

# Micromint Modules

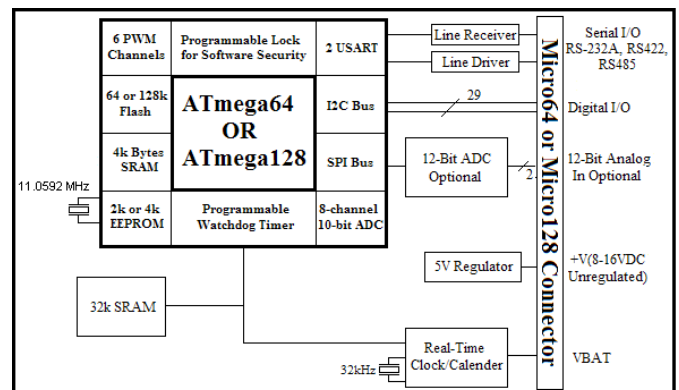
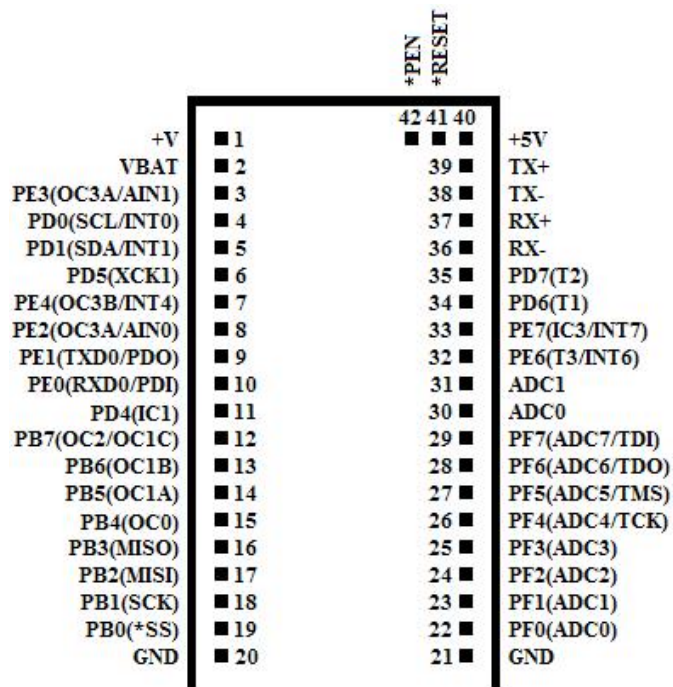


## Microcomputer/Controller Featuring the ATmega64 or the ATmega128

### FEATURES

- Small size – complete computer/controller with I/O less than 1.5 cubic inches (1.5” x 2.1” x 0.52 )
- Low power only 275 mW typical
- Dual powered – operates on +5V or 6.5-20V at 55 mA (typical)
- Program and Data Memories
  - 64k or 128k Bytes of In-System Reprogrammable Flash with 10,000 Write/Erase Cycles
  - In-System Programming by On-chip Bootloader
  - 2 or 4 K Bytes EEPROM with 100,000 Write/Erase Cycles
  - 36K Bytes SRAM
  - Programming lock for Software Security
  - SPI interface for In-System Programming

- Peripheral Features
  - Real-Time Clock Calendar
  - Optional 2-channel 12-bit ADC
  - 8-channel 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels
    - 2 Differential Channels with Programmable Gain (1x, 10x, 200x)
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
    - 1 TTL
    - 1 RS-232A, RS-422, or RS485
  - Master/Slave SPI Serial Interface
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 1 to 16 Bits
  - Programmable Watchdog Timer with On-chip Oscillator
  - On Chip Analog Comparator
  - 29 Digital I/O that can sink or source 20mA



## ABSOLUTE MAXIMUM RATINGS

Operating Temperature:		Voltage on +5V (Pin 40) referenced to GND	0 to +5.5 VDC Regulated
Commercial	0°C to +70°C	With pin 1 open	
Industrial	-40°C to +85°C	Voltage on Vbat (pin 2) referenced to GND	0 to +5.5 VDC Regulated
Storage Temperature	-50°C to +125°C		
Voltage on +V (Pin 1) referenced to GND	0 to +16 VDC Unregulated		

Industrial temperature version is available; minimum quantities apply.

## PIN DESCRIPTIONS

Micro64/128 is a 40-pin package (2.25" x 1.4" x 0.5") with 0.1" pin and 1.2" row spacing. Some pins have multiple functions depending on system configuration. **DIO – Digital Input/Output**

### Pin Signal Description

<b>1</b>	<b>+V</b>	Micro64/128 power supply input. +V is nominally 8-16 VDC. If pin 1 is open, Micro64/128 can be powered with +5 VDC directly on pin 40.
<b>2</b>	<b>Vbat</b>	2.5 VDC to 5.5 VDC Battery backup input for the optional Real-Time Clock Calendar.
<b>3</b>	<b>PE3</b>	DIO/AIN1/OC3A (Analog Comparator Negative Input or Output Compare and PWM Output A for Timer/Counter3)
<b>4</b>	<b>PD0</b>	DIO/INT0/SCL (External Interrupt0 Input or I <sup>2</sup> C clock) Optionally used as the 12-bit ADC Chip Select .
<b>5</b>	<b>PD1</b>	DIO/INT1/SDA (External Interrupt1 Input or I <sup>2</sup> C data) Optionally used as the 12-bit ADC Data I/O .
<b>6</b>	<b>PD5</b>	DIO/XCK1 (USART1 External Clock Input/Output) Optionally used as the 12-bit ADC Clock .
<b>7</b>	<b>PE4</b>	DIO/INT4/OC3B (External Interrupt4 Input or Output Compare and PWM Output for Timer/Counter 3)
<b>8</b>	<b>PE2</b>	DIO/AIN0/XCK0 (Analog Comparator Positive input or USART0 External Clock Input/Output)

### Pin Signal Description

<b>9</b>	<b>PE1</b>	DIO/PD0/TXD0 (Programming Data Output or UART0 Transmit Pin)
<b>10</b>	<b>PE0</b>	DIO/PDI/RXD0 (Programming Data Input or UART0 Receive Pin)
<b>11</b>	<b>PD4</b>	DIO/IC1 (Time/Counter1 Input Capture Trigger)
<b>12</b>	<b>PB7</b>	DIO/OC2/OC1C (Output Compare and PWM Output for Timer/Counter2 or Output Compare and PWM Output C for Timer/Counter1)
<b>13</b>	<b>PB6</b>	DIO/OC1B (Output Compare and PWM Output B for Timer/Counter1)
<b>14</b>	<b>PB5</b>	DIO/OC1B (Output Compare and PWM Output A for Timer/Counter1)
<b>15</b>	<b>PB4</b>	DIO/OC0 (Output Compare and PWM Output for Timer/Counter0)
<b>16</b>	<b>PB3</b>	DIO/MISO (SPI Bus Master Input/Slave Output)
<b>17</b>	<b>PB2</b>	DIO/MOSI (SPI Bus Master Output/Slave Input)
<b>18</b>	<b>PB1</b>	DIO/SCK (SPI Bus Serial Clock)

