Eta32mini

ATMEL AVR Development Kit
General Description

**Eta32mini** kit is an AVR development board designed particularly for students, beginners and recent graduated engineers to provide easy developing of AVR microcontroller projects.

**Eta32mini** supports ATMEGA32 microcontroller chip. **Eta32mini** kit provides the most common primary devices and circuits, such as LCD, KEYPAD, serial interface, and more. Thus, it saves most of the wasted time and efforts in hardware design and test, hence, a developer can focus his efforts on firmware development. All I/O pins are brought out via pin header for direct port accessing.

**Eta32mini** kit comes preprogrammed with UART bootloader, which eliminates the need for external programmer (programming using Eta32mini IDE from FARES PCB Co). However, kit can be programmed using Arduino IDE (another bootloader). Also standard 6 pin header socket is included (ICSP) for external programmers.
**Eta32mini key features**

- Power supply unit.
- ATMEGA32 Microcontroller with reset switch and crystal oscillator.
- 4X4 matrix keypad.
- Output unit involves three output LEDs, two output relays and one output buzzer.
- Multiplexed two digits 7 segments display.
- 2X16 LCD display.
- Variable analog input using high precise multi-turn resistor.
- Virtual COM port via USB.
- Standard 6 pin header socket ICSP for external programmers.
- All microcontroller I/O pins are brought out via pin header.
Power Supply Unit

Power could be supplied from DC power supply adaptor via DC power socket (12V - 20V).

Power supply unit features:
- On/Off Power switch.
- Green LED power indicator.
- 7805 (5V power regulator 5% tolerance).
- 1A Fuse for over current protection.

**Note:** - *Eta32mini* kit is protected against reversed polarity of power.

Power supply unit (schematic)
Microcontroller Unit

44 pin TQFP ATMEGA32A chip with 16 MHz crystal oscillator and Push button reset switch.
4X4 Matrix Keypad Unit

**Eta32mini** Kit includes 16 push button switches. Switches are configured as 4 rows intersected by 4 columns. Each intersection creates a switch position.

![4X4 Matrix Keypad Unit](image)

**4X4 Matrix Keypad Unit**

**4X4 Matrix Keypad Unit (schematic)**
Rows and Columns are connected to microcontroller as following.

<table>
<thead>
<tr>
<th>Row number</th>
<th>Microcontroller pin</th>
<th>Arduino pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW 1</td>
<td>PB4</td>
<td>4</td>
</tr>
<tr>
<td>ROW 2</td>
<td>PB5</td>
<td>5</td>
</tr>
<tr>
<td>ROW 3</td>
<td>PB6</td>
<td>6</td>
</tr>
<tr>
<td>ROW 4</td>
<td>PB7</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column number</th>
<th>Microcontroller pin</th>
<th>Arduino pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL 1</td>
<td>PD2</td>
<td>10</td>
</tr>
<tr>
<td>COL 2</td>
<td>PD3</td>
<td>11</td>
</tr>
<tr>
<td>COL 3</td>
<td>PD4</td>
<td>12</td>
</tr>
<tr>
<td>COL 4</td>
<td>PD5</td>
<td>13</td>
</tr>
</tbody>
</table>

Keypad can be configured as 4X4 matrix keypad (16 switches used) or as 4 direct switches.
Matrix Mode

In matrix mode, all 16 switches are activated and configured as four row lines and four column lines. Microcontroller scans these lines to detect a button pressed state. Column lines are pulled up by 10KΩ resistors. (i.e. microcontroller port pin reads high if no switch is pressed).

Scan process starts by setting all rows and columns as inputs. To scan switches in a row, microcontroller configures it as output and sets it to low, then checks columns one at a time. If a column line goes low, microcontroller detects a pressed switch, otherwise, no pressed switch and goes to scan next row and so on.

To set keypad in matrix mode, just set MODE jumper to Matrix position as shown below.
Direct Mode

In direct mode, just 4 switches are activated to save microcontroller pins in case of no need for more switches and reducing firmware complexity in scanning operation. Switches in first row only is enabled (i.e. SW1, SW2, SW3 and SW4) and can be read directly as inputs through column lines. Microcontroller checks the column lines. If a column line goes low, it means a switch is pressed.

