



Modtronix Engineering

Modular Electronic Solutions

Modtronix Engineering
55 Tipuana Drive
Elanora Qld 4221
Australia
Tel: +61 (0)7 31142536
Fax: +61 (0)7 31026298
ABN: 48 663 118 043

SBC65EC

Ethernet enabled Single Board Computer

Table of Contents

1 Introduction.....	2
2 Features.....	3
3 Web Server and Updating Firmware.....	3
4 Daughter Board Connectors.....	4
4.1 Analog Inputs.....	5
4.2 PWM Outputs.....	5
4.3 USART2.....	5
4.4 As a Daughter Board.....	5
4.5 Expansion boards.....	5
5 Interfaces.....	6
5.1 Ethernet.....	6
5.2 RS232.....	6
5.3 ICSP connector.....	6
5.4 Analog and Digital I/O pins.....	6
6 Configuration.....	6
7 External Memory.....	6
7.1 EEPROM.....	6
7.2 FRAM.....	7
8 Commands.....	7
8.1 Introduction.....	7
8.2 HTTP GET Commands.....	7
8.3 UDP Commands.....	7
8.4 Defined Commands.....	7
9 Tags.....	10
9.1 Introduction.....	10
9.2 Dynamic Web Pages.....	10
9.3 Tags via UDP command.....	11
9.4 Tags via HTTP GET method.....	11
9.5 Defined Tags.....	11
10 Specifications.....	15
10.1 Absolute Maximum Ratings.....	15
10.2 Electrical Characteristics.....	15
10.3 D.C. Characteristics of user I/O pins on Daughter Board connector.....	15
11 Dimensions.....	16

2 Features

- Has 32 general purpose user programmable I/O ports that can be monitored, configured and updated via web pages, HTTP and UDP. Twelve inputs can be configured as 10 bit Analog Inputs, and 4 as 10-bit PWM outputs.
- Programmed with Modtronix SBC65EC Web Server, for details [click here](#).
- Can be configured and controlled via web pages, for details [click here](#).
- Accepts commands via UDP or HTTP for configuring the board and controlling outputs, for details see modtronix.com/products/sbc65ec/doc/commands
- Implements tags that can be placed on web pages to display dynamic information and current input status, for details see modtronix.com/products/sbc65ec/doc/tags
- Programmed with a bootloader for updating firmware over the network or internet.
- Is part of our MicroX product range, meaning you can upgrade or expand it with any of the other MixroX products. For details see modtronix.com/microx
- Can be used as a daughter board to Ethernet enable any product.
- Diode protected 2.1mm power connector for standard DC transformer. Center is positive.
- 98KBytes FLASH, 3840 Bytes SRAM and pluggable EEPROM. Default TCP/IP stack uses less than half the available memory, which leaved heaps of code space for custom code.
- Has space for a 8 pin Ramtron SPI FRAM chip (32Kbyte FM25256 chip for example) to be assembled on the bottom of the board.
- Wide operating voltage range from 7 – 30V.
- Default operating frequency of 40MHz, software configurable low power mode that runs at 10MHz.
- Red 3mm User programmable LED.
- RJ45 connector with built in LEDs to indicate link and activity status.
- Micro Match connector for connecting a LCD2S serial LCD display with keypad decoder. For details on LCD2S range of LCD displays, see modtronix.com/products/lcd2s
- RS232 interface via 3 pin Molex type connector or Daughter Board connector.
- Has a 40 pin Daughter Board connector. For details see modtronix.com/microx/expansion.
- Assembled with brand name, quality components. For example, electrolytic capacitors used are extra long life rated (5 times more than standard), Industrial rated semiconductors (not commercial).
- Has an ICSP (In Circuit Serial Programming) connector (ICPC1 type) - CPU can be programmed and debugged in circuit. For details see modtronix.com/picboards/prog.
- Programmed with free Modtronix TCP/IP stack that features:
 - Includes MAC, IP, ARP, ICMP, TCP, UDP, HTTP, HTTP Compression, FTP, DHCP, NetBIOS, DNS, MXFS
 - Socket support for TCP and UDP
 - Portable across PIC18 MCUs
 - Out-of-box support for Microchip MPLAB C18 and Hi-Tech PICC-18 compilers
 - RTOS independent
 - Full TCP state machine
 - Modular Design

3 Web Server and Updating Firmware

At delivery, the SBC65EC is programmed with the *Modtronix SBC65EC Web Server*, and a bootloader. When powered, the SBC65EC will remain in bootloader mode for 3 second, after which the *Modtronix SBC65EC Web Server* will start up.

The *Modtronix SBC65EC Web Server* is a web based interface that allows the SBC65EC to be configured, controlled and monitored via web pages. For details, see modtronix.com/products/sbc65ec/doc/webpages.

The firmware on the SBC65EC can be updated via the Bootloader. To upload the firmware, the *Modtronix Network Bootloader* PC application is required. It can be downloaded for free from modtronix.com/soft/netloader. The bootloader is contained in a protected area of the SBC65EC's program memory, and can not be accidentally erased when updating the firmware. For details on

The board's IP address however can be configured, and is also used by the bootloader. If this gets corrupted, it is possible that the bootloader will not work anymore. In this case, the board can be started in Safemode. To enter Safemode, the jumper on the CON3 pin header must be moved from its default position (over pins 7 and 8) to pins 1 and 2. See pictures on the right. In safemode default values from the protected area of the program memory are used.