# Micro64/128

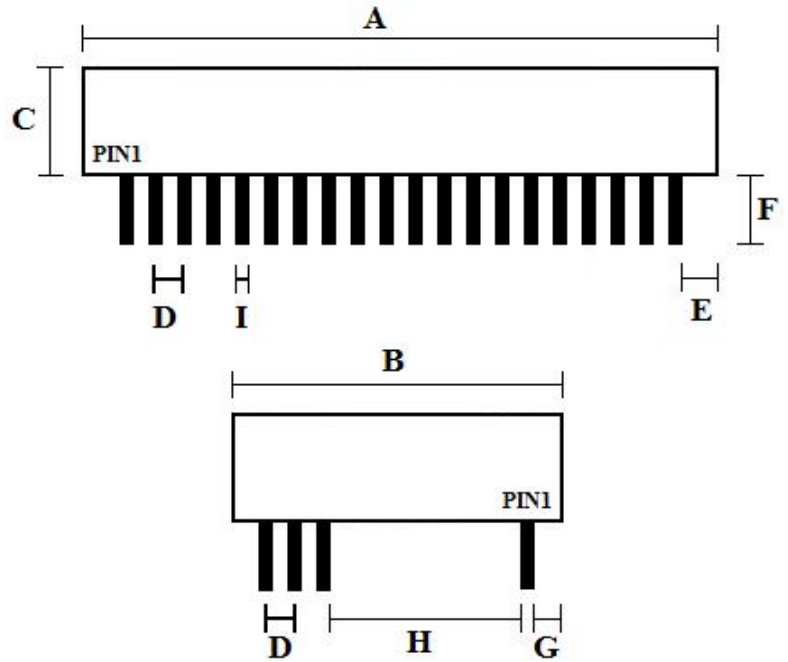
---

Pin	Signal	Description	Pin	Signal	Description
19	PB0	DIO/*SS (SPI Bus Slave Select)	33	PE7	DIO/INT7/IC3 (External Interrupt 7 Input or Timer/Counter3 Input Capture Trigger)
20	GND	Signal Point Analog and Digital Ground	34	PD6	DIO/T1 (Timer/Counter1 Clock Input) Serial Transmitter Disable Control
21	GND	Signal Point Analog and Digital Ground	35	PD7	DIO/T2 (Timer/Counter2 Clock Input)
22	PF0	DIO/ADC0 (10-Bit ADC Input Channel 0)	36	RX-	RS-422/-485/-232A Inverted Serial (Receive Pair/Recxmit Pair/Receive)
23	PF1	DIO/ADC1 (10-Bit ADC Input Channel 1)	37	RX+	RS-422/-485/-232A Noninverted Serial (Receive Pair/Rec-Xmit Pair)
24	PF2	DIO/ADC2 (10-Bit ADC Input Channel 2)	38	TX-	RS-422/-485/-232A Inverted Serial (Transmit Pair/Rec-Xmit Pair/Transmit)
25	PF3	DIO/ADC3 (10-Bit ADC Input Channel 3)	39	TX+	RS-422/-485/-232A Noninverted Serial (Transmit Pair/Rec-Xmit Pair)
26	PF4	DIO/ADC4/TCK(10-Bit ADC Input Channel 4 or JTAG Test Clock)	36	RX-	RS-422/-485/-232A Inverted Serial (Receive Pair/Recxmit Pair/Receive)
27	PF5	DIO/ADC5/TMS(10-Bit ADC Input Channel 5 or JTAG Test Mode Select)	40	+5V	This is the internal reference voltage for the 10 and 12 bit ADC. This output may be used to power minimal external circuitry or sensors. Micro64/128 may be powered on
28	PF6	DIO/ADC6/TDO (10-Bit ADC Input Channel 6 or JTAG Test Data Output)	+5V		only through this pin, provided Pin 1 is left unconnected.
29	PF7	DIO/ADC7/TDI (10-Bit ADC Input Channel 7 or JTAG Test Data Input)	41	RESET	When Brought to a logic low, RESET provides a master clear of the module
30	ADC0	12-bit ADC Channel 0 Input, Input Range 0 to +5VDC	42	*PEN	Programming enable pin for the SPI Serial Programming mode. By holding this pin low during a Power-on Reset, the device will enter the SPI Serial Programming mode. PEN has no function during normal operation.
31	ADC1	12-bit ADC Channel 1 Input, Input Range 0 to +5VDC			
32	PE6	DIO/INT6/T3 (External Interrupt 6 Input or Timer/Counter3 Clock Input)			

---

## MECHANICAL AND ENVIRONMENTAL CHARACTERISTICS

Length	2.25 inches
Width	1.375 inches
Height	0.52 inches
Weight	45 grams
Operating Temperature	0°C to +70°C (Optional -40°C to +85°C)
Humidity	0 to 100% (noncondensing)



DIM	Inches		Millimeters	
	min	max	min	max
A	2.240	2.260	56.896	57.404
B	1.365	1.385	34.671	35.179
C	0.510	0.530	12.954	13.462
D	0.095	0.105	2.413	2.667
E	0.153	0.193	3.886	4.902
F	0.220	0.280	5.588	7.112
G	0.170	0.200	4.318	5.080
H	0.990	1.100	25.146	27.940
I	0.230	0.270	0.508	0.686

## DC ELECTRICAL CHARACTERISTICS

Operating Temperature						Ta = 0°C to +70°C
Operating Voltage						Vcc = 4.75 V to 5.5V Vss = 0.0 V
Characteristic	Minimum	Typical	Maximum	Units	Condition	
Supply Voltage (V <sub>CC</sub> to Pin 40)	4.75	5	5.5	V		
(+V to Pin 1)	6.5	9	16	V		
(Vbat to Pin 2)	2.45	3	5.5	V		
Supply Current (I <sub>CC</sub> )		55	80	mA		
Input Low Voltage (V <sub>IL</sub> )	-0.5		0.5	V		
Input High Voltage (V <sub>IH</sub> )	3.5		5.5	V		
Output Low Voltage (V <sub>OL</sub> )			0.7	V	I <sub>OL</sub> = 20 mA, Vcc = 5V	
Output High Voltage (V <sub>OH</sub> )	4.0			V	I <sub>OH</sub> = -20 mA, Vcc = 5V	

# Micro64/128

COMMUNICATION LINE DC ELECTRICAL CHARACTERISTICS						
Characteristic		Minimum	Typical	Maximum	Units	Condition
Differential Driver	Output Voltage			5	V	Unloaded See Note 1
	RS-422	2		5	V	R=50 Ohms
	RS-485	1.5		5	V	R=27 Ohms
Maximum Receiver	Input voltage			+/-14	V	
	ESD Protection		2000		V	

12 BIT A/D CONVERTER CHARACTERISTICS						
Characteristic		Minimum	Typical	Maximum	Units	Condition
	Resolution	12			bits	
	Integral Nonlinearity Error		+/- 0.75	+/- 1.00	bits	
	Offset Error		+/- 1.25	+/- 3	bits	
	Gain Error		+/- 1.25	+/- 5	bits	
	Voltage Reference	4.75	5	5.5	V	$V_{REF}$ is $V_{CC}$
	Analog Input Range	$V_{SS}$		$V_{CC}$	V	
	Sample Rate			100,000	per second	

**Note 1:** RS-232A is characterized as a +/-5V bipolar signal (as opposed to RS-232C at +/-12 V). Drivers and receivers are

actually RS-42 and the interface is an RS-423 connection (single ended to differential).

## 1.0 Micro64/128 Hardware Overview

Micro64/128 is an industrial oriented controller in a 2.25" by 1.375" 40-pin DIP encapsulated package. It is a combination of an Atmel AVR technology RISC microcontroller that includes on chip Flash, EEPROM, SRAM, and other features,

an I<sup>2</sup>C Real-Time Clock/Calendar , and 32k of additional SRAM. Optionally a 2-channel 12-bit ADC can be added. **Custom hardware configurations are available upon request.**

### 1.1 Atmega64 Microcontroller

The Atmega64 has the following features available to the Micro64 user: 64K bytes of In-System Programmable Flash with Read-While-Write capabilities, 2K bytes EEPROM, 4K bytes internal SRAM, 32k bytes external SRAM, 29 general purpose I/O lines, 32 general purpose working registers, four flexible Timer/Counters with compare modes and PWM, two USARTs, a byte oriented, Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the

register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the asynchronous timer continues to run.

## 1.2 Atmega128 Microcontroller

The ATmega128 has the following features available to the Micro128 user: 128K bytes of In-System Programmable Flash with Read-While-Write capabilities, 4K bytes EEPROM, 4K bytes internal SRAM , 32k bytes external SRAM, 29 general purpose I/O lines, 32 general purpose working registers, four flexible Timer/Counters with compare modes and PWM, 2 USARTs, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down

mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

## 1.3 M41T80 I<sup>2</sup>C Real Time Clock

The M41T80 I<sup>2</sup>C real time clock/calendar has a low operating current of 200µA. Eight registers are used for the clock/calendar function and are configured in binary coded decimal (BCD) format. An additional 5 registers provide status/control of an Alarm. Addresses and data are transferred serially via a two line, bi-directional I<sup>2</sup>C interface. The built-in address register is incremented automatically after each WRITE or READ data byte.

