# *MicroNator*

# OVERVIEW

## UNIVERSAL
## DEVELOPMENT BOARD
**Version 4.04**

## Overview
**Version 4.04a**

***RF-232***

**ISBN 2-9803460-2-0**

**Dépôt Légal -** Bibliothèque Nationale du Québec, avril 1995.

***PRINTED IN CANADA***

# *MicroNator*

## OVERVIEW

## UNIVERSAL
## DEVELOPMENT BOARD
**Version 4.04**

## Overview
**Version 4.04a**

***RF-232***

# OVERVIEW

**MicroNator UNIVERSAL DEVELOPMENT BOARD**



**RF-232**
**1404 rue Galt**
**Montréal, Qc H4E 1H9**
**CANADA**
**Tél: (514) 761-4201**

**RF-232**
**21 rue André Gide**
**59123 ZUYDCOOTE**
**FRANCE**
**Tél: 03 28 58 28 39**

**micronator@micronator.com**



**ISBN 2-9803460-2-0**

**Dépôt Légal -** Bibliothèque Nationale du Québec, avril 1995

# MicroNator Single-Board Computer System

## for the 68HC11 by: Michel-André Robillard MCNE, PSS

In the following overview I will try to explain the underlying idea & reasoning for the **MicroNator** Single-Board Computer Concept and debate the choices I made in choosing this implementation instead of those many alternatives available on the market today.

### CONVENTION

Acronyms are omnipresent and the first time you encounter a new one it distracts your reading concentration. Refer to end of this overview for definitions.

Because a system may contain many power sources I always show the power rail by: VCC for 5 Volts, V3 for 3.3 Volts, and VBAT for the backup battery supply on every part to indicate by which rail it is powered. Decoupling capacitors are also always indicated if the chip requires one.

The * after the name of a signal is to show that this signal is asserted or active when it is LOW, 0 Volt, or logic level 0. HIGH refers to 5 Volts or logic level 1.

PC refers to the computer connected to the system serial port J2.

MCU or CPU both refer to the HC11 chip.

### THE "BEST" ON THE MARKET

The most outstanding of all the properties of the IBM-PC is not its CPU, memory, disks, nor its I/Os but its many standards. That is what made it so popular and accepted worldwide.

Today the most interesting standard emerging in the world of control is the PC/104™, mainly its connector concept which eliminates the motherboard itself. The only drawback of this standard, for a small volume designer, is the form factor that is 3.6" x 3.8" and almost requires SMT or a multi-layer board for dense through-hole components.

The easiest of all the 8 bits microcontrollers to use today is the 68HC11 family and its many derivative members.

For the ordinary designer that doesn't have a universal eprom-programmer and an eprom-eraser the best memory is the EEPROM.

So why don't we put all the best together?

### PCB FORM FACTOR

A good form factor for the PCB, which is large enough to hold quite a few through-hole components and still leave some area for Wire-Wrapped, easy and inexpensive to find a casing to fit into, and a standard in the instrumentation world, is 4.75" x 5.725". The four holes with the

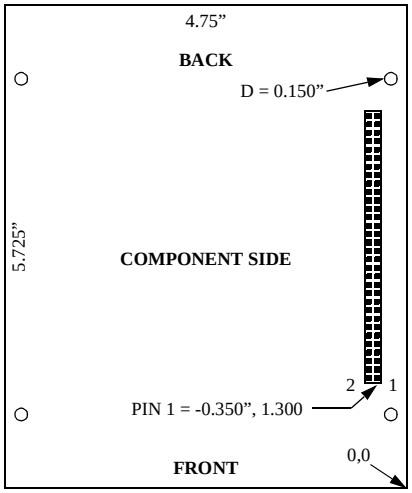



Fig: 1 Standard **MicroNator**

0.150" diameter are for stacking purposes only and are explained below. They are centered and separated by 4.38" horizontally and by 4.00" vertically.

### CASING

The ideal casing first must be good-looking, professional, and it should be:

- easily available in almost every electronic store,
- inexpensive,
- simple to assemble,
- available in more than one color,
- be offered in more than one height so as to be able to put one or more boards in it,
- easy to work with, i.e. not difficult to cut holes into it.

The ideal casings are the PACTEC CM series. They have a very attractive appearance, are stocked by distributors worldwide, cost from $10.00 to $20.00 US, have only two screws to hold them together, include all parts and hardware necessary for assembly, come in black and bone colors, and have openings that are easily cut into. The CM6-225 can hold from 1 to 4 boards and the CM6-300 1 to 5.

### PCB INTERCONNECTON

The most interesting connector in the industry nowadays is the one used by the PC/104™ consortium to distribute their two buses. One has 64

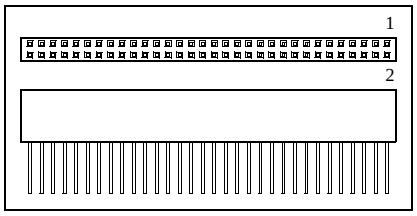



Fig: 2 Standard 64-pins bus connector

pins and the other 40. They look like wire-wrapped female headers and each pair plugs into the next one.

The pins are 0.025" square and the current rating for the gold-plated is 1 Amp from -65°C to +125°C. There are two kinds of pins: one looks wire-wrapped (0.480") and the other is of standard length (0.090") for PCB soldering. You can use the 0.090"

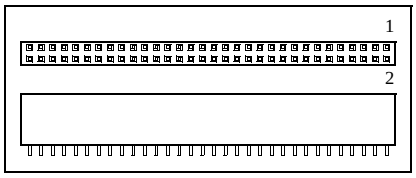



Fig: 3 Connector with 0.090" pins

connector for your bottom board as it will shorten the overall thickness of your system.

By using this kind of connector one eliminates the use of mother board which is quite a saving in money, and it resolves the question of space reservation and the number of bus connectors to install on the mother board. With those connectors, if you need another board, you just plug the new PCB into the ones already in place. You can insert the new board on the top, in the middle, or on the bottom of the original stack. The space between the two PCB excluding their thickness is exactly 0.625" so as

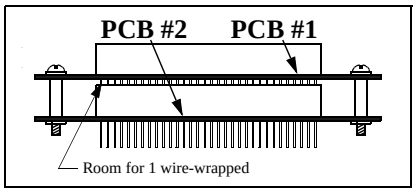



Fig: 4 System with two boards

to be able to use standard spacers to separate and hold them together.

There is still enough room 0.190", refer to Fig: 4, between the pins of the top board and the next bottom connector to have one standard wire-wrapped connection from each pin of the bus to the added chips in the user wire-wrapped area on the CPU circuit board. The same holds true for the wire-wrapped expansion board or any other member in the stack.

All the signals of the CPU, plus some extra ones such as VCC, Gnd ..., are on the bus connector. With your boards you can form a toast (one board), a "sandwich" (two boards), a "club sandwich" (three boards), or a "custom sandwich" of any number of layers you like...

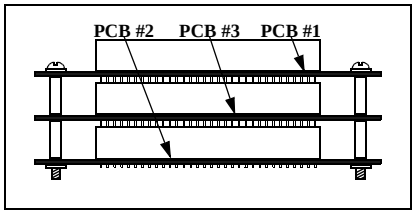



Fig: 5 System with three boards

## POWER SUPPLY

The general purpose wall-mounted power supplies are cheap, rugged, and available from any electronic or surplus store. They are adapted for 110VAC/60cycles or 220VAC/50 cycles.