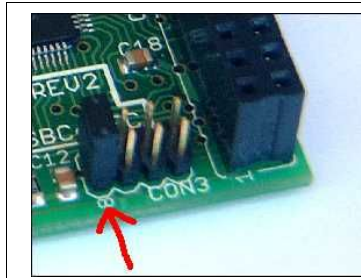


Figure 2 - Normal Mode

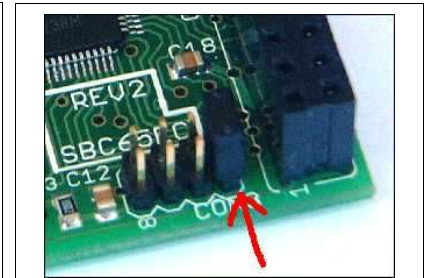


Figure 3 - Safemode

4 Daughter Board Connectors

The SBC65EC has two 2x10 pin, 2.54mm female connectors. They contain all free CPU port pins, power, I2C, SPI, RS232 signal,..... For the location of the Daughter Board connectors, see the *Dimensions* chapter of this document. The Daughter Board connectors pins are mapped to the following signals:

<i>CON2 Daughter Board Connector</i>			<i>CON1 Daughter Board Connector</i>		
<i>Name</i>	<i>Pin</i>	<i>Signal</i>	<i>Name</i>	<i>Pin</i>	<i>Signal</i>
T0	2	PIC pin RF0	T4	2	PIC pin RF4
T1	1	PIC pin RF1	T5	1	PIC pin RF5
T2	4	PIC pin RF2	T6	4	PIC pin RF6
T3	3	PIC pin RF3	T7	3	PIC pin RF7
SIG0	6	USART receive signal - RS232 or TTL signal level. ⁽³⁾	GND	5	Ground
SIG1	5	USART transmit signal - RS232 or TTL signal levels. ⁽³⁾	+5V	7	Regulated 0.5A 5V supply
B0	13	PIC pin RB0	VIN	8	Unregulated input voltage
B1	14	PIC pin RB1	CLR#	6	PIC pin /MCLR
B2	11	PIC pin RB2	A0	10	PIC pin RA0
B3	12	PIC pin RB3	A1	9	PIC pin RA1
B4	9	PIC pin RB4	A2	12	PIC pin RA2
B5	10	PIC pin RB5	A3	11	PIC pin RA3
B6	7	PIC pin RB6 – also used for ICP ⁽¹⁾	A4	14	PIC pin RA4
B7	8	PIC pin RB7 – also used for ICP ⁽¹⁾	A5	13	PIC pin RA5
C4	17	PIC pin RC4 – port pin assigned for I ² C ⁽²⁾	C0	16	PIC pin RC0
C5	18	PIC pin RC5	C1	15	PIC pin RC1
C6	15	PIC pin RC6	C2	18	PIC pin RC2
C7	16	PIC pin RC7	C3	17	PIC pin RC3 – port pin assigned for I ² C ⁽²⁾
D6	19	RG2 – USART2 Receive	D0	20	RG0
D7	20	RG3	D1	19	RG1 – USART2 Transmit

(1) Port Pins B6 and B7 are also used for in circuit programming, if the board is programmed in circuit! If they are used, and the board should still be in circuit programmable, make sure their impedance is greater than a 1000 ohms!

(2) Port Pins C3 and C4 are assigned to be used as I²C pins in Daughter and Frontend Boards.

(3) At delivery SIG0 and SIG1 pins are configured for RS232 voltage levels. See RS232 section in documentation for details

4.1 Analog Inputs

The following 10 Daughter Board Connector pins can be configured via the Web interface to be 10-bit Analog Inputs: A0 (CPU Signal RA0), A1 (CPU Signal RA1), A2 (CPU Signal RA2), A3 (CPU Signal RA3), A5 (CPU Signal RA5), T0 (CPU Signal RF0), T1 (CPU Signal RF1), T2 (CPU Signal RF2), T3 (CPU Signal RF3), T4 (CPU Signal RF4), T5 (CPU Signal RF5), T6 (CPU Signal RF6)

When configured to be Analog Inputs, the user must ensure to configure them as Input pins.

4.2 PWM Outputs

The following 4 Daughter Board Connector pins can be configured via the Web interface to be 10-bit PWM (Pulse Width Modulated) Outputs:

C1 (CPU Signal RC1), C2 (CPU Signal RC2), D0 (CPU Signal RG0), D7 (CPU Signal RG3)

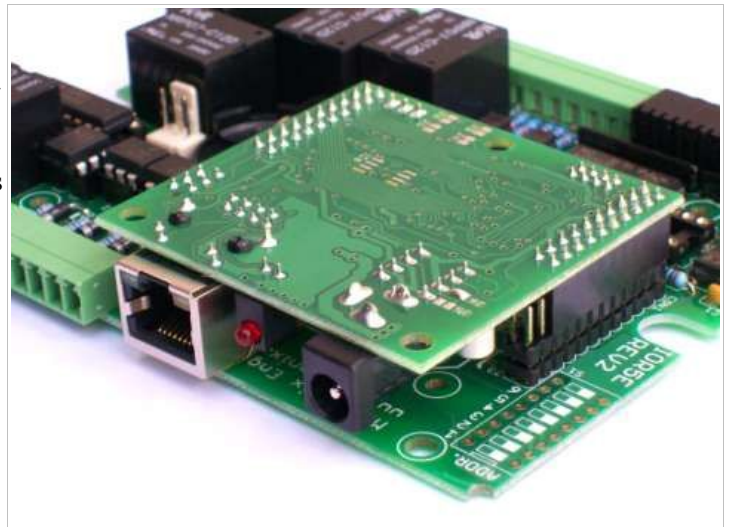
When configured to be PWM Outputs, the user must ensure to configure them as Output pins.

4.3 USART2

Daughter Board Connector D1 (CPU Signal RG1) and D6 (CPU Signal RG2) can be configured for USART2 Transmit and Receive. USART2 can be enabled via the web page interface, and configured for common serial port BAUD rates. It can also be linked to a configurable UDP Port, the I²C Master port or USART1 via the web interfaces. By doing this, it is possible implementing a Serial to UDP, Serial to I²C or Serial to Serial interface converter.

4.4 As a Daughter Board

The SBC65EC can be used to add Ethernet capabilities to any board by using it as a daughter board. All connectors and PCB standoffs required to do this can be purchased from our web site. The board that is to take the SBC65EC as a daughter board needs to provide two 2x10 pin, 2.54mm pin headers for the SBC65EC to plug into. Additionally, 3 PCB supports can also be provided if additional stability is required. Usually it is adequate providing the connectors on two PCB supports as shown in the picture to the right. The 2x10 pin header connectors available from Modtronix are specially made so that when mated with the connectors on the SBC65EC, the main board and SBC65EC will be 15.9mm apart. This is a standard height for PCB supports.