To set keypad in direct mode, just set MODE jumper to direct position, as shown below:
Output Unit
This unit contains six outputs distributed as three LEDs, two relays and one Buzzer.

Outputs are connected to microcontroller as following:

<table>
<thead>
<tr>
<th>Output</th>
<th>Microcontroller pin</th>
<th>Arduino pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buzzer</td>
<td>PC5</td>
<td>21</td>
</tr>
<tr>
<td>Relay1</td>
<td>PC4</td>
<td>20</td>
</tr>
<tr>
<td>Relay2</td>
<td>PC3</td>
<td>19</td>
</tr>
<tr>
<td>Blue LED</td>
<td>PC2</td>
<td>18</td>
</tr>
<tr>
<td>Green LED</td>
<td>PC1</td>
<td>17</td>
</tr>
<tr>
<td>Red LED</td>
<td>PC0</td>
<td>16</td>
</tr>
</tbody>
</table>
Output unit (Schematic)

Three output LEDs

Three LEDs with current limiting resistors are connected to port pins (PC0, PC1 and PC2). Red, Green and Blue LED.

Blue LED is connected to PC.2 in microcontroller.
Green LED is connected to PC.1 in microcontroller.
Red LED is connected to PC.0 in microcontroller.

Each LED can be enabled or disabled via DIP switch. LEDs are active high. i.e. output high turns on LED and output low turns off it.
Two output relays

Two output relays are added to *Eta32mini* kit to allow dry contact switches, which is suitable for AC or DC switching applications. Each relay has its own related LED for status indication and can be individually enabled via DIP switch. Relays are driven from NPN transistors. Freewheeling diodes are included to protect transistors from back EMF voltage that arises on relay coil during switching off.

Relay1 is attached to port pin **PC4** and relay2 is attached to port pin **PC3**. Relays are 5V coil and rated to 5A contacts (resistive load). Both of normally open and normally closed contacts are brought out via screw clamp terminals.

One output buzzer

One output buzzer (6VDC) is included in output unit to port pin **PC5**. Also, it may be enabled using DIP switch.
7Segment Display
7segment display is used to indicate numerical data. It can display digits from 0 to 9 and some characters such as A,B,C,H,E,F, etc. 7segment display is very popular and has many applications. Eta32mini includes multiplexed two digits 7segment display.

Multiplexed Two Digits 7seg Display Unit
Two multiplexed 7seg digits are added to Eta32mini kit. 7segments are referred to by letters “a”, “b”, “c”, “d”, “e”, “f”, “g”, “dot”. Both digits share the same segments. i.e. segment “a” is the same for DIG1 and DIG2. Each digit has its own common, so there are two commons for the two digits. 7segment display is common cathode, so a high on a segment turns it ON while low turns it OFF. Commons are driven from NPN transistors. So, a high from microcontroller turns transistor on, which enables the digit. Segments commons are connected to microcontroller as shown in table:

<table>
<thead>
<tr>
<th>Segment</th>
<th>microcontroller Pin</th>
<th>Arduino Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>PA1</td>
<td>25</td>
</tr>
<tr>
<td>B</td>
<td>PA2</td>
<td>26</td>
</tr>
<tr>
<td>C</td>
<td>PA3</td>
<td>27</td>
</tr>
<tr>
<td>D</td>
<td>PA4</td>
<td>28</td>
</tr>
<tr>
<td>E</td>
<td>PA5</td>
<td>29</td>
</tr>
<tr>
<td>F</td>
<td>PA6</td>
<td>30</td>
</tr>
<tr>
<td>G</td>
<td>PA7</td>
<td>31</td>
</tr>
<tr>
<td>Dot</td>
<td>PB0</td>
<td>0</td>
</tr>
</tbody>
</table>

7Seg Commons are connected to microcontroller as shown in table:

<table>
<thead>
<tr>
<th>Common</th>
<th>microcontroller Pin</th>
<th>Arduino Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digit 1</td>
<td>PC6</td>
<td>22</td>
</tr>
<tr>
<td>Digit 2</td>
<td>PC7</td>
<td>23</td>
</tr>
</tbody>
</table>