## POWER CONNECTOR

Connector P1 is the input for the power supply. It is male, has two pins,

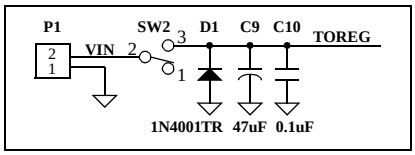



Fig: 6 Input Power "VIN"

is 5 mm, has only one way mating, and is from WECO. Male so as not to have uncovered pins coming from the wall-mounted power supply, two pins because there is only one way to mate it, 5 mm so it is quite small, and from WECO meaning availability. The rating is 380 Volts and 6 Amps. The female will accept 22-14 AWG wires.

VIN is also on the bus connector to pin 63. The main advantage of this is to use an expansion board and pack it with batteries to power a hand held instrument.

Diode D1 is used in case of plugging the power supply backward (unlikely). Capacitors C9 and C10 filter the input power to the regulator for more security.

## NOTES ABOUT VOLTAGE REGULATOR

As you can notice, on the figure of the regulator, the circuit is quite simple and uses a standard 7805 regulator chip. The only limitation of this family of regulators is its relatively high voltage drop. This drop can be as high as 2 volts @ 1 Amp. To compensate for the voltage drop I recommend a wall-mounted power supply of 7.5 Volts instead of 6.5 which might not supply enough voltage around its peek current. You can also use a 9-Volt battery for the input power. The regulator will have to work just a bit harder and will generate a little more heat. Another way around, for a few extra cents, is to use a 7805 "*clone*" which has less voltage drop.

Usually the 7805 can take up to 35 volts, but this is really an extreme. Try to avoid this situation.

## VOLTAGE REGULATOR

Diode D2 is to protect the regulator and the circuit itself by bleeding the current surge if the Vcc from the bus connector becomes too high.

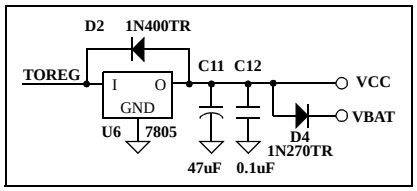



Fig: 7 VCC Regulator & VBAT

Capacitors C11 and C12 filter the output of the regulator. Diode D4 is a low drop germanium diode for supplying power to the battery circuit.

This Vcc is the main power rail for all the logic in the circuit, including the expansion boards if there are some in the system. Vcc is also connected to the bus connector pin 31 and Gnd to pin 59 if a user wants to power those pins instead of connector P1 and the regulator.

## RESET & POWER INDICATOR

The reset signal is made of the switch SW1 (momentarily closed), capacitor C8 and one of the units of the resistor network RN2. The network is a SIP which incorporates 7 resistors and they all have pin #1 in common so the package has 8 pins.

The time constant is roughly half a millisecond (4.7E3 x 0.1E-6) which gives enough time to the CPU to reset and initialize itself.

The first time the power is ap-

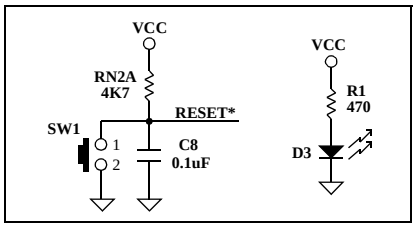


Fig: 8 RESET & Power Indicator

plied to the circuit the reset signal will be at zero volts since there is no energy stored in the capacitor C8. The current will flow from VCC through RN2A and C8 will start to charge, bringing up the RESET* signal near 5 Volts.

LED D3 is a super bright red LED and takes less current than the standard one. The 10 mA current through D3 is limited by R1. You can change it to have a more or less brighter LED.

## SYSTEM CLOCK

The system clock is taken from a crystal whose frequency is 4.9152 MHz. I choose that frequency because it is quite easy to work with since it is a multiple of 60 and when calculating time the software is straight forward.

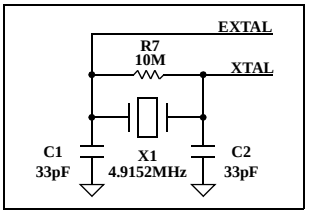


Fig: 9 CPU Crystal

It is popular and every electronic store carries it. Another reason is that it will require slower speed memory and again the cost is kept lower.

## CPU

In the family of HC11 which is the best device to choose? As far as versatility, I/O, analog input, pin-out, package, and ***availability*** the best choice is the A0, A1, E0, or E1 in a PLCC. The 'F1' is too expensive. The E9, K, and L with ROM... well let's say 'in-circuit programmable' is beautiful. The 'D' has no analog. The 'A' looks nice and the A1 is better than the A0 simply because of its availability and their cost are almost identical compared with the lowest priced, the 'D'. For the chip packaging the best again is the PLCC, the reason being the number of A/D. The PLCC has 8 of them and the DIP only 4, so I use the PLCC with a socket.

Most of the HC11s have some internal EEPROM. The A1, A8, E1 and E9 have 512 bytes of it but **MicroNator** system disables the internal EEPROM since it has external 32 KBytes and we don't want address conflict.

Because we use a socket the overall price difference compared with the 'D' is only about 50 cents, and you have 8 A/D. So again the best choice is: A0, A1, E0 or E1.

Toshiba manufactures the HC11 for Motorola and they have the right to second-source. In 1994 they publicized the HC11 family in every electronic magazines. The chips are exactly the same except for the number. Motorola starts with MC and uses 'F' for PLCC, Toshiba with TMP and 'T' for PLCC. You can use: MC68HC11A8FN, MC68HC11A1FN, MC68HC11A0FN, MC68HC11E1FN, MC68HC11E0FN by Motorola or: TMP68HC11A8T, TMP68HC11A1T, TMP68HC11A0T, TMP68HC11E1T, and the TMP68HC11E0T by Toshiba. They are all 52 pins PLCC with the same pin-out. You can also use the 48-pin version with four A/D instead of eight.

## PROGRAM MEMORY

As said earlier, EEPROM is the best memory for the ordinary designer who doesn't have eprom-eraser nor eprom-programmer and who prefer "***in-circuit programming***". Over all other kind of non-volatile memory, the greatest advantage of the EEPROM technology is that it is "***in-circuit programmable***" ***using only standard 5 volts DC***.

That is why the non-volatile memory for **MicroNator** will be a standard 32-KByte EEPROM. Two years ago such a device was expensive, but now they are around $12.00 to $15.00 US per unit. Many compa-

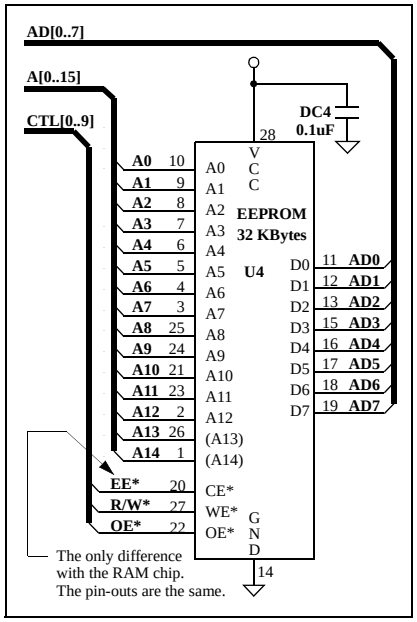


Fig: 10 EEPROM 32-KByte

nies offer them and they guarantee that they can be reprogrammed at least 100,000 times but usually they will go as high as 400,000.

The EEPROMs are better than FLASH because the technology is quite mature, well mastered, and requires only 5 volts DC for the programming voltage. 5 volts-only FLASH are now starting to be available but they are still too expensive but maybe in the future...

The only disadvantage of EEPROM is that it goes tri-stated for 5 to 10 msec right after been programmed, but ***there is a way around*** as we will see later. XICOR, ATMEL, CATALYST, SEEQ etc... manufacture them so they are easy to get. I recommend XICOR but they are very expensive and the best alternative is ATMEL for the price.