4.5 Expansion boards

The SBC65EC can be used as a full functional Single Board Computer. It's Daughter Board connector can be used as an expansion port to add additional functionality. It contains all free CPU port pins, power, I²C, SPI, RS232 signal,..... For a list of available daughter boards from our site see www.modtronix.com/products/sbc65ec/#expansion. The Picture to the right shows the SBC65EC with a PT01TC prototype daughter board plugged onto it.

Additionally, users can download PCB templates for creating their own Daughter Board from our Download page – see www.modtronix.com/downloads. The *daughter_compact.brd* PCB fits onto the SBC65EC.

5 Interfaces

5.1 Ethernet

The SBC65EC has a 10Mbps Ethernet port. The RJ45 connector meets IEEE 802.3 standards and FCC mechanical requirements. The RJ45 connector has two built in LEDs, a green LED for link indication, and a yellow LED for activity.



5.2 RS232

The SBC65EC has a USART interface with +/- 15kV ESD protection. The USART signals are available via a 3 pin Molex type connector or the Daughter Board connector. Four solder jumpers (SJ1 to SJ4) on the back of the board are used to configure if the USART signals are RS232 or TTL voltage levels – see circuit diagram at end of document for details.

At delivery solder jumpers SJ3 and SJ4 are made, which configures the USART signals for RS232 voltage levels. By making solder jumpers SJ1 and SJ2, and opening SJ3 and SJ4, the USART pins can be configured for TTL signal levels.

5.3 ICSP connector

The SBC65EC has an ICSP (In Circuit Serial Programming) connector (ICPC1 type). This enables the PIC to be programmed and debugged in circuit. For details on programming and debugging in circuit see <http://www.modtronix.com/picboards/prog>.

5.4 Analog and Digital I/O pins

The SBC65EC has 32 I/O pins available for general purpose user I/O. Each of these pins can be configured separately to be inputs or outputs. Digital inputs and outputs are 0 to 5V. Inputs are 3V tolerant, and outputs can be made 3V tolerant by adding a series resistor (assuming 3V input will have clamping diodes).

The SBC65EC can be configured to have between 1 to 12 analog inputs. Each channel has a 10 bit resolution. Consult the PIC18F6627 data sheet (available from www.microchip.com) for further details.

6 Configuration

The SBC65EC board can be configured via solder jumpers SJ1 to SJ5.

SJ1 to SJ4 are used to select RS232 or TTL signals for the USART – see section on RS232 above for details.

SJ5 is currently not used.

7 External Memory

7.1 EEPROM

The SBC65EC board has a 8 pin IC socket for mounting a serial EEPROM, like the 24LC256 (32Kbytes) or 24LC512 (64 Kbytes) chips.

Depending on the SBC65EC variant, a EEPROM might be fitted. The standard SBC65EC board is fitted with a 24LC256 EEPROM and the PIC programmed with the Modtronix TCP/IP stack (modified Microchip TCP/IP stack) that uses the external EEPROM for storing configuration data and web pages. The 24LC256 has 32Kbytes of non volatile memory, which is large enough for several web pages, including some small pictures. If this is not large enough, a larger 24LC512 chip can be fitted that can hold twice as much data.

7.2 FRAM

There is space on the bottom of the board for mounting a RAMTRON FRAM chip. FRAM offers features consistent with a RAM technology, but is nonvolatile like a ROM technology. FRAM bridges the gap between the two categories and creates something completely new -- a nonvolatile RAM. For further details see www.ramtron.com. Currently the following chips can be used:

- **FM25256** – SPI Interface, 4.0V to 5.5V, 32KBytes, 15MHz, 7mA, Unlimited Read/Write Cycles, 10 Year data retention, SPI Mode 0 & 3
- **FM25640** – SPI Interface, 4.0V to 5.5V, 8KBytes, 5MHz, 3mA, 1 Trillion Read/Writes Cycles, 45 Year data retention, SPI Mode 0 & 3
- Older, smaller memory version of the the SPI FRAM chip can also be used.

See the circuit diagram for details on what PIC port pins are connected to the FRAM chip. Notice that the PIC data lines used are used for the SPI2 bus on new PIC chips like the PIC18F6627 for example. Future versions of the SBC65EC will use this chip!

8 Commands

8.1 Introduction

For an up to date list of all current commands, see the online documentation at: modtronix.com/products/sbc65ec/doc/commands

Commands can be sent to the target board via UDP messages, or the HTTP GET command. All commands follow the HTTP GET syntax of name=value.

For example, to set PIC port pin A2, we can:

- Send the following HTTP GET command to the target board: `http://10.1.0.1/?a2=1`
- Send the following UDP message to the UDP Command Port: `a2=1`

In this example the name part is "a2" and the value part is "1".

8.2 HTTP GET Commands

Any of the commands listed below can be executed on the target by using the HTTP GET command. To issue a HTTP GET command, simply append the command to the web page address, after a '?' character.

For example, to set Port pin A2, we can send the following HTTP GET command to the target board: `http://10.1.0.1/?a2=1`

Multiple commands can be send by seperating each command with a '&' character. For example, to set Port pin A2, and clear Port C0, we can send the following HTTP GET command to the target board: <http://10.1.0.1/?a2=1&c0=0>

8.3 UDP Commands

Any of the commands listed below can be executed on the target by sending them to UDP port 54123. This port is configurable, and can be changed.