Functions available to the user include a time-of-day

clock/calendar and Alarm interrupts. The eight clock address locations contain the century, year, month, date, day, hour, minute, second and tenths/hundredths of a second in 24 hour BCD format. Corrections for 28, 29 (leap year - valid until year 2100), 30 and 31-day months are made automatically. The Alarm interrupt's output pin is connected PE5 (INT5) of the Atmega64 or Atmega128. The I<sup>2</sup>C clock is connected to PD0 and the data line is connected to PD1.

## 1.4 Optional MCP3202 12-bit ADC

The MCP3202 is a successive approximation 12-bit Analog to Digital Converter with on board sample and hold circuitry. It is programmable to provide a single pseudo-differential input pair or dual single-ended inputs.

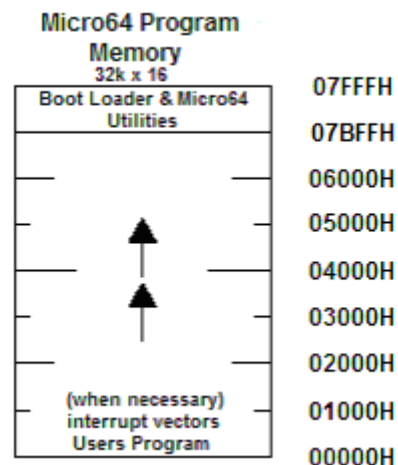
Communication to the ADC is done using a bit banded SPI bus. The ADC's chip select is connected to PD0, the data in and data out are connected to PD1, and the clock signal is connected to PD5.

## 2.0 Memory Map

Micro64 and Micro128 memory is broken up into three different sections. Flash for program space, SRAM for volatile data storage, and EEPROM for nonvolatile data storage.

### 2.1 Micro64 Program Space

Micro64 has a total of 64k of program space. The upper 2k contains the bootloader and Micro64 Utilities. This leaves 62k available for the end user.



# Micro64/128

## 2.2 Micro128 Program Space

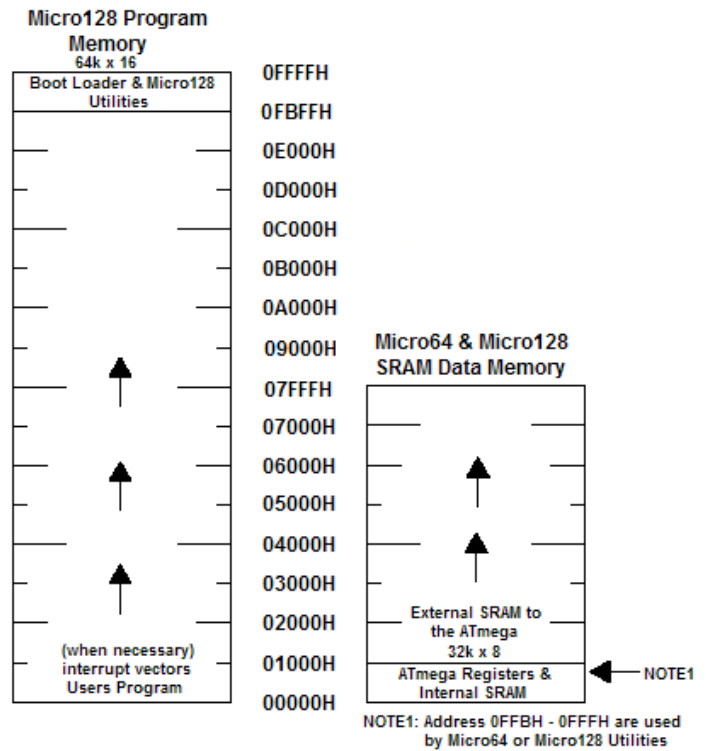
Micro64 has a total of 128k of program space. The upper 2k contains the bootloader and Micro128 Utilities. This leaves 126k available for the end user.

## 2.3 Micro64 & Micro128 SRAM Data Memory

Micro64 and Micro128 have 36k of SRAM for volatile data storage. The SRAM is broken up into three blocks. A two 4k blocks and a 28k block. One of the 4k blocks resides inside the ATmega64 or ATmega128 microcontroller. The 28k and the other 4 k block reside external to the ATmega64 or ATmega128. Micro64 and Micro128 Utilities uses 4 bytes in the 4k block. The bytes that the utilities use are located from address 0FFBH through 0FFFH.

## 2.4 Micro64 & Micro128 EEPROM Data Memory

Micro64 contains 2k bytes of EEPROM data space. Micro128 contains 4k bytes of EEPROM data space. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is done with three registers. The EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.



## 3.0 Micro64 & Micro128 Utilities

Micromint has included utilities with it's built in boot loader. Not only do you have the power of a boot loader, but you also have extra functions to help make your application extremely easy to produce. These functions are loaded into

the upper 4k of the program space along with the bootloader at the factory. The added functions include reading the 12-bit ADC, reading and writing the real-time clock colander registers, and using the I<sup>2</sup>C Bus.

## 3.1 Micro64 Utilities Function Calls

SRAM Address			Feature	Function	Functions Address	SRAM Address
0FFBH	0FFCH	0FFDH				
			12- bit ADC	Single-ended CH0	07C04H	Result
				Single-ended CH1	07BFDH	Result
				Differential +/- (CH0-CH1)	07C0BH	Result
				Differential -/+ (CH1-CH0)	07C12H	Result
			RTC	Read tenth of a second	07CF3H	Result/Error
				Read seconds	07CF7H	Result/Error
				Read minutes	07CFCH	Result/Error
				Read hours	07D00H	Result/Error

## Micro64 Utilities Continued

SRAM Address			Feature	Function	Functions Address	SRAM Address OFFEH - OFFFH
OFFBH	OFFCH	OFFDH				
			RTC	Read the day of the week	07D05H	Result/Error
				Read the day of the month	07D09H	Result/Error
				Read the month	07D0DH	Result/Error
				Read the year	07D11H	Result/Error
		Data		Write seconds	07D15H	Result/Error
		Data		Write minutes	07D1DH	Result/Error
		Data		Write hours	07D21H	Result/Error
		Data		Write the day of the week	07D29H	Result/Error
		Data		Write the day of the month	07D2DH	Result/Error
		Data		Write the month	07D31H	Result/Error
		Data		Write the year	07D35H	Result/Error
				Read the alarm seconds	07D39H	Result/Error
				Read the alarm minutes	07D3EH	Result/Error
				Read the alarm hour	07D43H	Result/Error
				Read the alarm day of the month	07D48H	Result/Error
				Read the alarm month	07D4DH	Result/Error
		Data		Write the alarm seconds	07D58H	Result/Error
		Data		Write the alarm minutes	07D6BH	Result/Error
		Data		Write the alarm hour	07D7EH	Result/Error
		Data		Write the alarm day of the month	07D91H	Result/Error
		Data		Write the alarm month	07DA4H	Result/Error
				Enable the alarm	07DB9H	Result/Error
				Disable the alarm	07DC7H	Result/Error
				Set the alarm to repeat every second	07DD2H	Result/Error
				Set the alarm to repeat every minute	07DFBH	Result/Error
				Set the alarm to repeat every hour	07E24H	Result/Error
				Set the alarm to repeat every day	07E4DH	Result/Error
				Set the alarm to repeat every month	07E76H	Result/Error
				Set the alarm to repeat every year	07EA2H	Result/Error
				Read the alarm flags	07ECBH	Result/Error
			I <sup>2</sup> C Byte Transfer	Initialize the I <sup>2</sup> C at 400kHz	07C20H	
				Initialize the I <sup>2</sup> C at 100kHz	07C23H	
Slave Address	Register			Read registered byte	07C4DH	Result/Error
Slave Address	Register	Data		Send register byte	07C8AH	Result/Error
Slave Address		Data		Send Byte	07CB8H	Result/Error
Slave Address				Read Byte	07CDDH	Result/Error
				Disable the I <sup>2</sup> C	07C37H	
			Utilities Version Number		07C19H	Result