As you notice in the figure of the EEPROM the addresses A13 and A14 are in parentheses. This is because you can use an 8-KByte EEPROM instead of the 32-KByte to lower the price of your application. In that case those addresses are not needed. If you do use an 8-KByte, please *remember* that, it will appeared in multiple addresses i.e $8000, $A000, $C000, and

at $E000 as the EEPROM chip-select is made to respond from $8000 to $FFFF.

To use ***only one 8-KByte*** EEPROM you first debug your design using the included monitor "MONITEUR" using the 32-KByte. After being sure your software is perfect, edit your source to change the origin statement, the stack pointer, and the RESET vector. Recompile, download your program and *"voilà"*. You use only your code without "MONITEUR" and usually 8 KBytes will be more than sufficient.

## EEPROM SECURITY

The most probable time the EEPROM can modify itself is when the power turns on or off.

Usually when the power is below 3 Volts, there is a hardware data protection made out of a voltage detector inside the device, that disables the write operation. There is also a software protection consisting of a proper sequence of operations, the user has to follow, in order to write to the EEPROM.

The first 8-KByte EEPROM had only the hardware data protection and the 32-KByte had both hard and soft one.

A few years ago I was working on an Electronic Security System and nobody trusted EEPROM to store the data if it didn't have software protection. I made a 50 Hertz switch to turn on and off the power supply of the system. I used XICOR, and CATALYST parts and I proved that none of the bits of the 8-KByte was modified even after I ran 2 million cycles of power on and power off. So software protection is a very nice looking feature but not really necessary.

XICOR made a special EEPROM with address and data multiplexed, the X88C64. "The feature provides the ability to perform non-volatile memory updates in one array and continue operation out of code stored in the other array, effectively eliminating the need for an auxiliary memory device for code storage."

The part is very expensive and the only device it eliminates is the demultiplexer. It was made especially for the INTEL micros as those can not execute from their internal RAM.

## RAM MEMORY

There is enough internal RAM inside the 68HC11A, 256 bytes, (512 bytes for the E) to fill the needs of most control circuits. If you need more memory, it is possible to add 8 or 32 KBytes of external RAM since

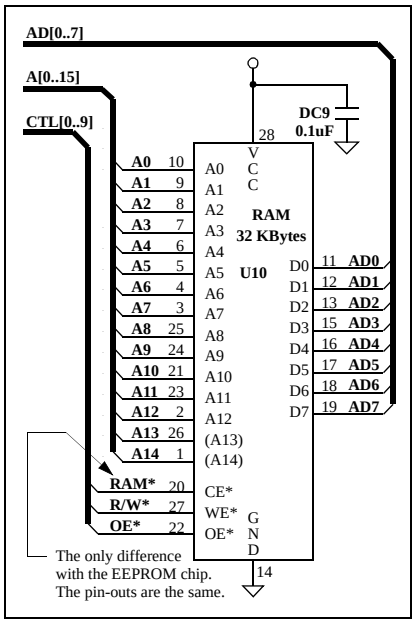



Fig: 11 RAM 32-KByte

the decoder will respond to the RAM "chip select" from $1040 to $7FFF.

For the addresses A13 and A14, the same reason should be applied to the RAM as for the EEPROM chip.

You just have to make sure the pin-out is the standard JEDEC, the speed is fast enough for the clock cycle of the CPU, and when read after have been written it doesn't toggle pin # 1 as some older non-standard devices do.

## MEMORY MAP

```
$0000-$00FF Reserved for the user
$0100-$01FF Other HC11, i.e. "E",
            internal RAM
$0200-$020F Reserved
$0210-$021F LCD & KBY
$0220-$022F UIO (Relays & Opto cou-
            plers)
$0230-$023F GAL Programmer
$0240-$027F SPARE chip select for WW
$0280-$02BF *** If Read, enable RTC
            chip select for SPI
            *** If Written, disable
            RTC chip select for SPI
$02C0-$0FFF Reserved for future ex-
            pansion and I/O, in 16
            bytes increment
$1000-$103F MCU Registers
$1040-$7EFF User RAM

            MONITEUR
$7F00-$7F40 User Stack for "MONI-
            TEUR"
$7F41-$7F53 Reserved
$7F54       B7..B4, 10 of seconds
            B3..B0, unit of seconds
$7F55       B7..B4, 10 of minutes
            B3..B0, unit of minutes
$7F56       B7..B4, 10 of hours
            B3..B0, unit of hours
            *** To set the RTC time,
            the order are SSMMHH for
            $7F54 to $7F56 and
            HHMMSS to read the RTC
            time (***$7F54-$7F56 will
            be modified in near fu-
            ture***)
$7F57       Seconds for RTC
$7F58       Minutes for RTC
$7F59       Hours for RTC
$7F5A       Day of the week for RTC
$7F5B       Day of the month for RTC
$7F5C       Month for RTC
$7F5D       Year for RTC
$7F5E-$7F95 "MONITEUR" Stack
$7F95-$7FF0 "MONITEUR" Storage RAM

$7FF1       JMP SCI
$7FF4       JMP SPI
$7FF7       JMP TOC5
$7FFA       JMP XIRW
$7FFD       JMP SWI

$8000-$DEFF User program in EEPROM
$DF00-$E857 In-line Assembler
$E858-$EB11 Disassembler
$EB12-$FF64 "MONITEUR"

$FF65-$FFCE JUMP Table
            All JMPs are Buffalo
            3.XX Compatible except
            for VECTINIT

$FFD0-$FFD5 Special Reserved

$FFD6-$FFFF CPU Vector Table
```

## MEMORY DECODING

The decoder's output chip selects respond from $0240 to $02BF for the user WW area on the CPU board, from $0280 to $03BF for the Real Time Clock, from $1040 to $7FFF for the RAM, and from $8000 to $FFFF for the EEPROM. The decoder circuit is achieved by a #PALCE22V10.

The best choices, for the hardware, are the PALCE because they are electrically erasable and cost around $3.00 US per unit. I am al-

ready gathering data to design a board for the **MicroNator** system to program GAL and PALCE.

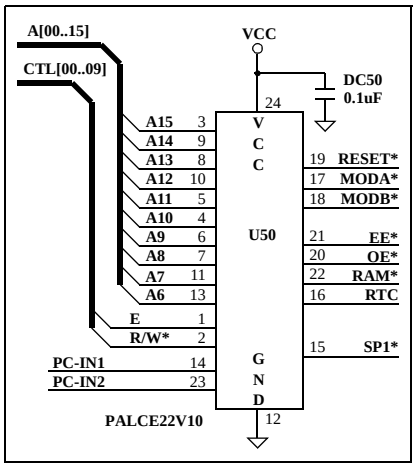



Fig: 12 Decoder

This decoder circuit is straightforward. Notice that SP1* is not required for the proper operation of the system. Its purpose is to have SP1* (spare #1) responding from $0240 to $027F. Those addresses are reserved for the user if he wants to add some circuits in the CPU board wire-wrapped area. This pin is connected to the wire wrapped hole below the capacitor at the left of the CPU crystal, see the figure on page 10.

If you want to build the CPU circuit and you don't have access to a GAL programmer there is another alternative to the decoder. You can design it using only one 74HC00 and one 74HC02.