For example, to set Port pin A2, we can send the following UDP message to the target board:

```
a2=1
```

Multiple commands can be send by seperating each command with a '&' character. For example, to set Port pin A2, and clear Port C0, we can send the following UDP message to the target board:

```
a2=1&c0=0
```

8.4 Defined Commands

Port Commands

The following Port commands are implemented:

Command Syntax	Description
p=XX	Set Port - set the given PIC port to the given hex value. The given hex value (XX part or command) must be in uppercase! For example: "a=8A" will set PIC port A to 0x8A.
p[0-7]=[1,0]	Set Port Pin - set or clear the given PIC port pin. For example: "a0=0" will clear PIC port A0 (Port A, pin 0) "f7=1" will set PIC port F7 (Port F, pin 7)
px[0-7]=[1,0]	Inverted Set Port Pin - Set or clear the given PIC port pin. The value is inverted! For example: "ax0=0" will set PIC port A0 (Port A, pin 0) "fx7=1" will clear PIC port F7 (Port F, pin 7)
p[0-7]r=[1,0]	Set Port Pin, Remember - - Same as "Set Port Pin" command, except that the settings are saved to the application configuration, and are remembered after power up.
p[0-7]p=[0-9]	Pulse Port Pin, Positive - The given PIC port pin is set high and then low again. The value gives the 'pulse width' specifying for how many micro seconds the pin should be kept high. If 0, the pin is kept high for a minimum time, which is usually between 200ns to 400ns. For example: "a0p=0" will set PIC port A0 high and after 200ns to 400ns low again "f7p=9" will set PIC port F7 high and after 9us low again
p[0-7]n=[0-9]	Pulse Port Pin, Negative - The given PIC port pin is set low and then high again. The value gives the 'pulse width' specifying for how many micro seconds the pin should be kept low. If 0, the pin is kept low for a minimum time, which is usually about 200ns to 400ns. For example: "a0n=0" will set PIC port A0 low and after 200ns to 400ns high again "f7n=5" will set PIC port F7 low and after 5us high again
pc[0-7]=[1,0]	Port Pin Configuration - Configure given PIC port pin as input or output. '0' will configure pin to be an output, and '1' an input. The port direction configured with this command will NOT be remembered after the next power up, use the "Port Pin Configuration, Remember" command to achieve this! Example to use this command: "ac0=0" will configure PIC port A0 (Port A, pin 0) to be an output. "fc7=1" will configure PIC port F7 (Port F, pin 7) to be an input
pc[0-7]r=[1,0]	Port Pin Configuration, Remember - Same as "Port Pin Configuration" command, except that the settings are saved to the application configuration, and are remembered after power up.

Config Commands

The following Config commands are implemented:

Command Syntax	Description
kxx=[0-255]	Set Config Byte to given decimal value - set the given config byte to the given decimal value. The 'xx' part gives the offset (in uppercase hex) of the config byte in the "Application Configuration bytes" structure. The value has to be a decimal number from 0 - 255. Examples to use this command: "k00=100" Will set the the "Application Configuration byte" at 0x00 (MSB of IP address) to 100 "k2A=16" Will set the "Application Configuration byte" at 0x2A (Port F direction) to 16 (0x0f). Address 0x2A contains the TRIS register for PIC Port F. Setting a bit to 0 will configure the port as an output, and 1 as an input. In this example we set TRISF to 0x0f = binary 00001111. This will configure port F pins 0-3 as inputs, and pins 4-7 as outputs. "k2F=16" Will set the "Application Configuration byte" at 0x2F (Port F default value) to 10 (0x0a). Address 0x2F contains the default value for PIC Port F at power up. In this example we set it to 0x0a = binary 00001010. This will set the default value of port F pins 1 and 3 to 1 (5V), and pins 0, 2, 4, 5, 6, 7 to 0 (0V) at power up.
kxx=Hyy	Set Config Byte to given hex value - set the given config byte to the given hex value. The 'xx' part gives the offset (in uppercase hex) of the config byte in the "Application Configuration bytes" structure. The value ('yy' part of command) has to be an uppercase hex value from 00 - FF. Examples to use this command: "k00=H1F" Will set the "Application Configuration byte" at 0x00 (MSB of IP address) to 0x1F

Command Syntax	Description
kxxmzz=Hyy	<p>Set Config Byte to given hex value, using mask - set the given config byte to the given hex value, using the given mask. The 'xx' part gives the offset (in uppercase hex) of the config byte in the "Application Configuration bytes" structure. The 'zz' part gives the mask (in uppercase hex) of the bits in the config byte to change. If the mask is 00, no bits are changed. If the mask is FF, all bits are set to the given value. The value ('yy' part of command) has to be an uppercase hex value from 00 - FF.</p> <p>Examples to use this command:</p> <p>"k05m04=HFF" Will set the third bit (bit position 2) of the "Application Configuration byte" at 0x05.</p> <p>"k1Am80=H00" Will clear the eighth bit (bit position 7) of the "Application Configuration byte" at 0x1A.</p>
kxxby=[0,1]	<p>Set or Clear bit in Config Byte - set or clear a bit in the given config byte to the given value. The 'xx' part gives the offset (in uppercase hex) of the config byte in the "Application Configuration bytes" structure. The 'y' part gives the position (value from '0' to '7') of the bit in the config byte to change. The value has to be a '0' or '1' character, indicating if the given bit is to be cleared or set.</p> <p>Examples to use this command:</p> <p>"k05b6=1" Will set the seventh bit (bit position 6) of the "Application Configuration byte" at 0x05.</p> <p>"kA5b0=0" Will clear the first bit (bit position 0) of the "Application Configuration byte" at 0xA5.</p>

General Commands

The following General commands are implemented:

Command Syntax	Description
la=str	<p>New Username - assigns a new Username. The given "str" is the new username, and must be an alpha numeric string of 1 to 8 characters. For example:</p> <p>"la=wombat" will set the new username to "wombat".</p>
lb=str	<p>New Password - assigns a new Password. The given "str" is the new password, and must be an alpha numeric string of 1 to 8 characters. For example:</p> <p>"lb=gumtree" will set the new username to "gumtree".</p>
ln=str	<p>New NetBIOS name - assigns a new NetBIOS name. The given "str" is the new name, and must be an alpha numeric string of 1 to 15 characters. For example:</p> <p>"lb=CROC1" will set the new username to "CROC1".</p>

Short Commands

The following Short commands are implemented:

Command Syntax	Description
m=r	<p>Reset Board - reset's the board. This command can be used to remotely reset the board. This can be used to update the firmware via the network bootloader for example.</p>
m=o	<p>Log Out - log out the current user.</p>

PWM Commands

The following PWM commands are implemented:

Command Syntax	Description
w[1-4]=[0-1023]	<p>Set PWM Value. This command will set the value (Duty Cycle) of the given PWM channel. The possible values must be in the range 0-255 for 8-bit mode, and 0-1023 for 10-bit mode. A value of 0 will give a duty cycle of 0% (output off), where a maximum value will give a duty cycle of 100% (permanently on).</p> <p>Examples to use this command:</p> <p>"w1=128" will set the duty cycle of PWM channel 1 to 50% when in 8-bit mode</p> <p>"w3=256" will set the duty cycle of PWM channel 3 to 25% when in 10-bit mode</p>
w[1-4]r=[0-1023]	<p>Set PWM Value, Remember. Same as the <i>Set PWM Value</i> command, except that the settings are</p>

Command Syntax	Description
	saved to the application configuration, and are remembered after power up.
wf=[0-3]	Set PWM Frequency. This command will set the PWM frequency. The available frequencies depends on the PWM mode (8-bit or 10-bit) used. The frequency is given by a constant value between 0 to 3. For 8-bit mode , the following values are defined: 0 and 1=9.766kHz, 2=39.062kHz, 3=156.25kHz For 10-bit mode , the following values are defined: 0=2.441kHz, 1=9.766kHz, 2 and 3=39.062kHz
wfr=[0-3]	Set PWM Frequency, Remember. Same as the <i>Set PWM Frequency</i> command, except that the settings are saved to the application configuration, and are remembered after power up.
we=XX	Enable PWM Channels. This command will enable/disable the given PWM channels. The command value (XX part) is a two character, uppercase hex value indicating which channels are to be enabled. Each bit in the byte represents a PWM channel. Bit 0 represents PWM channel 1, bit 1 PWM channel 2, ..., and bit 7 PWM channel 8. The SBC65EC has 4 PWM channels (C1, C2, D0 and D3), so only the first 4 bits are used. Examples to use this command: "we=00" will disable all channels "we=1F" will enable PWM channels 1 to 5, and disable all the rest (if any) "we=14" will enable PWM channel 3 and 5, and disable all the rest (if any)
wer=XX	Enable PWM Channels,Remember. Same as the <i>Enable PWM Channels</i> command, except that the settings are saved to the application configuration, and are remembered after power up.
wm=[8,a]	Set PWM Mode. This command will set the PWM mode. When the value is '8', the mode is set to 8-bit mode. When the value is 'a', the mode is set to 10-bit mode.
wmr=[8,a]	Set PWM Mode, Remember. Same as the <i>Set PWM Mode</i> command, except that the settings are saved to the application configuration, and are remembered after power up.

9 Tags

9.1 Introduction

For an up to date list of all current tags, see the online documentation at:

www.modtronix.com/products/sbc65ec/doc/tags

Data can be requested from the target board via Tags. Tags can be:

- Placed in Dynamic Web Pages
- Requested via a UDP messages
- Requested via the HTTP GET method

All Tags must have the format:

%nxx

Where the '%' character serves as a control code, 'n' represents the **variable group**, and "xx" represents a two-digit **variable value** (in upper case hex format).

The **variable value** has a range of 00-FF (Which translates to 0-255 decimal), and must use upper case characters!

The **variable group** ('n' character) can be any alpha numeric character ('0-9', 'a-z', 'A-Z'), giving a total of $10+26+26 = 62$ groups. Each group can have 256 variable values. This gives a total of 15872 possible variables.

An example tag is "%a02". This tag will be replaced by a '1' or '0' character representing the current value of Port A2. If this tag is placed on a Dynamic Web page, it will be shown on a Web Browser requesting that page as a '1' or '0'.

9.2 Dynamic Web Pages

Any of the Tags listed below can be placed on a CGI web page, and will be replaced by the requested data when sent to the HTTP Client (Web Browser for example).

The HTTP Server can dynamically alter pages and substitute real-time information, such as input/output status. The web server will parse CGI files for special tags, and replace them with user defined strings. By default, only "*.cgi" files are parsed for these

special tags. It is however possible configuring other files to be parsed for tags too.

To incorporate this real-time information, the corresponding Dynamic file (*.cgi by default) must contain a text string with the format: "%nxx". Where the '%' character serves as a control code, 'n' represents the variable group and "xx" represents a two-digit variable value (in upper case hex format).

The variable value has a range of 00-FF (Which translates to 0-255 decimal), and must use upper case characters!

The variable group ('n' character) can be any alpha numeric character ('0-9', 'a-z', 'A-Z'), giving a total of $10+26+26 = 62$ groups. Each group can have 256 variable values. This gives a total of 15872 possible variables.

When the HTTP Server encounters this text string, it removes the '%' character and calls the HTTPGetVar() function. This function has been implemented by the Modtronix SBC65EC Web Server in the "httpexec.c" file! All tags implemented by the Modtronix SBC65EC Web Server are listed below.

The HTTPGetVar() function will define strings that will replace tags found in the Dynamic file. If the page requires '%' as a display character, it should be preceded by another '%' character. For example, to display "23%" in a page, put "23%%".

9.3 Tags via UDP command

Any of the Tags listed below can be executed on the target by sending them to UDP port 54123. This port is configurable, and can be changed. The reply (requested data) will be returned to the UDP Socket that sent this Request Command. For example, to request the value of port B2, returned as a 0 or 1 character (0=0V at input, 1=5V at input), send the following command string via UDP to port 54123:

```
%b02
```

The value of port B2 will be returned via UDP.

Another example could be to request the analog input value of Analog port 2, returned as a decimal value between 0 - 1023. To do this, send the following command string via UDP to port 54123:

```
%n12
```

9.4 Tags via HTTP GET method

This feature has not been implemented yet! Will be implemented in a future version! Any of the Tags listed below can be executed on the target by sending them via the HTTP GET method. The reply (requested data) will be returned to the UDP Request Command Port (54124 by default).