### 3.2 Micro128 Utilities Function Calls

SRAM Address	Feature	Function	Functions Address	SRAM Address
--------------	---------	----------	-------------------	--------------

# Micro64/128

<u>0FFBH</u>	<u>0FFCH</u>	<u>0FFDH</u>			<u>0FFEH - 0FFFH</u>
			12- bit ADC	Single-ended CH0	0FC09H Result
				Single-ended CH1	0FC02H Result
				Differential +/- (CH0-CH1)	0FC10H Result
				Differential -/+ (CH1-CH0)	0FC17H Result
			RTC	Read tenth of a second	0FCF8H Result/Error
				Read seconds	0FCFCH Result/Error
				Read minutes	0FD01H Result/Error
				Read hours	0FD05H Result/Error
				Read the day of the week	0FD0AH Result/Error
				Read the day of the month	0FD0EH Result/Error
				Read the month	0FD12H Result/Error
				Read the year	0FD16H Result/Error
	Data			Write seconds	0FD1AH Result/Error
	Data			Write minutes	0FD22H Result/Error
	Data			Write hours	0FD26H Result/Error
	Data			Write the day of the week	0FD2EH Result/Error
	Data			Write the day of the month	0FD32H Result/Error
	Data			Write the month	0FD36H Result/Error
	Data			Write the year	0FD3AH Result/Error
				Read the alarm seconds	0FD3EH Result/Error
				Read the alarm minutes	0FD43H Result/Error
				Read the alarm hour	0FD48H Result/Error
				Read the alarm day of the month	0FD4DH Result/Error
				Read the alarm month	0FD52H Result/Error
	Data			Write the alarm seconds	0FD5DH Result/Error
	Data			Write the alarm minutes	0FD70H Result/Error
	Data			Write the alarm hour	0FD83H Result/Error
	Data			Write the alarm day of the month	0FD96H Result/Error
	Data			Write the alarm month	0FDA9H Result/Error
				Enable the alarm	0FDBEH Result/Error
				Disable the alarm	0FDCCCH Result/Error
				Set the alarm to repeat every second	0FDD7H Result/Error
				Set the alarm to repeat every minute	0FE00H Result/Error
				Set the alarm to repeat every hour	0FE29H Result/Error
				Set the alarm to repeat every day	0FE52H Result/Error
				Set the alarm to repeat every month	0FE7BH Result/Error
				Set the alarm to repeat every year	0FEA7H Result/Error
				Read the alarm flags	0FED0H Result/Error

## Micro128 Utilities Continued

<u>SRAM Address</u>			<u>Feature</u>	<u>Function</u>	<u>Functions Address</u>	<u>SRAM Address</u>
<u>0FFBH</u>	<u>0FFCH</u>	<u>0FFDH</u>				<u>0FFEH - 0FFFH</u>
			I <sup>2</sup> C Byte Transfer	Initialize the I <sup>2</sup> C at 400kHz	0FC25H	

Slave Address	Register		Initialize the I <sup>2</sup> C at 100kHz	0FC28H	
Slave Address	Register	Data	Read registered byte	0FC52H	Result/Error
Slave Address		Data	Send register byte	0FC8FH	Result/Error
Slave Address			Send Byte	0FCBDH	Result/Error
Slave Address			Read Byte	0FCE2H	Result/Error
			Disable the I <sup>2</sup> C	0FC3CH	
			Utilities Version Number	0FC1EH	Result

## 4.0 Micro64/128 Software

When it comes from the factory, the Micro64/128 has no software on the board itself. Programs are developed using cross-development tools running on a desktop PC and is

programmed into the Micro64/128 for execution. There are several development environments from which to choose.

### 4.1 Assembly

Any cross assembler capable of creating programs for Atmel's Atmega64 or Atmega128 microcontroller can be used to write assembly language programs for the Micro64/128. Atmel's assembler is available for free by downloading it at [www.atmel.com/products/avr/](http://www.atmel.com/products/avr/)

Other cross assemblers may be found by looking under "Third Party Support" on the website above.

### 4.2 BASCOM-AVR BASIC Compiler

BASCOM-AVR has a complete Windows IDE (Integrated Development Environment) with a Terminal Emulator. It is a structured BASIC with labels, that have statements highly compatible with Microsoft's Quick BASIC and Visual BASIC. BASCOM AVR supports IF-THEN-ELSE-END IF, DO-LOOP, WHILE-WEND, SELECT- CASE and in line assembly. It has a large set of Trig Floating point functions. Variables and labels can be 32 characters. Bit, Byte, Integer, Word, Long, Single, and Strings are supported for variable types.

A bit is 1/8 of a byte and can only be a 1 or a 0. A byte is stored as an unsigned 8-bit binary number ranging in value

from 0 to 255. An integer is stored as a signed sixteen-bit binary number ranging in value from -32,768 to +32,767. A word is stored as an unsigned sixteen-bit binary number ranging in value from 0 to 65535. A long is stored as signed 32-bit binary number ranging in value from -2147483648 to 2147483647. A single is stored as a signed 32 bit binary number. Ranging in value from  $1.5 \times 10^{-45}$  to  $3.4 \times 10^{38}$ . A string is stored as bytes and is terminated with a 0-byte. A string dimensioned with a length of 10 bytes will occupy 11 bytes of memory. For more information please visit [www.mcselec.com](http://www.mcselec.com).

### 4.3 CodeVisionAVR C Compiler

CodeVisionAVR runs under Windows 95, 98, Me, NT 4.0, 2000 and XP. It is an easy to use integrated development Environment with a built-in serial communication terminal

for debugging and has an editor with auto indentation and keywords highlighting. The C Compiler supports bit, char, int, short, long, and float data types.

# Micro64/128

It also has supplementary libraries for Alphanumeric LCD modules for up to 4x40 characters, Philips I<sup>2</sup>C Bus, National Semiconductor LM75 Temperature Sensor, Dallas DS1621 Thermometer/Thermostat, Philips PCF8563 and PCF8583 Real Time Clocks, Dallas DS1302 and DS1307 Real Time Clocks, Dallas 1 Wire protocol, Dallas DS1820/DS1822 1 Wire Temperature sensors, Dallas DS2430/DS2433 1 Wire EEPROMs, SPI, Power management, Delays, BCD and Gray code conversion.

The compiler comes with a built-in CodeWizardAVR Automatic Program Generator, that allows you to write in a matter of minutes all the code needed for implementing the following functions: External memory access setup, Chip

reset source identification, Input/Output Port initialization, External Interrupts initialization, Timers/Counters initialization, Watchdog Timer initialization, UART initialization and interrupt driven buffered serial communication with the following parameters: 7N2, 7E1, 7O1, 8N1, 8N2, 8E1 and 8O1, Analog Comparator initialization, ADC initialization, SPI Interface initialization, I<sup>2</sup>C Bus, LM75 Temperature Sensor, DS1621 Thermometer/Thermostat, PCF8563, PCF8583, DS1302 and DS1307 Real Time Clocks initialization, 1 Wire Bus and DS1820/DS1822 Temperature Sensors initialization, and LCD module initialization. For further information please visit <http://www.hpinfofotech.ro>.

## 5.0 Programming the Micro64/128

The user-programmable part of the Micro64/128 uses Atmel's Atmega64 or Atmega128 FLASH micro-controller that can be reprogrammed thousands of times. These programs can be created using a number of resources, as described above.