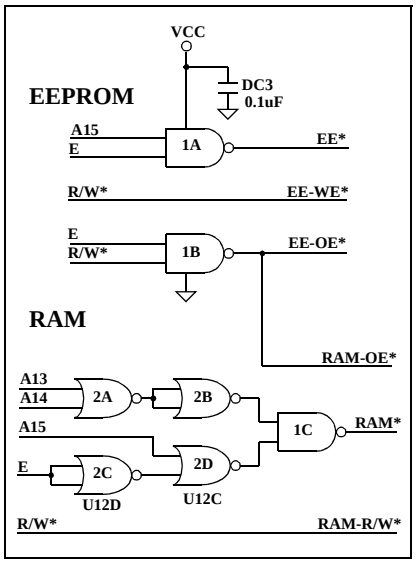



Fig: 13 Memory Decoder Alternative

That way the EEPROM will respond from $8000 to $FFFF and the RAM from $2000 to $7FFF. You will lose the first 8 KBytes of the RAM but 24 KBytes is quite enough for most situations. You will have to decode the RTC or tie the SS* pin of the CPU to the SS pin of the RTC and provide your own routine for the clock. The downloader will not work with this circuit.

## REAL TIME CLOCK

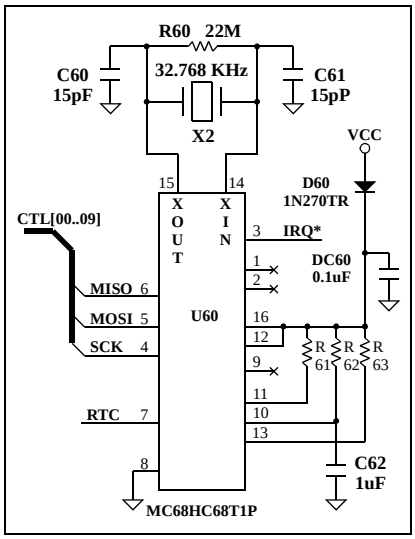



Fig: 14 Real Time Clock

From $0280 to $03BF the decoder will select the Real Time Clock. The chip select of the RTC is its SS pin # 7. It is active HIGH and it has to stay HIGH for the entire communication time when using the SPI protocol. When not in communication it has to be LOW.

So the decoder is made to latch to the level of the R/W* signal. If you read $0280, the RTC chips select will latch to a HIGH level and stay that way until a write to $0280 occurs.

So before using the SPI to communicate with the Real Time Clock, you do a read of $0280 by: LDAA $0280 and the SS pin of the RTC will become active i.e. HIGH and stay that way.

You exchange information with the RTC using the SPI.

After the communication is finished, you do a write to $0280 by: STAA $0280 and the SS pin of the RTC will become inactive i.e. LOW and stay that way until the next time you want to communicate with the RTC.

This way of implementing the circuit is to free the ***SS* pin*** of the CPU in case the user wants to add another ***slave*** or even another ***master*** on the SPI bus.

## SERIAL COMMUNICATION

The designer has to keep in mind that the connector and socket industry is one and a half bigger than the silicon industry worldwide, so price is a factor not to overlook in interconnection.

To keep as much room as possible for the WW area on the CPU board I choose a DB9 connector for the serial port. It is the small footprint one 0.318" deep.

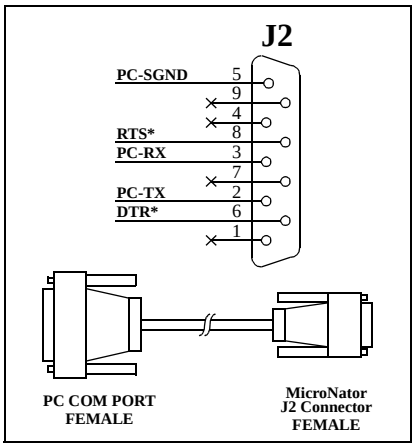



Fig: 15 Serial Connector DB25-DB9

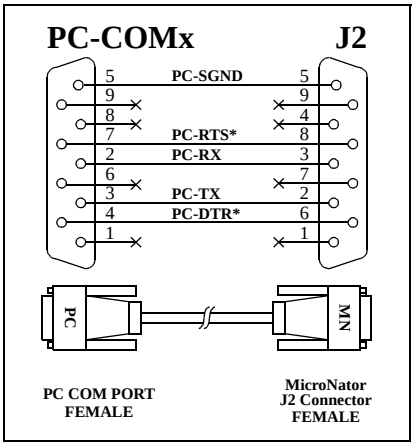



Fig: 16 Serial Connector DB9-DB9

I chose a male connector for the CPU circuit board because that way it is possible to use an old AT-Cross-Over cable for the serial communication with the PC. Again it is standard and the cost is lower.

It is still possible to use a RJ-11 telephone connector but the problem is that it stands too high and it doesn't fit well between two boards especially if the top one has IC pins right over the top of the RJ-11 connector of the lower board.

The PC-SGND goes to the DC GND of the **MicroNator** system. The other signals go to/from the line driver/receiver.

## RS232 INTERFACE

The only output to the serial communication PC cable, from the **MicroNator** system, is the PC-RX (RS-232 Receive Data of the PC). All the other communication signals are input from the PC to the CPU-11/64 system. So for the communication signals we need 1 driver and 3 receivers.

We don't want to use 12 Vdc to keep the cost of the power supply as low as possible.

For those reasons we will use a MAXIM line driver/receiver. There are many different kinds of them but a good choice is the MAX236CNG which is a 24DIP300, requires only four small 1 uF capacitors to generate the RS-232 voltage, and has a ***power down pin*** to lower the power and still be able to receive

It is possible to design a system with a MCU and a RTC that use only ***25 to 30 µA*** in power-down mode.

Why not the MAX232 which is more standard? The MAX232 has only 2 receivers and the circuit needs three, meaning that two of those chips will be required, and with them more capacitors, meaning more than the 24 pins of the MAX236 plus the 8 pins for the capacitors. The cost is almost the same but not the footprint.

DC5 and DC5A are decoupling capacitors. DC5A is for filtering the high speed switching and DC5 is to supply current when many or all internal signals are all switching at the same time. Capacitors C4, C5, C6, and C7 are for the internal charge pump circuit. They store the energy for the RS-232 voltage level converter.

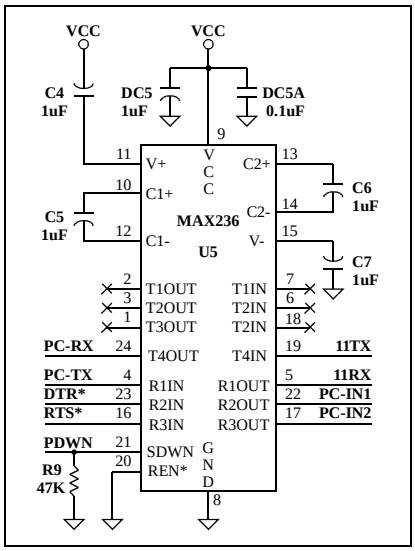



Fig. 17. RS232 Drivers & Receivers

It is not necessary to have pull-downs on the unused pins of the MAX236 for it has internal ones.

This is the reason why, when there is nothing connected to the J2 connector (stand-alone-mode), the **MicroNator** system is still operating properly.

Resistor R9 is for the user to be able to drive this pin HIGH if he wants to shut down the chip to use minimum energy for special low power system.

Please, keep in mind that the ouputs of the MAX236 are inverted in relation to their inputs.

The 11TX (HC11 SCI-Transmitter) and the 11RX (HC11 SCI-Receiver) go directly to pins 21 and 20 of the CPU.

11TX and 11RX also go to pins 26 and 24 of the ***bus connector***.

I recommend ***pulling up*** the TX pin of the MCU as, under certain circumstances, port D is configured as open drain output.

## MCU I/O CONNECTOR

The main I/Os from the MCU are brought to a DB25 connector at the rear of the system.

I chose a DB25 connector because it is standard, everybody has one to spare, it is guaranteed for 1 Amp, and is solderable. I chose a female connector as not to short pins together when inserting a wire, or a pin, into it when testing.