9.5 Defined Tags

Port Tags

The variable groups **a** to **j** can be used to display the value of any input pin or port on the PIC. The lower 3 bits of the 8 bit variable value is used to specify the desired bit of the given port (port is given by variable group). The upper 5 bits are used to specify how the ports value should be displayed. The following variable groups are defined:

<i>Variable Group</i>	<i>Description</i>
a	Port A
b	Port B
c	Port C
d	Port D
e	Port E
f	Port F
g	Port G
h	Reserved for future use
j	Reserved for future use

The following variable values(in hex) are defined:

Variable Value	String returned by HTTP Server - displayed on web page
00 - 07	'1' or '0' returned depending on PORT state
10 - 17	"on" or "off" returned depending on PORT state
20 - 27	"<!--" returned if port is configured as input
28 - 2f	"-->" returned if port is configured as input
30 - 37	"<!--" returned if port is configured as output
38 - 3f	"-->" returned if port is configured as output
40 - 47	"checked" returned if port is configured as output
48 - 4f	"checked" returned if port is configured as input
50 - 57	"0" returned if port is configured as output. "1" returned if port is configured as input.

For variable values where bit 3 is set (values from x8 to xf):

x8=port bit 0, x9=port bit 1, xa=port bit 2, xb=port bit 3

xc=port bit 4, xd=port bit 5, xe=port bit 6, xf=port bit 8

For example, variable value 48 will return "checked" if port **bit 0** is configured as an input

Examples:

Example	Description
%a02	Variable group = a, Variable value = 0x02. This example will display the value of port A2 as '1' or '0'. So, if port A2 is set, a '1' will be displayed on the web page in stead of '%a02'
%c17	Variable group = c, Variable value = 0x17. This example will display the value of port C7 as 'on' or 'off'. So, if port C7 is clear for example, 'off' will be displayed on the web page in stead of '%c17'
%f22	Variable group = f, Variable value = 0x22. Variable values 20 to 2F can be used to place HTML comments around HTML code if a port is configured as an input. For example, to only display an image if a Port pin RF1 is configured as an output, you could write: <code>%f21%f29</code>
%g52	Variable group = g, Variable value = 0x52. This example will display '1' if port G2 is configured as an input, and '0' if port G2 is configured as an output.
%b45	Variable group = b, Variable value = 0x45. This example will display 'checked' if port B5 is configured as an output.
%c4c	Variable group = c, Variable value = 0x4c. This example will display 'checked' if port C5 is configured as an input.

Analog Inputs Tags

The variable group **n** can be used to display the value of any Analog Input on the PIC.

The following variable values(in hex) are defined:

Variable Value	String returned by HTTP Server - displayed on web page
00 - 0A	3 digit uppercase hex value
10 - 1A	Decimal value 0 - 255 or 0 - 1023
20 - 2A	Decimale value, 2 decimal places for 5V reference
30 - 3A	"<!--" returned if the channels is configured for ADC
40 - 4A	"-->" returned if the channels is configured for ADC
50 - 5A	"<!--" returned if the channels is NOT configured for ADC
60 - 6A	"-->" returned if the channels is NOT configured for ADC

Examples:

Example	Description
%n02	Variable group = n, Variable value = 0x02. This example will display the value of Analog Input 2 in uppercase hex. For example, "0A8" will be displayed on the web page in stead of '%n02' if Analog Input 2 has the value 0x0A8.
%n1A	Variable group = n, Variable value = 0xaA. This example will display the value of Analog input 10 in decimal. For example, "210" will be displayed on the web page in stead of '%n1A' if Analog Input 10 has the value 210.

Example	Description
%n32	Variable group = n, Variable value = 0x32. Variable values 30-3A and 40-4A can be used to place HTML comments around HTML code if a Port pin is configured as an Analog Input. For example, to only display an image if a Analog Input 2 is configured as an Analog Input, you could write: %n32%n42

General Tags

The variable group **I** is used for displaying general information.

The following variable values(in hex) are defined.

Variable Value	String returned by HTTP Server - displayed on web page
00	Deprecated - use the Username command in the Secure Tags group! Displays the Username of the current user. For example "Guest" or "Admin". The current user will be "Guest" if the user has not logged in. This is a Secure Tag, and requires Authentication to be displayed!
01	Displays the TCP/IP stack version . For example "V2.04".
02	Displays the Application version . For example "V3.00".
03	Returns '1' if Authentication has been provided, else '0'.
04	Displays 'Yes' if the board has a bootloader , else 'No'
05	Displays our NetBIOS name
10	Displays first part of current IP address . For example, will be '5' if our IP is '10.1.0.5'.
11	Displays second part of current IP address .
12	Displays third part of current IP address .
13	Displays fourth part of current IP address . For example, will be '10' if our IP is '10.1.0.5'.
14	Displays first part of MAC address . For example, will be '5' if our MAC is '0.1.2.3.4.5'.
15	Displays second part of MAC address .
16	Displays third part of MAC address .
17	Displays fourth part of MAC address .
18	Displays fifth part of MAC address .
19	Displays sixth part of MAC address . For example, will be '200' if our MAC is '200.1.2.3.4.5'.
1A	Displays first part of current Network MASK . For example, will be '255' if our mask is '0.0.0.255'.
1B	Displays second part of current Network MASK .
1C	Displays third part of current Network MASK .
1D	Displays fourth part of current Network MASK . For example, will be '15' if our mask is '15.0.0.255'.
1E	Displays first part of current Gateway address . For example, will be '5' if our Gateway Address is '10.1.0.5'.
1F	Displays second part of current Gateway address .
20	Displays third part of current Gateway address .
21	Displays fourth part of current Gateway address . For example, will be '10' for '10.1.0.5'.

Examples:

Example	Description
%100	Variable group = 1, Variable value = 00. This example will display the name of the user currently logged in. If the username of the user currently logged in is "Admin", then "Admin" will be displayed on the web page in stead of "%100"
%110.%111.%112.%113	Variable group = 1, Variable value = 10 to 13. This example will display our IP address.

Configuration Tags

The variable group **k** is used for displaying the contents of the Application Configuration bytes. The values are displayed in decimal! The variable value (in uppercase hex) is used to specify the offset of the configuration byte to display. All of these Tags are **Secure Tags**, and will only be parsed if Authentication has been provided! If no Authentication has been provided, they will return 0. The "Authentication" tag in the "General Tags" group can be used to determine if Authentication has been provided.