Note:-
- Each 7seg digit can be enabled or disabled individually using DIP switch.
- 7seg module and 2X16LCD share the same port. So, disable LCD before using 7Segments display.
7Segment display

7Segment display (schematic)
2X16 LCD Display Unit

2X16 alphabetic LCD with contrast control configured in 4 bit mode is connected to PORTA via DIP switch as following:

LCD module pinout number, symbol and function:

<table>
<thead>
<tr>
<th>LCD pin number</th>
<th>LCD pin symbol</th>
<th>LCD pin function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VSS</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>VCC</td>
<td>+5V</td>
</tr>
<tr>
<td>3</td>
<td>VO</td>
<td>Contrast adjustment</td>
</tr>
<tr>
<td>4</td>
<td>RS</td>
<td>Register Select(0:Command, 1:Data)</td>
</tr>
<tr>
<td>5</td>
<td>R/W</td>
<td>R/W(0:Write, 1:Read)</td>
</tr>
<tr>
<td>6</td>
<td>EN</td>
<td>Enable</td>
</tr>
<tr>
<td>7</td>
<td>D0</td>
<td>Data bit 0</td>
</tr>
<tr>
<td>8</td>
<td>D1</td>
<td>Data bit 1</td>
</tr>
<tr>
<td>9</td>
<td>D2</td>
<td>Data bit 2</td>
</tr>
<tr>
<td>10</td>
<td>D3</td>
<td>Data bit 3</td>
</tr>
<tr>
<td>11</td>
<td>D4</td>
<td>Data bit 4</td>
</tr>
<tr>
<td>12</td>
<td>D5</td>
<td>Data bit 5</td>
</tr>
<tr>
<td>13</td>
<td>D6</td>
<td>Data bit 6</td>
</tr>
<tr>
<td>14</td>
<td>D7</td>
<td>Data bit 7</td>
</tr>
<tr>
<td>15</td>
<td>A</td>
<td>Back light anode(+)</td>
</tr>
<tr>
<td>16</td>
<td>K</td>
<td>Back light cathode(-)</td>
</tr>
</tbody>
</table>

LCD connection to microcontroller port pins:

<table>
<thead>
<tr>
<th>LCD pin number</th>
<th>LCD pin symbol</th>
<th>Microcontroller pin</th>
<th>Arduino Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>RS</td>
<td>PA1</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>EN</td>
<td>PA2</td>
<td>26</td>
</tr>
<tr>
<td>11</td>
<td>D4</td>
<td>PA3</td>
<td>27</td>
</tr>
<tr>
<td>12</td>
<td>D5</td>
<td>PA4</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>D6</td>
<td>PA5</td>
<td>29</td>
</tr>
<tr>
<td>14</td>
<td>D7</td>
<td>PA6</td>
<td>30</td>
</tr>
</tbody>
</table>

Note:
- LCD R/W signal is tied to ground.
- 10KΩ variable resistor labeled "Contrast" is adjusted to control the LCD contrast.
- Disable 7Segment module before using LCD module.
2X16 LCD display unit

2X16 LCD display unit (Schematic)
Analog Input Unit

_Eta32mini_ includes 10KΩ multi-turn variable resistor. The fixed terminals of variable resistor are connected to GND and +5VDC whereas the variable terminal is connected to Port pin **PA0** of the microcontroller. The resistor may be adjusted precisely to the required voltage (0.00V to 5.00V).

Analog input connected to microcontroller as shown in table:

<table>
<thead>
<tr>
<th>Analog Input</th>
<th>microcontroller Pin</th>
<th>Arduino Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable resistor</td>
<td>PA0</td>
<td>24/A0</td>
</tr>
</tbody>
</table>

Analog input unit

![Analog input unit](image)

Analog input unit (schematic)
USB Virtual COM Unit

This unit includes USB/TTL logic converter using CH340 IC and provides USB type B socket for PC serial communication. Serial interface circuit can be enabled or disabled using DIP switch (see figure 25). USB/TTL Converter IC is connected to ATMEGA32 microcontroller port pins **PD0 (RX)** and **PD1 (TX)** via DIP switch.
Uploading Hex Code
There are three ways to burn hex code on *Eta32mini* kit.