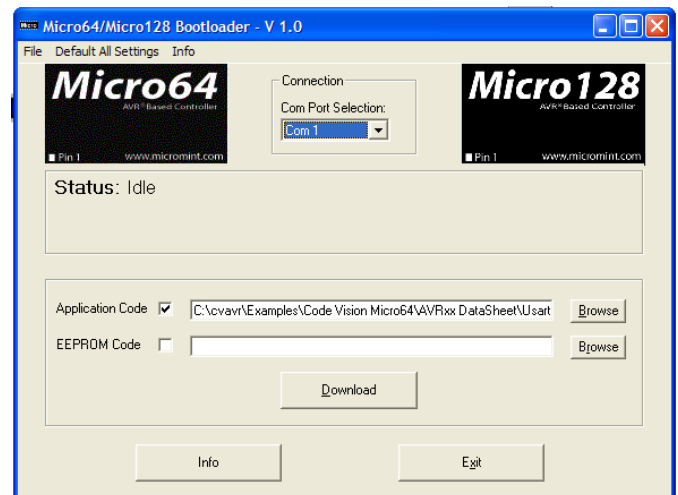
Programming the Micro64/128 is done through USART1, the RS232a/RS422/RS485 serial port, by using the Micro64/128 Boot Loader IDE. The Micro64/128 **cannot** be programmed over a RS485 network.

## 5.1 Using the Boot Loader IDE

The Boot Loader IDE comes free with the Micro64 and the Micro128. It is used to program a program and data EEPROM files into the module. It is very easy to use. There are only six steps to send a program to the Micro64/128.

### Step

1. Open the Boot Loader IDE.
2. Select the appropriate COM Port.
3. Click on the top "Browse" button to find the program you want to send to the Micro64/128.
4. **Optional Step:** Click on the bottom "Browse" button to find the EEPROM file you want to send to the Micro64/128.
5. Click on the "Download" button to send the files.
6. Apply power to the board or reset the module.



## 5.2 Boot Loader Programming Protocol

After the Boot Loader is started (via a reset or a power-up), the following protocol must be observed:

1. Upon power-up or reset Boot Loader sends a '^' (BOOTLOADER\_ACTIVE\_CHAR) at 115,200 bits per second (bps) using X-ON/X-OFF handshaking.
2. The host is then required to send the three-character entry sequence of '@&\$'. This is used to prevent an

inadvertent attempt of reprogramming from taking place. If the Boot Loader does not receive these characters within the timeout period of 5 seconds, the Boot Loader tests to see if there is code located in the main application area of flash. If there is, then the Boot Loader jumps to it, otherwise, execution stays within the Boot Loader indefinitely, waiting for the entry sequence.

3. Once the three-character entry sequence has been sent, the Boot Loader sends the version string (Vx.xx) followed by a '?' (READY\_CHAR).
4. Upon receipt of the READY\_CHAR, the host application should send the hex file for the new/updated application program observing an X-ON / X-OFF handshaking protocol to control data flow. The handshaking is very important as the flash

memory area writes much more slowly than the serial port can send data. The programming software continues sending the hex file until it is all sent.

After each line of ".hex" file is received by the Boot Loader, one of three characters is transmitted by the Boot Loader:

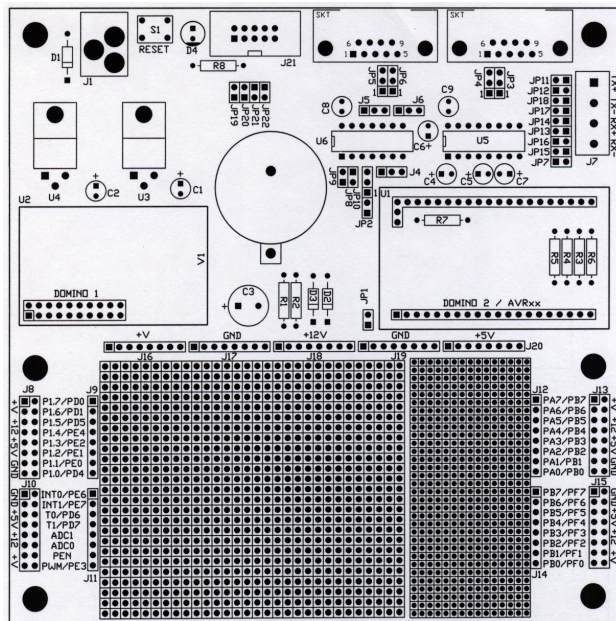
- '~' Line received with no errors.
  - '%' Line received with no error, but an error occurred while flashing.
  - '-' Checksum error detected while receiving the line.
5. After the programming is complete, the Boot Loader sends either a '#', meaning the programming is all right, or and '@' indicating that an error has occurred and the program did not load successfully. In most cases an error during programming means that the main application program is corrupted and will need to be resent.
  6. The AVRBL then starts the newly programmed application software. As stated in step 2, the AVRBL tests to see if there is code located in the main application area of flash. If there is, the AVRBL jumps to it, otherwise, execution stays within the AVRBL indefinitely, waiting for the entry sequence

---

## 6.0 Micro64/128 Development Board

The Micro64/128 Development Board was designed, for use as an evaluation platform, for prototyping additional circuitry around applications using the Micro64/128.

# Micro64/128



## 6.1 Development Board Power Supply

An unregulated 12 VDC wall transformer with a 2.5mm power plug is supplied to power the board. The plugs center tap should be negative. A diode (D1) will protect the regulator if a power supply with the wrong polarity is accidentally plugged in. The development board can power

the Micro64/128 with a regulated +12VDC by placing a jumper on JP1 or with a regulated +5V by placing a jumper on JP2. **NOTE: Inserting jumpers on both JP1 and JP2 will damage the module.**

## 6.2 Jumper Descriptions

There are twenty-two jumpers on the development board. Figure 4-1 list each jumper and what they are connected to.

JP1	<ul style="list-style-type: none"> <li>■ VOUT of 7812</li> <li>● +V of Micro64/128 (pin 1)</li> </ul>	<ul style="list-style-type: none"> <li>● R2IN of U5 (pin 8)</li> <li>■ Pin 3 of J3 (DB9)</li> </ul>	JP11	<ul style="list-style-type: none"> <li>■ PED of Micro64/128 (pin 10)</li> <li>● R2OUT of U5 (pin 9)</li> </ul>	JP17	<ul style="list-style-type: none"> <li>■ Pull-down Resistor</li> <li>● TX+ of Micro64/128 (pin 39)</li> </ul>		
JP2	<ul style="list-style-type: none"> <li>■ VOUT of 7805</li> <li>● +5 of Micro64/128 (pin 40)</li> </ul>	<ul style="list-style-type: none"> <li>1 ● T1OUT of U5 (pin 14)</li> <li>■ T2OUT of U5 (pin 7)</li> </ul>	JP6	<ul style="list-style-type: none"> <li>● RX- of Micro64/128 (pin 36)</li> <li>● +2.5V</li> </ul>	JP12	<ul style="list-style-type: none"> <li>● TX+ of Micro64/128 (pin 39)</li> <li>● T1IN of U5 (pin 11)</li> </ul>	JP18	<ul style="list-style-type: none"> <li>● Termination Resistor</li> <li>■ TX+ of Micro64/128 (pin 39)</li> </ul>
JP3	<ul style="list-style-type: none"> <li>● T1OUT of U5 (pin 14)</li> <li>■ Pin 3 of J2 (DB9)</li> </ul>	<ul style="list-style-type: none"> <li>1 ● R1IN of U5 (pin 13)</li> <li>■ RX- of Micro64/128 (pin 36)</li> </ul>	JP7	<ul style="list-style-type: none"> <li>■ 1INB of U6 (pin 2) of AND gate input</li> <li>● PD0 of Micro64/128 (pin 4)</li> </ul>	JP13	<ul style="list-style-type: none"> <li>● RX+ of Micro64/128 (pin 37)</li> <li>■ R1OUT of U5 (pin 12)</li> </ul>	JP19	<ul style="list-style-type: none"> <li>● J21 pin 1</li> <li>■ PED of Micro64/128 (pin 10)</li> </ul>
JP4	<ul style="list-style-type: none"> <li>● T1OUT of U5 (pin 14)</li> <li>■ Pin 2 of J2 (DB9)</li> </ul>	<ul style="list-style-type: none"> <li>1 ● R1IN of U5 (pin 13)</li> <li>■ 1INA of U6 (pin 1) AND gate input</li> </ul>	JP8	<ul style="list-style-type: none"> <li>● PED of Micro64/128 (pin 10)</li> <li>■ 1OUT of U6 (pin 3) AND gate output</li> </ul>	JP14	<ul style="list-style-type: none"> <li>■ TX+ of Micro64/128 (pin 39)</li> <li>● RX+ of Micro64/128 (pin 37)</li> </ul>	JP20	<ul style="list-style-type: none"> <li>■ J21 pin 3</li> <li>● RESET of Micro64/128 (pin 42)</li> </ul>
JP5	<ul style="list-style-type: none"> <li>● R2IN of U5 (pin 8)</li> <li>■ Pin 2 of J3 (DB9)</li> </ul>	<ul style="list-style-type: none"> <li>1 ● T2OUT of U5 (pin 7)</li> <li>■ 1OUT of U6 (pin 3) AND gate output</li> </ul>	JP9	<ul style="list-style-type: none"> <li>● PED of Micro64/128 (pin 10)</li> <li>■ T2IN of U5 (pin 10)</li> </ul>	JP15	<ul style="list-style-type: none"> <li>■ TX- of Micro64/128 (pin 38)</li> <li>● RX- of Micro64/128 (pin 36)</li> </ul>	JP21	<ul style="list-style-type: none"> <li>■ J21 pin 5</li> <li>● PB0 of Micro64/128 (pin 19)</li> </ul>
JP6	<ul style="list-style-type: none"> <li>1 ● T2OUT of U5 (pin 7)</li> <li>■ PE1 of Micro64/128 (pin 9)</li> </ul>		JP10		JP16	<ul style="list-style-type: none"> <li>■ Pull-up Resistor</li> <li>● TX- of Micro64/128 (pin 38)</li> </ul>	JP22	<ul style="list-style-type: none"> <li>■ J21 pin 4</li> <li>● PE1 of Micro64/128 (pin 9)</li> </ul>