Examples:

Example	Description
%k00.%k01.%k02.%k03	This example will display the currently configured IP address. If the IP address is "10.1.0.1", then "10.1.0.1" will be displayed on the web page in stead of "%k00.%k01.%k02.%k03".
%k0A.%k0B.%k0C.%k0D	This example will display the currently configured network MASK. If the MASK is "255.0.0.0", then "255.0.0.0" will be displayed on the web page in stead of "%k0A.%k0B.%k0C.%k0D".

PWM Tags

The variable group **w** can be used to display the current settings of the PWM outputs.

The following variable values(in hex) will display the current value of a PWM channel.

The SBC65EC has 4 PWM channels, so the second character can have a value from 1-4.

Variable Value	String returned by HTTP Server - displayed on web page
01 - 05	3 digit uppercase hex value
11 - 15	Decimal value 0 - 255 (for 8-bit mode) or 0 - 1023 (for 10-bit mode)
31 - 35	"<!--" returned if the PWM output is enabled
41 - 45	"-->" returned if the PWM output is enabled
51 - 55	"<!--" returned if the PWM output is disabled
61 - 65	"-->" returned if the PWM output is disabled

The following additional variable values are defined for the PWM channels.

Variable Value	String returned by HTTP Server - displayed on web page
F0	Display the set frequency value. The frequency is given by a constant value between 0 to 3. For 8-bit mode , the following values are defined: 0 and 1=9.766kHz, 2=39.062kHz, 3=156.25kHz For 10-bit mode , the following values are defined: 0=2.441kHz, 1=9.766kHz, 2 and 3=39.062kHz
F4	Displays the set mode. A '8' indicates we are currently configured for 8-bit mode. A 'a' indicates we are currently configured for 10-bit mode.
F8	Displays the enabled PWM channels. An uppercase hex value is returned that indicates what channels are enabled. Each bit in the hex value represents a PWM channel.

Secure Tags

The variable group **s** is used for displaying secure information. All of these Tags are **Secure Tags**, and will only be parsed if Authentication has been provided! If no Authentication has been provided, they will return 0. The "Authentication" tag in the "General Tags" group can be used to determine if Authentication has been provided.

The following variable values(in hex) are defined.

Variable Value	String returned by HTTP Server - displayed on web page
00	Displays the Username of the current user. For example "Guest" or "Admin". The current user will be "Guest" if the user has not provided Authentication.
01	Displays the Password of the current user.

10 Specifications

10.1 Absolute Maximum Ratings

<i>Item</i>	<i>Symbol</i>	<i>Min</i>	<i>Typ</i>	<i>Max</i>	<i>Unit</i>
Operating Temperature:	Top	0		70	°C
Storage Temperature:	Tst	-65		140	°C

10.2 Electrical Characteristics

<i>Item</i>	<i>Symbol</i>	<i>Condition</i>	<i>Min</i>	<i>Typ</i>	<i>Max</i>	<i>Unit</i>
DC Supply Voltage:	Vdd	-	7		35	V
Typical Operating Current at 40MHz	Idd	Vdd = 12V	55	58	65	mA
Typical Operating Current at 10MHz	Idd	Vdd = 12V	32	35	39	mA
RJ45 Ethernet connector DCR RX/TX		T=25°C		0.35		Ω
RJ45 Ethernet connector inductance		T=25°C		0.3		uH
RJ45 Ethernet connector capacitance		T=25°C		12		pF
RJ45 Ethernet connector Hi-Pot test		T=25°C		1500		Vrms

The RJ45 connector (Ethernet connector) meets IEEE 802.3 standards and FCC mechanical requirements.

10.3 D.C. Characteristics of user I/O pins on Daughter Board connector.

<i>Item</i>	<i>Symbol</i>	<i>Condition</i>	<i>Min</i>	<i>Typ</i>	<i>Max</i>	<i>Unit</i>
Input Low Voltage - configured as TTL input:	V _{IL}		0		0.75	V
Input Low Voltage - configured as Schmitt Trigger input:	V _{IL}		0		1	V
Input High Voltage - configured as TTL input:	V _{IH}		2.05		5	V
Input High Voltage - configured as Schmitt Trigger input:	V _{IH}		4		5	V
Output High Voltage:	V _{OL}	I _{OL} = 8.5mA			0.6	V
Output Low Voltage:	V _{OH}	I _{OH} = 3mA	4.3			V
Capacitive loading:	C _{IO}			50		pF

Many inputs on the PIC18F6627 are Schmitt Trigger inputs, consult the data sheet for details.

11 Dimensions

The SBC65EC conforms to the MicroX Compact Main Board Dimensions, as shown in Figure 4.