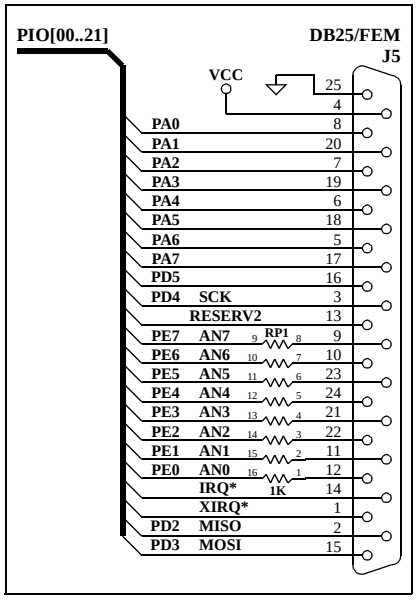



Fig: 18 MCU I/O Connector

Port A is available, on the bus, but please ***take note*** that PA3 is also used as OC5 for the "TRACE" command as explained further in the "MONITEUR" paragraph.

Port PD0 and PD1 are not on the I/O since they are reserved for the communication with the PC.

All the pins of port E are for A/D and have a 1 Kohms resistor in series with each channel to limit the current into each pin. Refer to section 12.16/ 12.17 of the HC11 Reference Manual for further explanations.

Pin 13 is reserved for future expansion. Most likely, in future versions, it will be E.

Pins 2, 3, and 15 are "reserved" for the SPI communication on expansion board. Pin 16 "PD5/SS*" is not used for the Real Time Clock SPI communication protocol so to be able to have more than one master in the system. That way, pin 16 is available for I/O.

IRQ* and XIRQ* are also present as they are frequently used in

data acquisition.

None of the I/O are buffered (except for AN0-AN7, with 1 K resistors) and they are directly connected to the MCU. So take special care when working with them as you might blow the CPU.

## BUS CONNECTOR

I thought a lot about this connector but I still believe I have never made a better choice than this one.

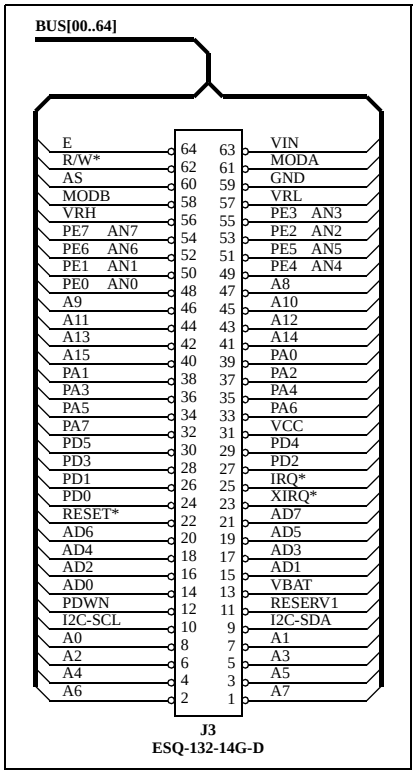



Fig: 19 Bus Description

All of the pins of the MCU are available on the bus and 3 pins are reserved for future expansion (BUS11, not dedicated at all. BUS10 and BUS9 for I2C-SCL and I2C-SDA) Some pins required special attention: PD0 #24, PD1 #26, PD2 #27, PD3 #28, PD4 # 29 and PA3 #31.

PD0 and PD1 are the SCI receiver/transmitter pins so they are also connected to U5 for communication with the PC.

PD2 "MISO", PD3 "MOSI", and PD4 "SCK" are used with other expansion boards for SPI communication protocol. The SPI communication is quite popular and there are a lot of good chips such as serial EEPROM and D/A, to name just a few, that support it.

Finally PA3 is used as OC5 by "MONITEUR" for the TRACE command as further explained in the "MONITEUR" paragraph.

## BOOTSTRAP MODE

The HC11 family has the most marvelous feature of all the 8-bit microcontrollers on the market to-day: the **"BOOTSTRAP"** mode of operation. The word comes from the early days of computer, when the designers wanted to boot a system. They strapped with wires the address bits of the first instruction to VCC or GND before cycling the system. In the good old days, I saw this kind of operation on a PDP-11 from Digital Equipment.

The HC11 enters the bootstrap mode when both MODA and MODB pins are LOW and a RESET condition occurs.

When the RESET pin goes from LOW to HIGH the first thing the CPU does is to sense the voltage level of MODA and MODB pins. If those pins are both LOW the CPU enters the special BOOTSTRAP mode of operation.

***"THE BEAUTY OF THIS SYSTEM"***

## FORCING MICRONATOR TO ENTER BOOTSTRAP MODE

To enter the **MicroNator** system into the bootstrap mode of operation is a condition simple to achieve. All the operations are done by the PC through the COM port and the PALCE22V10 under the control of the TALK communication special program.

## TALK

Written in Borland C++, TALK, is the communication program between the PC and the **MicroNator** system. It takes care of the communication and downloading of programs.

BS_EEPRM.BIN is used by TALK for the downloading of programs from the PC into the system memory. This special program **has to be in the same directory** as TALK.EXE if you want to be able to download any program. The same applies to UCT2.S19 if you want to download the "MONITEUR" program.

The switches to be used with TALK are [-?], [-r], [-f], [-p] and [-b]. They are always preceded by "-".

- "Talk -?" will print a small help screen giving a brief description of the available switches.
- "Talk -r-" will not reset the system when establishing the communication. The default (-r+) is to reset the system after contact is made. It is handy if you are in the middle of a program and you want to return to DOS for any reason. To go to DOS you just have to exit "TALK" with the standard "ALT X" command. You do what you want in DOS and then execute "talk -r-". A new white screen will appear but you will be at exactly the same place as you were before executing the "ALT X" command.
- "Talk -f" selects the French language interface for the dialogue boxes and the small help file, when you are in "MONITEUR". Default is English. There is a Spanish interface under development.
- "Talk -p1" selects the COM1 port (the default) of your PC for the communication with the system. "Talk -p2" selects COM2.
- "Talk -b19200" sets the baud rate to 19,200 (the default). Other choices are 2,400, 4,800, and 9,600 baud.
- "Talk" with no switches will select English language, COM1 port, 19,200 baud and will reset the **MicroNator** system after having established the communication with it.

## "MONITEUR"

UCT2.S19 is the binary file of the "MONITEUR" program. It is almost the same as BUFFALO but has more routines such as the one used to write to the system external EEPROM.

To reload "MONITEUR", in the event the user corrupts the RESET vector or his program modifies "MONITEUR" so much that the system doesn't responds anymore, you just

press “ALT B”. TALK will take care of all the necessary steps to re-initialize the system to its factory settings.

After a user program is downloaded, you can use “MONITEUR” to verify/change memory, insert breakpoints, single step, etc... this is the same as the BUFFALO monitor.

You can use the subroutines included in “MONITEUR” with your program. The jumps’ addresses start at $FF65 and are all listed in the User Manual. Those routines are compatible with BUFFALO 3.XX except for the VECTINIT

Talk is always running on top of the monitor and if you need some help with the different commands offered, press “F1”. A small help screen will appear showing the available choices.

***TRACE*** is a special command to single step a program. Please *take note* that it is not running exactly as the one included in the BUFFALO monitor. I use the OC5 to generate a standard interrupt to the system instead of using XIRQ.

There are three reasons for that choice. First, it makes PA3 available for I/O. Secondly, it liberates XIRQ, which is a very important pin, in data acquisition, to monitor the power. Thirdly, I hate jumpers. I always forget which one is which and what they are for. As you can notice, there are none in the **MicroNator** system, so I resolved the problem.