Figure 4-1 Development Board Jumper Descriptions

## 6.3 Serial Communication

The development board can be configured to have the Micro64/128's USART1 communicate in RS-232, RS-422,

and RS-485. It also has the ability to configure USART0 to communicate in RS-232.

### 6.3.1 USART1 as RS-232.

In order to use USART1 for RS-232 communications the user must take into account the following jumpers JP3, JP4, JP7, JP12, and JP13. The RS-232 connection is made using a DB-9M and connecting it to the development boards, DB-9F, J2 connector. JP12 connects TX+ (pin 39 of Micro64/128) to the RS-232 driver chips T1IN signal (pin 11 of U5). JP13 connects RX+ (pin 37 of Micro64/128) to the RS-232 driver chips R1OUT signal (pin 12 of U5). Since the Micro64/128 uses a differential receiver, 2.5 VDC must be connected to RX- (pin 36 of Micro64/128). By placing a jumper on JP7, 2.5VDC is applied to RX-. JP3 is used to connect either the receiving or transmitting signal to pin 3 of J2's DB-9F. JP4 is used to connect either the receiving or transmitting signal to pin 2 of J2's DB-9F. If the user were going to communicate to a Personal Computer they would need to put jumpers on JP7, JP12, JP13, JP3 pins 1&2, and JP4 pins 2&3. Figure 4-2 illustrates what the jumpers might look like.

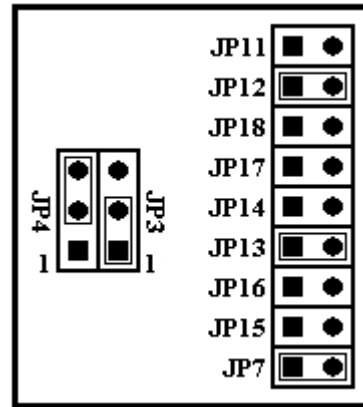


Figure 4-2 USART1 Jumper Configurations for Connecting to a PC

### 6.3.2 USART1 as RS-422

RS-422 communications require two twisted pairs. One pair connects the console transmitter to Micro64/128's receiver while the second pair connects the console receiver to Micro64/128's transmitter.

RS-422 uses two unidirectional data paths – one path for each direction. The data transmission is differential, enabling the noise picked-up on the pairs to cancel itself out. Each twisted pair should have termination enabled at each end of the line. Pull-up and pull-down termination may be required, but only at one end of each pair.

### 6.3.3 USART1 as RS-485

RS-485 communication requires one twisted pair to connect the console to Micro64/128. RS-485 uses one data path, so the drives at each end must NOT be enabled at the same time. The user is responsible for NOT breaking this rule.

The easiest protocol to follow is a master/slave(s) relationship, were the slaves DO NOT enable their transmitter (respond) unless the master asks them to. The data transmission is differential allowing picked up noise to cancel itself out. The twisted pair should have termination enabled at each end of the line. Pull-up and pull-down termination may be required, but only at one end of the pair.

The only jumpers that are associated with RS-422 communications are JP16, JP17, and JP18 if termination is necessary. The twisted wire pairs should be connected to the screw terminals J7. J7 pin 1 is for TX+, pin 2 is for TX- , pin 3 is for RX+ and pin 4 is for RX-. JP16, JP17 and JP18 are the jumpers used for terminating the network. Since termination is needed to match the impedance of a node to the impedance of the transmission line the termination (R7), pull-up (R5) and pull-down (R6) resistors are not populated by the factory.

The only jumpers that are associated with RS-485 communications are JP14, JP15, JP16, JP17, and JP18 if termination is necessary. JP14 is used to connect the positive side (TX+ & RX+) of the network together and JP15 is used to connect the negative (TX- & RX-) side together. The twisted wire pair should be connected to the screw terminal J7 pin 1 or 3 for the positive side and pin 2 or 4 for negative side. JP16, JP17 and JP18 are the jumpers used for terminating the network. Since termination is needed to match the impedance of a node to the impedance of the transmission line the termination (R7), pull-up (R5) and pull-down (R6) resistors are not populated by the factory.

### 6.3.4 USART0 as RS-232

# Micro64/128

In order to use USART0 for RS-232 communications the user must take into account the following jumpers JP5, JP6, JP10, and JP11. The RS-232 connection is made using a DB-9M and connecting it to the development boards, DB-9F, J3 connector. JP10 connects PE1 (USART0's transmit pin and pin 9 of Micro64/128) to the RS-232 driver chips T2IN signal (pin 10 of U5). JP13 connects PE0 (USART0's receive pin and pin 10 of Micro64/128) to the RS-232 driver chips R2OUT signal (pin 9 of U5). JP6 is used to connect either the receiving or transmitting signal to pin 3 of J3's DB-9F. JP5 is used to connect either the receiving or transmitting signal to pin 2 of J3's DB-9F. If the user were going to communicate to a Personal Computer they would need to put jumpers on JP7, JP12, JP13, JP3 pins 1&2, and JP4 pins 2&3. Figure 4-3 illustrates what the jumpers might look like.

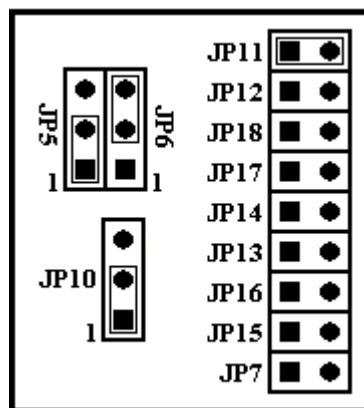


Figure 4-3 USART0 Jumper Configurations for Connecting to a PC

## 6.4 Micro64/128 Connections

All of the Micro64/128's signals except for VBAT, \*RESET, TX+, TX-, RX+, and RX- are brought out to four 2x8 headers (J8, J10, J13, & J15) and four 1x8 plated through solder holes (J9, J11, J12, & J14). Please refer to figure 8 for their pin out. J16, J17, J18, J19 and J20 are connected to the development boards power supply and can be used to power your external circuitry and are located above the prototyping area. VBAT can be connected to either a coin cell battery or a Super Capacitor. If the user chooses to backup the real-time clock with a battery, a coin cell battery holder (V1) and a 0 ohm resistor (D2) would have to be installed on the board. The 0 ohm resistor instead of a diode would need to be installed because the Micro64/128 already has a diode built in. To back-up the real-time clock with a super capacitor, two 1k resistors need to be installed in R1 and R2, a BAT85 Diode needs to be installed in D3 and the 1 Farad Super Capacitor needs to be installed in C3. The RESET pin for Micro64/128 is connected to a pushbutton located near the power connector. The user can press and release the pushbutton to reset the Micro64/128.

