like instruction **pr**)
beep	Transmits a tone burst SYNTAX: **beep <x>** (x=tone frequency in Hz), setting the duration via **spd** in 10ms steps, e.g. **init beep** followed by **spd=50** for 500ms, then for example **beep 1000**
tone	Transmits constantly frequency **f0**, which may also be defined as a variable. Works like a V/f - Converter SYNTAX: **tone** (written in a single line without extension, then a Label and a loop for reading the desired variable. See Program Examples). **tx 0** terminates the output. When using the instructions **sls** or **slms** together with **tone** the frequency that was sent before switching into idle mode will be kept
f3	Measured voltage level at a selected port will be 1:1 transferred to DA0
init	Sets default values (Parameter) for the selected modulation type until another modulation is called up SYNTAX: **init <x>** (e.g. **init cwa1**)
sls	Adds a waiting period to the program, the UC100 switches into „Sleep" – Mode SYNTAX: **sls <x>** (x = waiting time in seconds, only full seconds possible, e.g. **sls 2**)

slms Like instruction **sls** , but time input in milliseconds

tx 1 Turns the transmitter on (PTT)

tx 0 Turns the transmitter off (PTT)

led 1 Turns the LED on (see also instruction **cfg**)

led 0 Turns the LED off (see also instruction **cfg**)

shdn 1 Switches the SDN Pin through (see also instruction **cfg**)

shdn 0 Switches the SDN Pin off (see also instruction **cfg**)

wait Stops the program, will be continued when a control signal appears on the selected port. Starting by leading or falling edge can be selected individually (Threshold 1,25V).
SYNTAX for start at leading edge **wait <x>** (x = Port address 0...7, 17..19, e.g. **wait 0**)
SYNTAX for falling edge **wait <-x>** (x = Port address 0...7, 17..19, e.g. **wait -1**, not possible on port 0, there only leading edge)
Start by tone burst: **wait <frequ>** (frequ = audio frequency to be detected [Hz]), the program stops until the desired audio frequency appears on the selected port and is continued after the burst.

lcd Writes fixed texts or variables to Standard-LCDs
For this feature a simple hardware interface will be necessary to convert serial data from the UC100 into parallel control signals for the LCD (see chapter "Connection examples").

Basically most LCDs can be controlled by the available commands, but preferred types are dual line LCDs with 16 to 40 characters each.

SYNTAX: **lcd** <**string**> for writing to first position in first line (e.g. **lcd Hallo**)
lcd %128+x <**string**> for writing to a particular position in the first line
e.g. **lcd %130 Hallo** writes „Hallo" to the 3rd position of the first line
lcd %192+x <**string**> for writing to a particular position in the 2nd line
SYNTAX for variables: e.g. temperature in line 2, row 3 : **lcd %194 %d31:temp C**

Control instructions: Control nibbles (Dec. 0…15) can be sent to the connected display
Example: **lcd %0 %1** for clearing the screen
Between some **lcd** commands it may be necessary to add a short pause of a few milliseconds for correct operation (**slms**)
Initialization: Initialization string is being sent automatically with first call of **lcd** instruction after switch on.

app Connects the following string **x** at position **xl** (**xl** will be incremented by the string length)

Adds a comment line, does not count as an instruction, SYNTAX: **#<text>** (no blank)
Needs no memory space inside the chip.

Mathematical / Logical 2 Parameter Operands

+	Adds 2 constants or variables (16 Bit), e.g. **a=b+1**
-	Subtracts 2 constants or variables (16 Bit), e.g. **a=b-1**
*****	Multiplies 2 constants or variables (16 Bits each, result limited to 16 Bit), e.g. **a=adc*2**
/	Divides 2 constants or variables (16 Bit), e.g. **b=a/3**
~	Multiplies by a fractional factor, used for simplified input of following operations: **a=b*~c** is the same as **a=b*(c/65536)** and **a=b~/c** ist he same as **a=(b*65536)/c** for positive numbers and b<c
#	Modulo, e.g. **c=a#b** (16 Bit)
&	Logic AND, e.g. **a=b&c** (16 Bit)
|	Logic OR, e.g. **a=b|c** (16 Bit)
%	Logic XOR, e.g. **a=b%c** (16 Bit)
<	Criteria LESS THAN, e.g. **if a<b** (16 Bit)
>	Criteria GREATER THAN, e.g. **if a>b** (16 Bit)
=	Criteria EQUAL TO, e.g. **if a=b** (16 Bit)
<>	Criteria NOT EQUAL TO, e.g. **if a<>b** (16 Bit)
>=	Criteria GREATER THAN OR EQUAL TO, e.g. **if a>=b** (16 Bit)
<=	Criteria LESS THAN OR EQUAL TO, e.g. **if a<=b** (16 Bit)
^0	Bit 0 of right variable, Bits 1..15 of left variable
^1	Bit 1 of right variable, Bits 0, 2..15 of left variable
^2	Bit 2 of right variable, Bits 0..1, 3..15 of left variable
^3	Bit 3 of right variable, Bits 0..2, 4..15 of left variable
^4	Bit 4 of right variable, Bits 0..3, 5..15 of left variable
^5	Bit 5 of right variable, Bits 0..4, 6..15 of left variable
^6	Bit 6 of right variable, Bits 0..5, 7..15 of left variable
^7	Bit 7 of right variable, Bits 0..6, 8..15 of left variable
+^	Sets Bit n of left variable (e.g. **p3=p3+^a**)
-^	Erases Bit n of left variables (e.g. **p3=p3-^a**)

Program Instructions

if	If following expression false, skip next line (e.g. **if a=10**)
goto	Jump to Label No. 1…255 (e.g. **goto 10**)
loop	Back to first line of program
halt	Stops the running program, UC100 switches into „Sleep-Mode"
end	Last line of the user program in BURN - Mode
<label>**:**	Defines the begin of a certain program block for jumping to (instruction **goto**) SYNTAX **<x>:** INFO: x = any number between 1 and 255 (e.g. **10:**) Label needs a complete instruction line, put next instruction into new line

Interactive Instructions

list	Lists all instruction lines from RAM on the screen
run	Starts the user program in RAM
line	Input of an instruction line and immediate execution
test	Syntax-Test of one instruction line, Echo off for keyboard input
test&	Syntax Test of one instruction line, Echo on for input from data file
burn	Following instruction lines are stored in the non volatile memory (EEPROM), Echo on for keyboard input. Has to be terminated by the instruction **end**.
burn&	Following instruction lines are stored in the non volatile memory (EEPROM), Echo off for input from data file. Has to be terminated by the instruction **end**.
ram	Following instruction lines are stored into the RAM. Memory space is about 1/3 of the EEPROM. Used for quick test of smaller user programs. Program will be lost when +Vs is switched off. Echo on for keyboard input.
ram&	Following instruction lines are stored into the RAM. Memory space is about 1/3 of the EEPROM. Used for quick test of smaller user programs. Program will be lost when +Vs is switched off. Echo off for input from data file.