There is only one small disadvantage of using standard interrupt compared to the XIRQ. To use TRACE, ***the interrupts have to be enabled***. The way around is to enable interrupts only when you are debugging and when your program is running perfectly, you disable them if you have to. Please *remember* that ***the response***, from the CPU, to the interrupt service subroutine disables the interrupts and that **OC5** pin is the same as **PA3**.

It is always possible to TRACE a routine with “MONITEUR”, even an IRQ one, as long as the interrupts are enable. The IRQ routine doesn’t disable the interrupts, it is the CPU who does it, ***when responding*** to the interrupt.

## DOWNLOADING PROGRAMS

To download a user program to the **MicroNator** system you just have to press “ALT L” when you are running TALK. A dialogue box appears requesting the name of the file to load into the system memory. You don’t have to specify the extension of the file. The name only is sufficient as TALK will add the extension “S19”. The format has to be the Motorola S19. The user program can be loaded anywhere into external RAM or external EEPROM.

## WRITE_EEPROM SUBROUTINE

As said earlier, the only disadvantage of EEPROM is that they go tri-stated 100 uSec after being last written to and then for 10 mSec you have no more access to it, i.e you can not read or write it. During those 10 mSec, the EEPROM is busy programming the byte(s).

If the program is stored entirely into only one EEPROM, you cannot have access to it, for the 10 mSec delay subroutine stored in it. You must execute the delay from another device, usually a system RAM.

If you want to keep the cost and size of your system as low as possible ***and don’t want to add a RAM chip*** you have to use, for the 10 mSec delay subroutine, the only RAM available to your system: ***“THE INTERNAL RAM”.***

With the Motorola micros the internal RAM is part of the standard memory map meaning you can use it to execute code. ***So you store the delay subroutine inside the CPU***. You have two choices: store the routine at fixed addresses or pass it through the stack.

Let’s say we want to store $55 at address $A000. One way the code might look is as follows:

```
0000 A7 00    EEWRITE  STAA   0,X
0002 CE 0A 00          LDX    #$0A00
0005 09       LOOP_EE  DEX
0006 26 FD             BNE    LOOP_EE
0008 39                RTS
```

If you want the code at fixed addresses, you store those nine bytes at $0000 after RESET when initializing the system. When you want to write to the EEPROM, you put the data into the accumulator ‘A’ and the address into ‘X’ then you call EEWRITE by the instruction “JSR $0000”. On return, as seen from the above example, $55 is stored at $A000.

While an EEPROM is busy programming its data, it will output the last byte written, in its complemented form, when read. If the last byte was $FF and you read the EEPROM, you will read $00 and when the EEPROM is finished programming, it will output the exact data, $FF.

Usually it will take much less than 10 mSec to do its job, so you can shorten the time by checking the last byte. The following subroutine is much faster:

```
0000 A7 00    EEWRITE  STAA   0,X
0002 A1 00    LOOP_EE  CMPA   0,X
0004 26 FC             BNE    LOOP_EE
0006 39                RTS
```

The danger with this routine is that it might be an endless loop if the byte doesn’t program properly. Please remember that all the EEPROM are not born equal. Some are better than others. ***This routine will not work with all EEPROM***.

You can use the COP to loop out in 10 Msec if the EEPROM fails to write properly.

With the first routine, you can verify, at the end of the delay, if the writing was successful and if not, you can send an error message.

Without external memory RAM in earlier **MicroNator** System and since there were only 256 bytes of internal RAM in HC11A0, I decided to pass the delay subroutine through the stack, which is a little more tricky. I still keep the same strategy for version 4.00 but this time the STACK is in external RAM.

## CONCLUSION

**MicroNator** is an easy to use system with a lot of potential. It is becoming a standard as it is so simple to

expand. It is the perfect choice for colleges and universities as well as for the serious control designers. ***A "C" language and a BASIC11 interpreter with EEPROM storage is available.***

To use the BASIC11 interpreter you just download it with the "alternate-L" and it is ready to be use. No jumper to solder, no PCB traces to cut, nor EEPROM chip to add. Just download and enjoy...

| | |
|---|---|
| AT | Advanced Technology. |
| CMOS | Complementary Metal Oxide Semi-conductor. |
| COM | Communication Port. |
| CPU | Central Processing Unit. |
| CTS | Clear To Send. |
| DIP | Dual In line Package. |
| DOS | Disk Operating System. |
| DTR | Data Terminal Ready. |
| E | Enable, system clock. |
| EEPROM | Electrically Erasable Programmable Read Only Memory. |
| EPLD | Electrically Programmable Logic Device. |
| EPROM | Erasable Programmable Read Only Memory. |
| GAL | Generic Array Logic. |
| GND | Ground potential, 0 Volt DC, logic 0. |
| HC | High Speed CMOS. |
| I/O | Input / Output. |
| JTAG | Joint Test Action Group. |
| LED | Light Emitting Diode. |
| LIR | Load Instruction Register. |
| LOW | Low Voltage Level, 0 Volt DC, logic 0. |
| MCU | Micro Controller Unit. |
| PALCE | Programmable Array Logic CMOS EEPROM. |
| PC | Personal Computer. |
| PCB | Printed Circuit Board. |
| PLCC | Plastic Leaded Chip Carrier. |
| RAM | Random Access Memory. |
| RN | Resistor Network. |
| ROM | Read Only Memory. |
| RTS | Request To Send. |
| SCI | Serial Communication Interface. |
| SIP | Single In-line Package. |
| SMT | Surface Mount Technology. |
| SPI | Serial Peripheral Interface |
| TTL | Transistor-Transistor Logic. |
| VCC | Voltage Continuous Current. |
| WW | Wire-Wrapped. |

Table 2. Acronyms

**PARTS for MicroNator System**

**Resistors**

| | |
|---|---|
| **R1** | 470, 1/8W, 5% // |
| **R7** | 10M, 1/8W, 5% // |
| **R9, R61, R62** | 47K, 1/8W, 5% // |
| **R10, R63** | 1K, 1/8W, 5% // |
| **R60** | 22M, 1/8W, 5% // |
| **RN1** | 47K, 7-RES BUS, SIP-8 /DALE #4608X-101-473 // |
| **RN2** | 4K7, 7-RES BUS, SIP-8 /DALE #4608X-101-472 // |
| **RP1** | 1K, 8 ISOLATED, DIP-16 /DALE #4116R-001-102 // |

**Capacitors**

| | |
|---|---|
| **C2, C1** | 33pF, 200V, CER., 0.200" / AVX-CK05BX330K // |
| **C3, C4, DC5, C5, C6, C7, C62** | 1uF, 35V, Tantalum, 0.100" // |
| **C60, C61** | 15pF |
| **DC1, DC2, DC4, DC5A, C8, DC9, C10, C12, C13, DC50, DC60** | 0.1uF, 50V, CER., 0.200" / AVX-CK05BX104K // |
| **C9,C11** | 47uF, 35V47, RADIAL.100 // ACTIVE |

**Diodes**

| | |
|---|---|
| **D2, D1** | 1N4001TR // |
| **D3** | Red LED, 1.0MCD, 1.6V FWD, T-1 3/4 // |
| **D4, D60** | 1N270TR, Germanium, 80V, 200mA, DO-7 //ACTIVE |