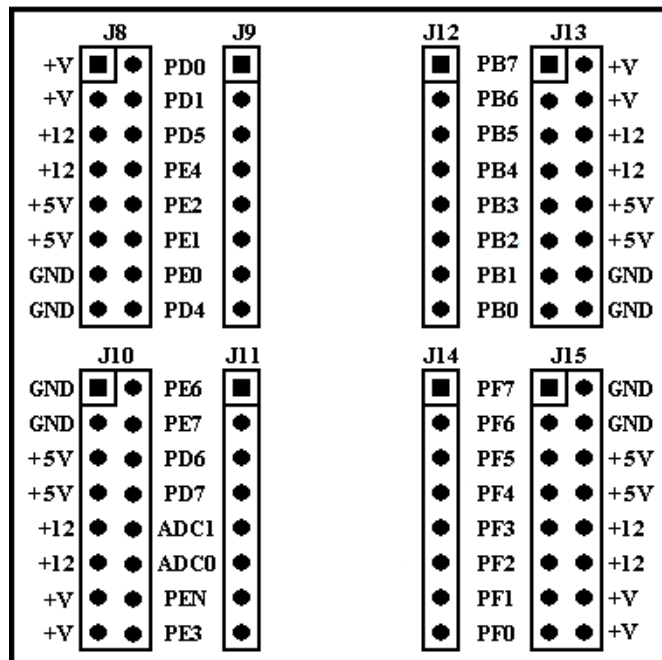
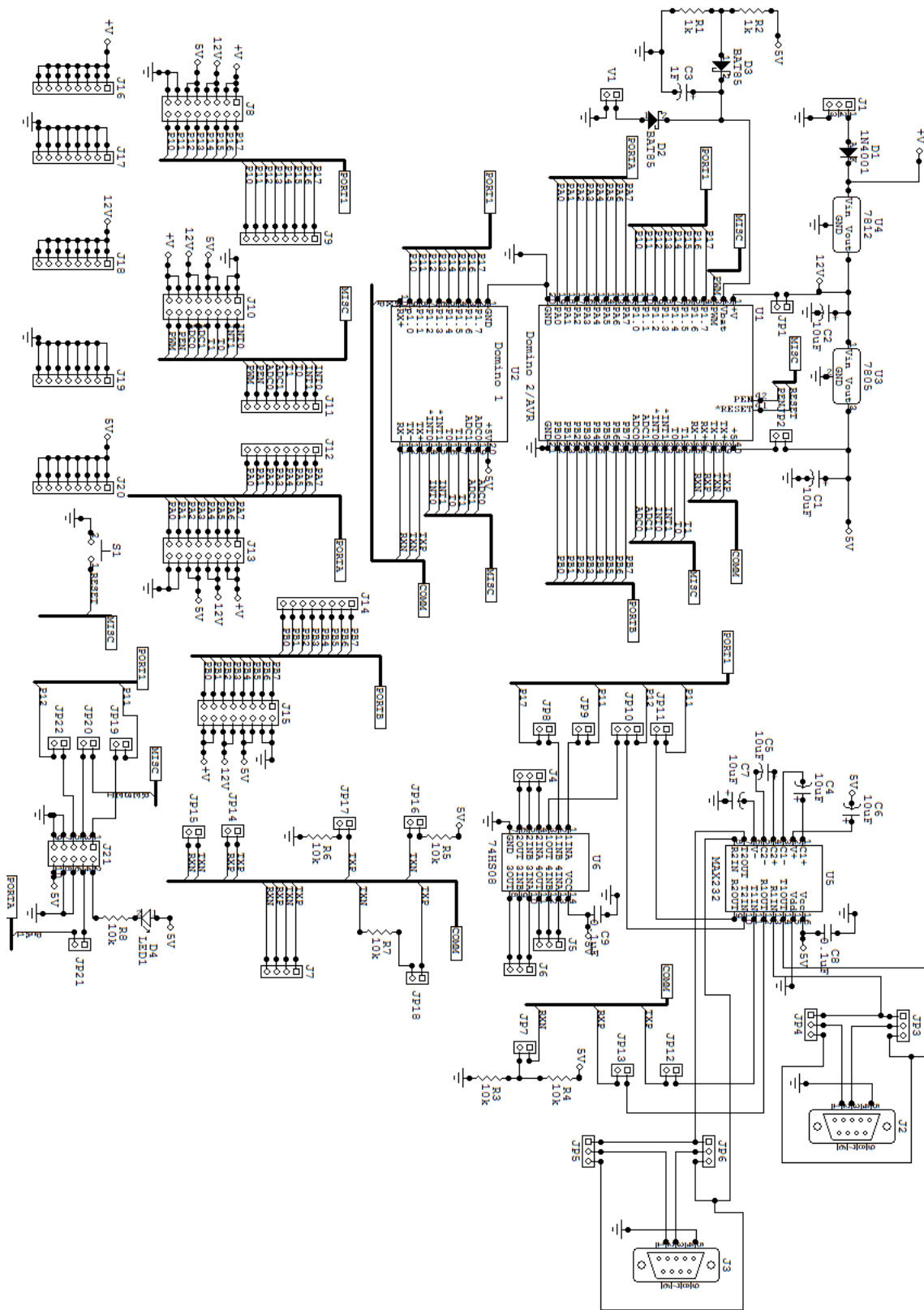


Figure 4-4 Micro64/128 Connections

## 6.5 Development Board Schematic



# Micro64/128

## Appendix 1.0

### 1.1 Sample Application: Communications

Micro64/128 can communicate with other serial devices at up to 230.4 kbps. Usart1 can be connected in one of three configurations: RS-232A, RS422, and RS-485. Micro64/128's RS-232A output can be used with most full duplex PC-type serial devices which normally handle RS-232C provided they can reconcile receiving the lower-voltage transmit level of RS-232A. This three wire (Tx/Rx/GND) using the RS-422 input receivers as simple level-shifting inverters as shown in Figure 1 creates RS-232A connection. If the user needs the full voltage levels of RS-232C an additional chip is need. This is demonstrated in Figure 2.

RS-422 is an alternate full-duplex connection which uses two twisted-pair transmission lines (i.e., Tx+/Tx-/Rx+/Rx-) offering long transmission paths and noise-canceling techniques. This distance is typically 4000 feet. This connection is shown in Figure 3. RS-485 is similar to RS-422 with the exception that it uses a single twisted pair in a half-duplex arrangement (i.e., +/-). This means data transmissions must use the same twisted-pair path to travel in both directions, requiring a simple protocol of only one unit seizing the transmission pair at a time while all others listen. This connection is shown in Figure 4.