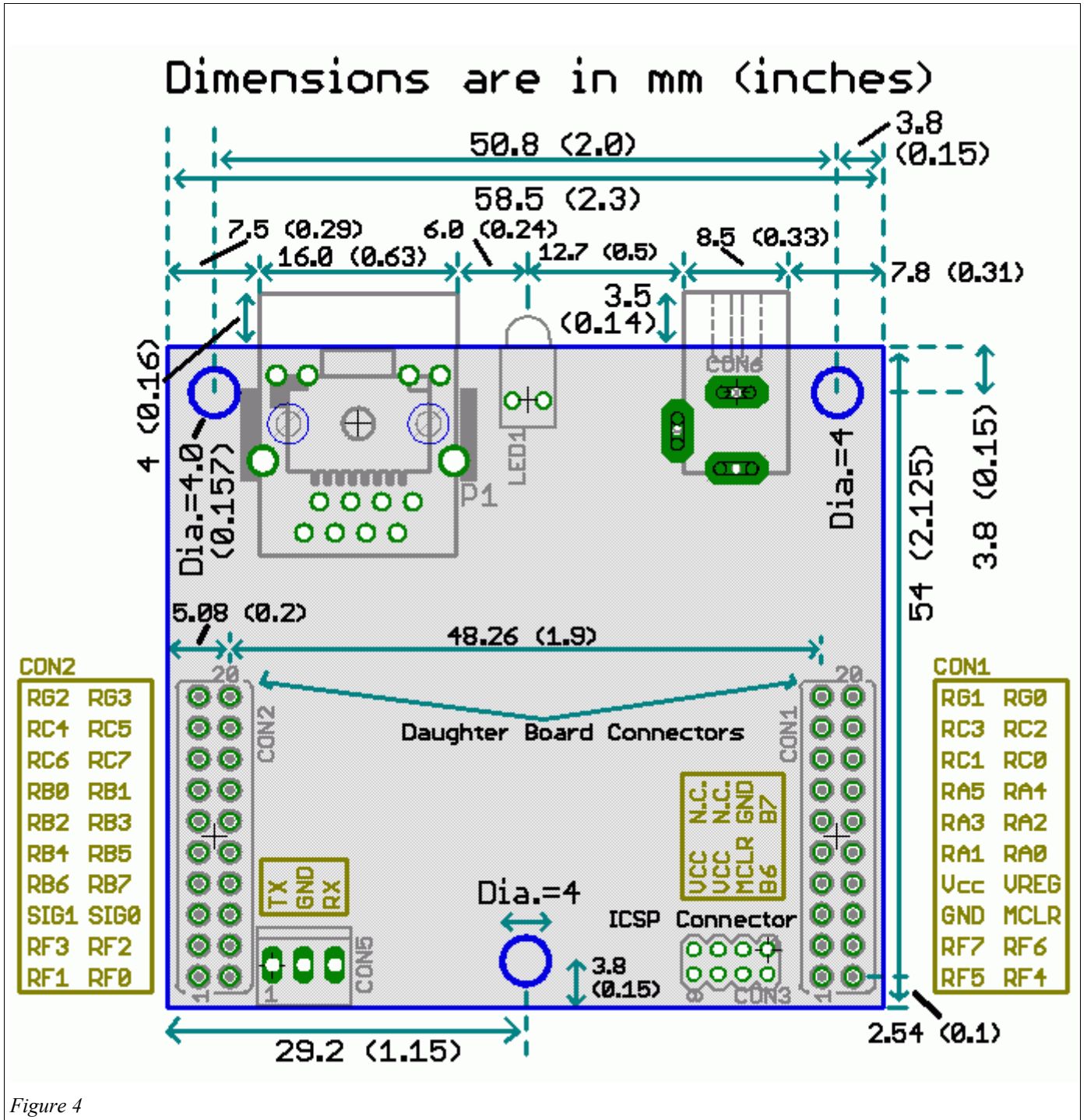


Figure 4

The SBC65EC's PCB layout is shown in Figure 5.

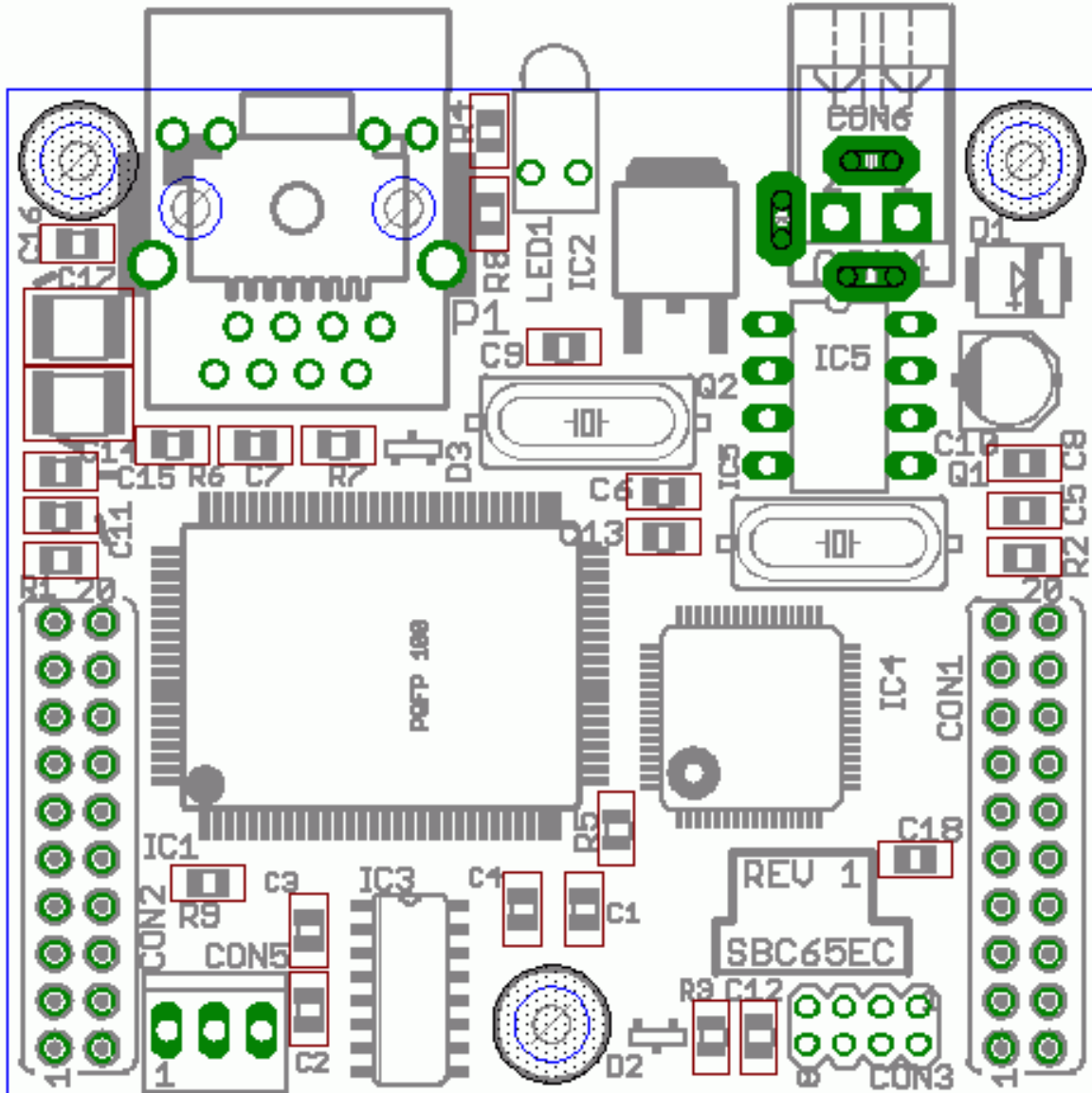


Figure 5