**ICs**

| | |
|---|---|
| **U1** | CPU, PLCC 52 /MOTOROLA # MC68HC11A1FN // |
| **U2** | SN74HC373N, OCTAL Demux., 3ST, NON-INV /TI // |
| **U4** | 32K x 8 EEPROM, 28DIP600 /XICOR #X28C256D-25 // **ONLY XICOR** |
| **U5** | RS-232 DRV-RCV, 4IN-3OUT, 24DIP300 /MAXIM #MAX236CNG // |
| **U6** | Regulator, 5V/1A, TO-220 / TI #UA7805UC / |
| **U10** | 32K x 8 RAM, 350ns or better, 28DIP600 / #62256 // |
| **U50** | PALCE22V10H-25, // ACTIVE |
| **U60** | MC68HC68T1P, // |
| **X1** | Crystal, 4.9152MHz, HC-49U / KDS, FOX // ACTIVE |
| **X2** | Crystal, 32.768 KHz, Watch // |

**Connectors & Switches**

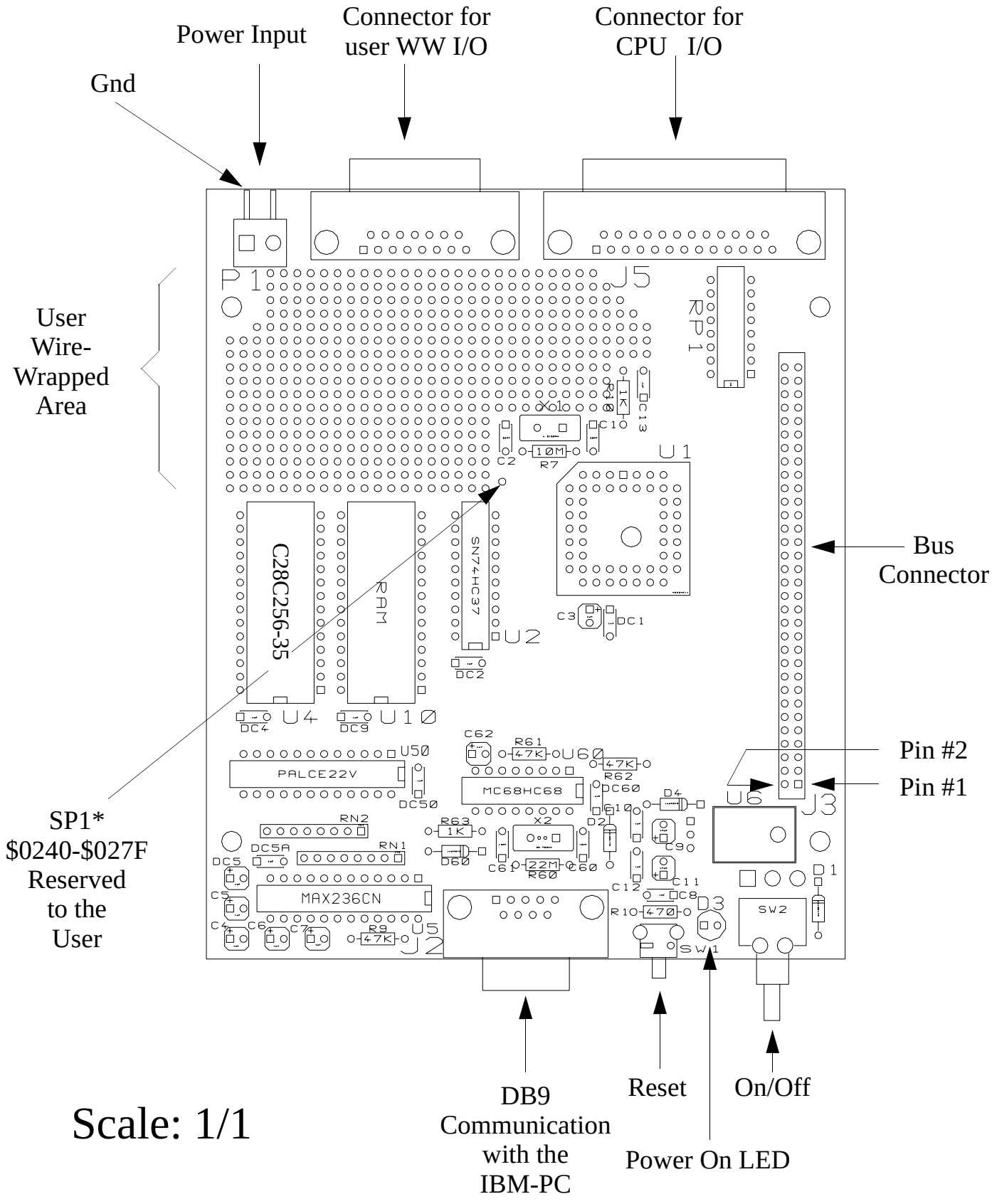
| | |
|---|---|
| **J2** | DB9-RH, PCB, MALE 0.318 // |
| **J3** | 32x2 / SAMTEC # ESQ-132-14G-D // |
| **J5** | DB25, PCB, FEM, RH, 0.318" /SPC // |
| **P1** | 2 PINS, 5mm, MALE / WECO #90,871,012 // For the female for Wall-power-supply / WECO #20,873,102 |
| **SW1** | SPST MOM-NO, 4 PINS, PCB, RH / OMRON #B3F-3152 // |
| **SW2** | SPDT, 5Amp, 28Vdc, PCB, RH / SWITCHCRAFT #A1010MD7AQ // |

---

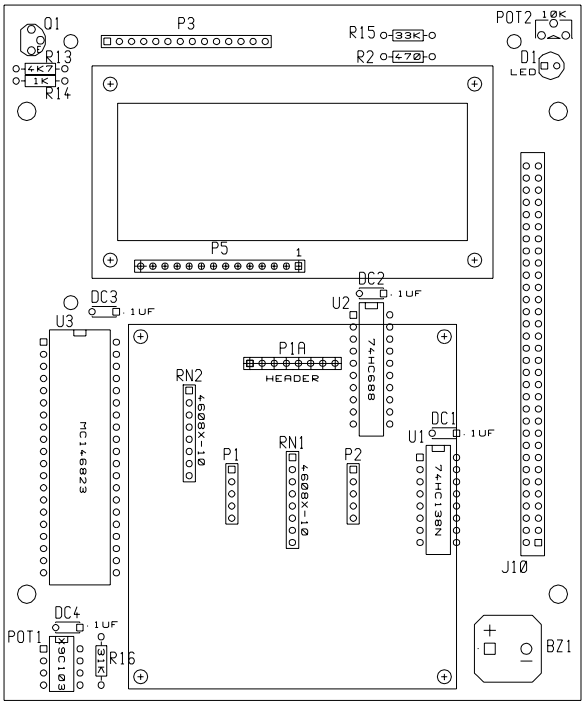

**RF-232**
**1404 rue Galt**
**Montréal, Qc H4E 1H9**
**CANADA**
**Tél: (514) 761-4201**

**RF-232**
**21 rue André Gide**
**59123 ZUYDCOOTE**
**FRANCE**
**Tél: 03 28 58 28 39**

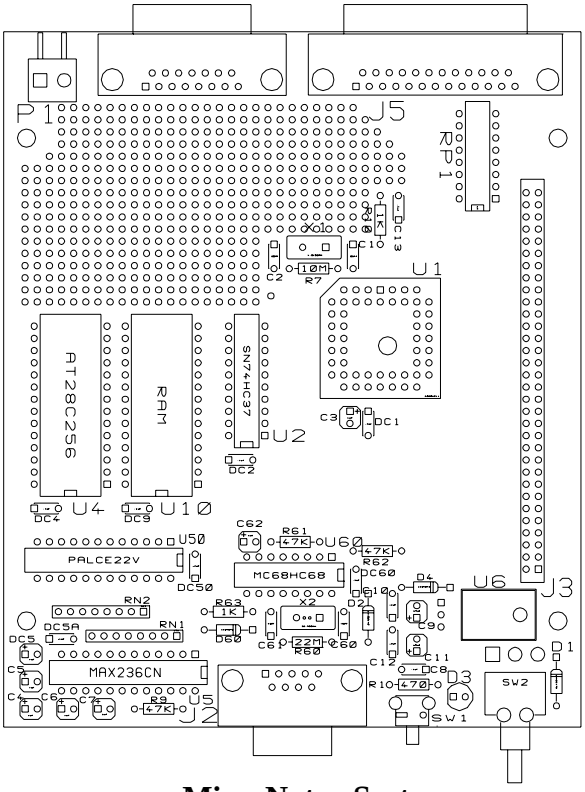
**68hc11@micronator.com**
**support@micronator.com**