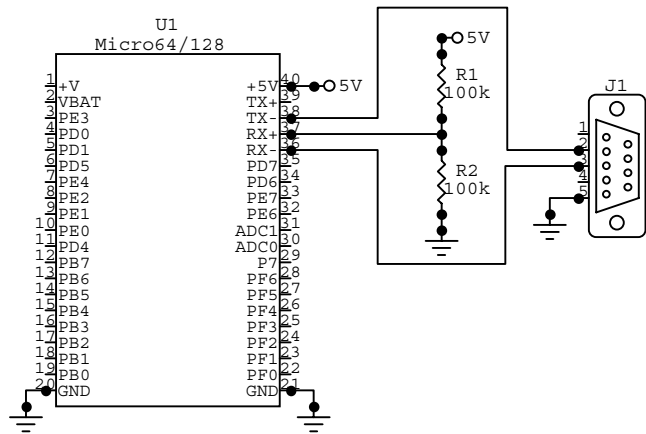


Figure 1: Typical RS-232A Connections

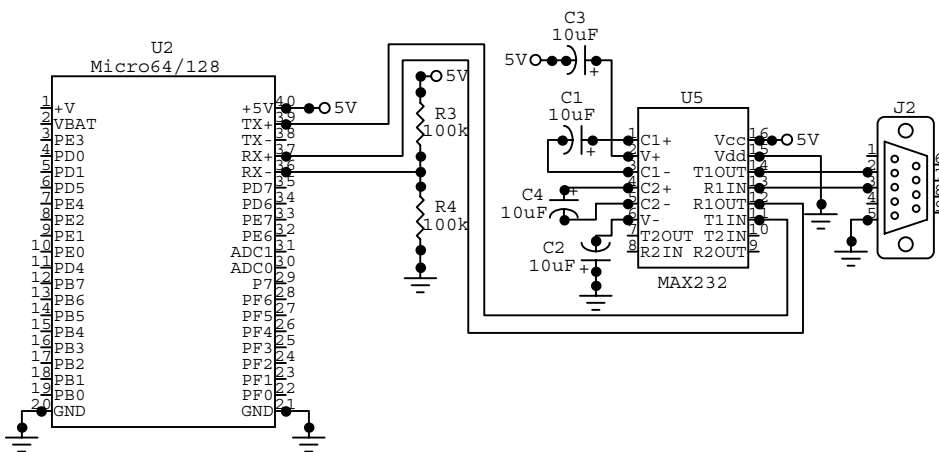


Figure 2: Typical RS-232C Connections

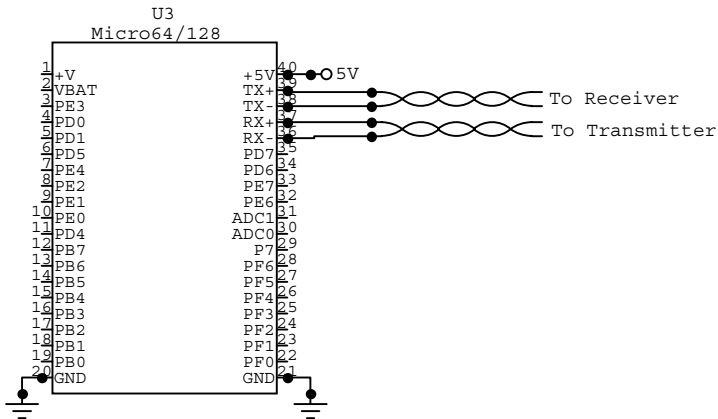


Figure 3: Typical RS-422 Connections

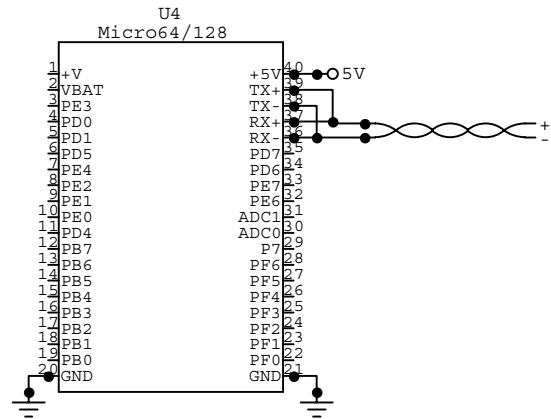


Figure 4: Typical RS-485 Connection

## 1.2 Sample Application: Networking Micro64/128

Multiple Micro64/128s can be used in a networked multi-drop configuration. Network protocol requires that only one unit is allowed to transmit on the line at a time. All other units are listening in receive mode.

This is accomplished by requiring one Micro64/128 or a device like a PC to be the net master. The master talks to any slave unit either passing information to it or requesting information to

it or requesting information from it. The slaves must never answer the master until a response is requested. The master then relinquishes the net to that slave for the response and regains the net when the slave is finished. This arrangement enables multiple controllers to work together gathering numerous inputs and controlling innumerable outputs, independent of the system size. Figure 5 demonstrates a network.

# Micro64/128

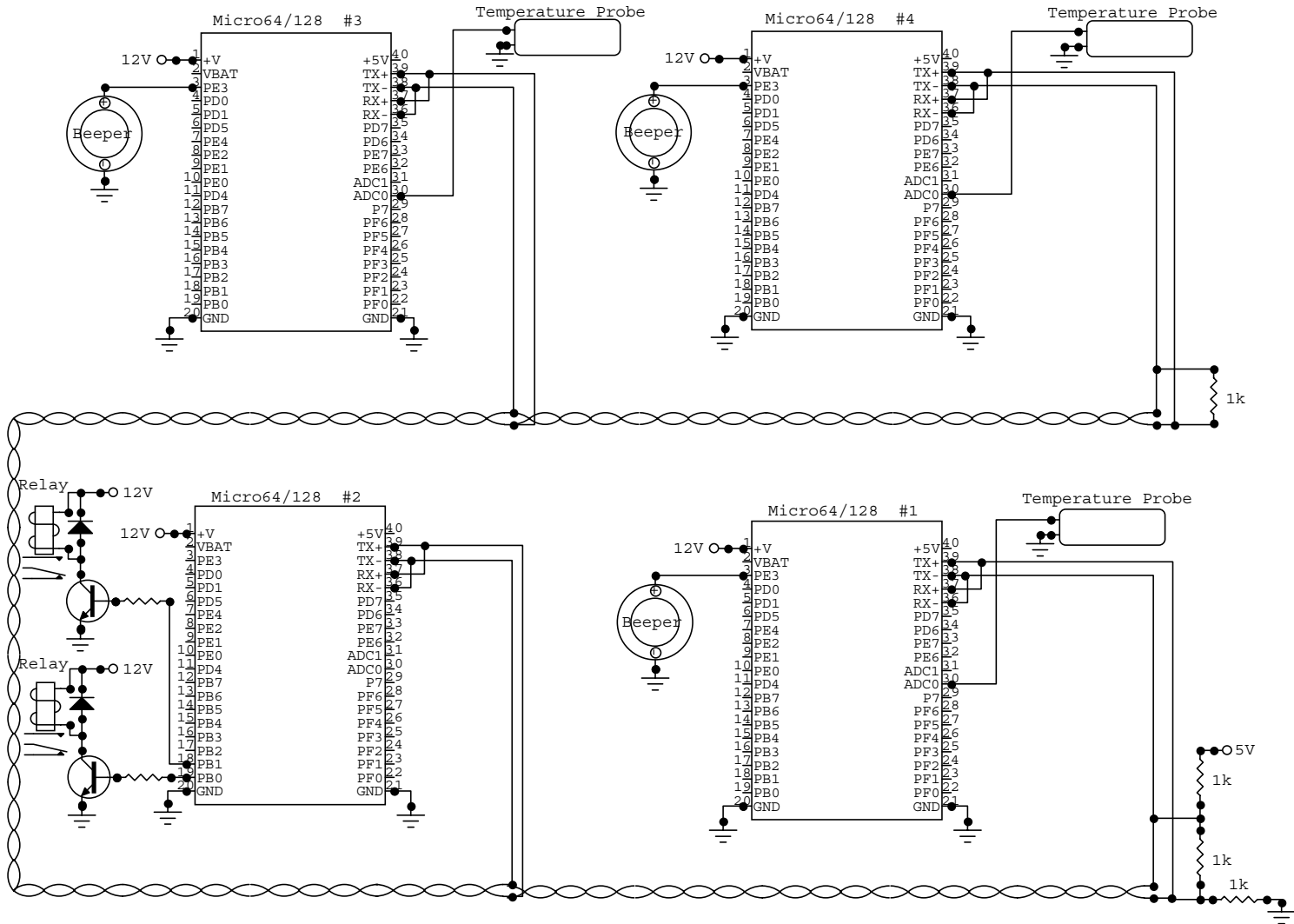


Figure 5: RS-485 Network