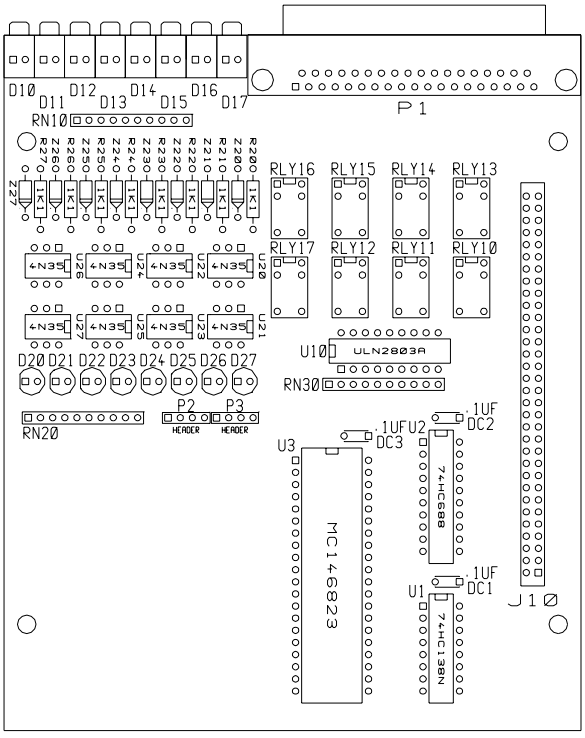

Scale: 1/1



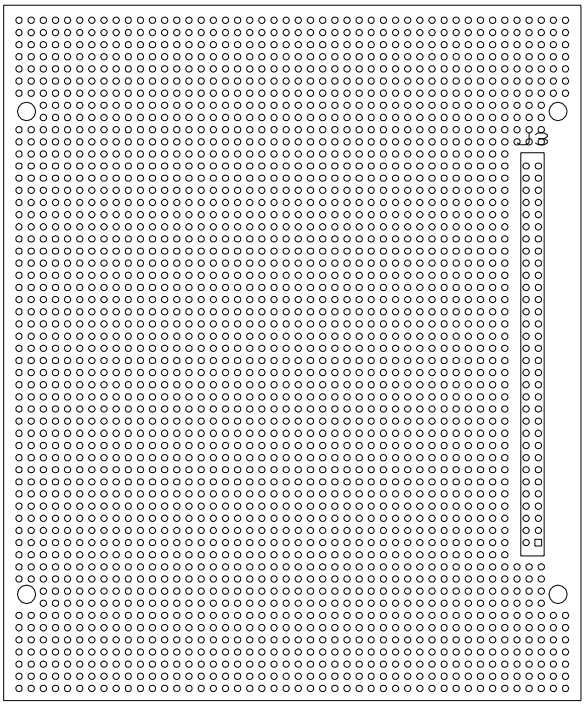

**User Board (KBY & LCD)**



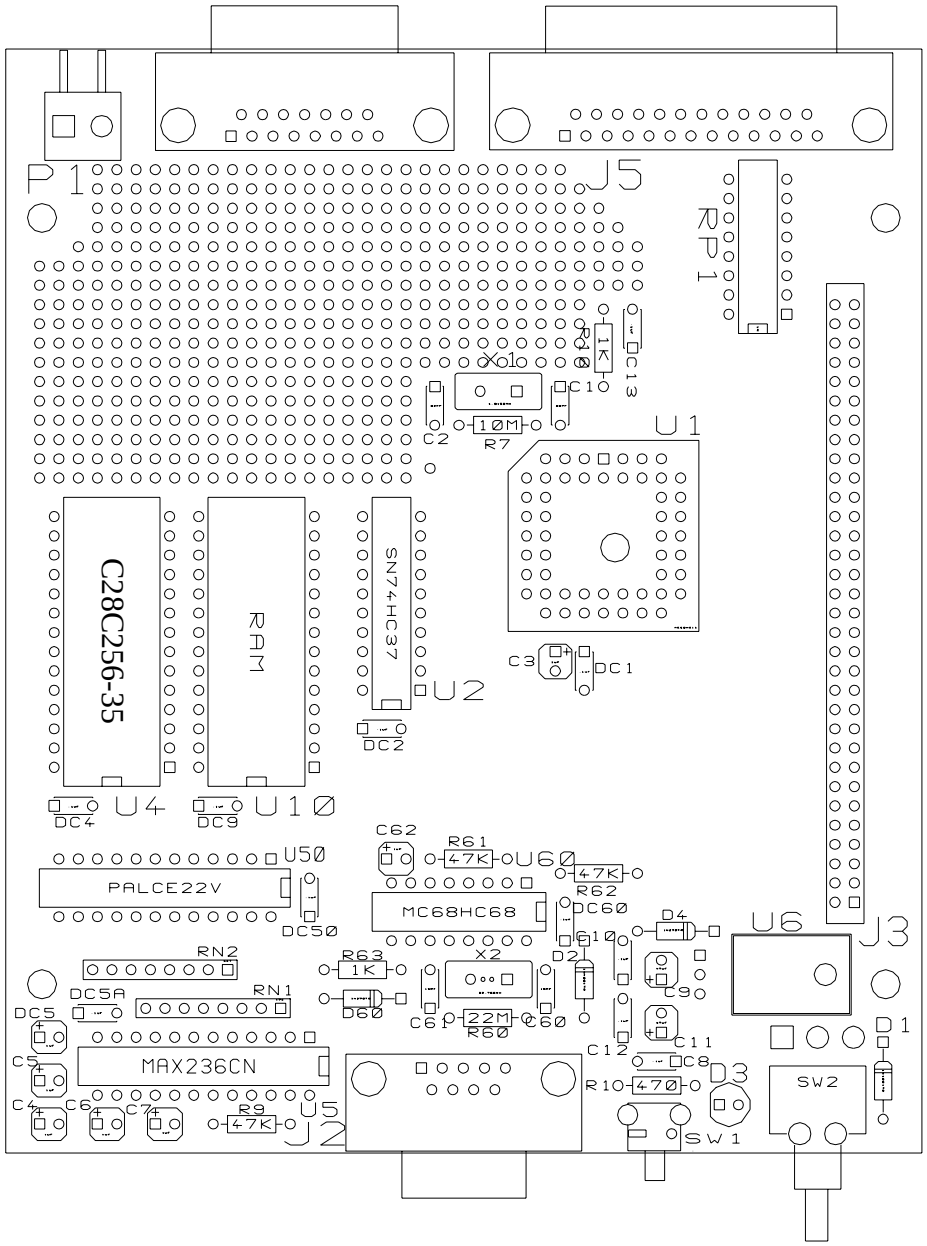

**MicroNator System**




**User I/O (Opto & Relays)**



**Expansion Wire Wrapped Board**




**RF-232**
**1404 rue Galt**
**Montréal, Qc H4E 1H9**
**CANADA**
**Tél: (514) 761-4201**

**RF-232**
**21 rue André Gide**
**59123 ZUYDCOOTE**
**FRANCE**
**Tél: 03 28 58 28 39**

**micronator@micronator.com**