1) *Eta32mini* software tool (Eta32mini bootloader from FARES PCB)
2) Arduino IDE.
3) External programmer.

Programming using *Eta32mini* tool
*Eta32mini* does not require any sort of external programmers to be programmed. *Eta32mini* kit comes with Eta32mini bootloader, which interfaces an *Eta32mini* software tool (from FARES PCB). *Eta32mini* software tool is used to download/upload hex code from/to ROM and EEPROM contents of ATMEGA32 microcontroller.
**Eta32mini** Software tool is developed specifically for **Eta32mini** kit. It detects the kit automatically just after opening it or by clicking re-connect button. Select memory type (Flash, EEPROM or both) before any programming operation. Use “Load” button to load hex file to buffer related to selected memory. Use “Save” button to store data in the buffer related to selected memory into a hex file (Intel format). Use “Write” button to burn loaded hex file to selected memory on microcontroller. Use “Read” button to read hex code from selected memory of microcontroller. Use “Erase” button to erase selected memory of microcontroller. Use “Verify” button to compare hex code from selected memory of microcontroller to the related memory buffer.

**Fuse settings**
By default, fuse bytes are set to the proper values as shown in table

<table>
<thead>
<tr>
<th>Fuse byte</th>
<th>Hex value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>9F</td>
</tr>
<tr>
<td>High</td>
<td>CA</td>
</tr>
</tbody>
</table>

Brown-out Detector (BOD): Enabled (2.7V).
Clock Source: External crystal oscillator.
Boot reset vector is selected. Bootloader occupies 2K bytes from flash memory.

**Note:**
If fuse bytes are changed to any values other than those shown in the previous table, they must be reprogrammed to the proper values before using **Eta32mini** programming tool.

After programming is completed, a reset operation is performed to ensure microcontroller correct operation.

**Warning:**
Random fuse settings changing is risky. You should take special care while changing these settings. Incorrect fuse settings may cause incorrect microcontroller functioning.
Programming using Arduino IDE

If Arduino software tool is your preferred IDE, follow the next steps to add *Eta32mini* kit to Arduino boards,

<table>
<thead>
<tr>
<th>Fuse byte</th>
<th>Hex value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>9F</td>
</tr>
<tr>
<td>High</td>
<td>CE</td>
</tr>
</tbody>
</table>

1 - Upload Arduino bootloader firmware (*ArduinoBL.hex*) included in CD using any external programmer such as USBasp programmer. (See programming using external programmer). Fuse Bytes must be set as shown in table below

2 - Download Arduino IDE from the following link
https://www.arduino.cc/en/Main/Software

3 - Close Arduino IDE.

4 - Run the self-extracted file (*Eta32miniDuino.exe*) included in CD, and browse to the Arduino IDE installation path (C:\Program Files (x86)\Arduino\hardware), and press Extract button to start extraction

5 – After extraction, *Eta32mini* board is added to Arduino Boards. You can select Eta32mini board as following:
ATMEL AVR Development Kit (Eta32mini)

Tools >> Board: >> Eta32mini.
6 - Also, you will find many Arduino example codes to test all units in kit. You can go to:

File >> Examples >> Examples for Eta32mini.
Programming using external programmer

Although Eta32mini is designed mainly to be programmed using bootloader. However, it offers ICSP socket to enable programming using any external programmer that supports standard 6 pin ICSP socket, such as USBasp programmer. Use external programmer to download your own application code or even a bootloader firmware such as Arduino bootloader.

For more details about programming using USBasp programmer, please refer to USBasp CD from FARES PCB products from this link:
I/O PORT

In addition to the popular built-in circuits and devices included in *Eta32mini* kit, all microcontroller port pins are brought out for external using via header sockets.

External connectors are grouped into 4 units. Each represents one microcontroller port (8 I/O), in addition to GND and 5V.

---

**Microcontroller port pins**

---

**Microcontroller port pins (Schematic)**

**Note:**
Please ensure to disconnect the built-in circuits attached to the port pins intended to use.
**HOW TO START?**

**Step 1**
Install USB driver for ch340.

**Step 2**
Install *Eta32mini* Software tool.

**Step 3**
Connect USB cable to *Eta32mini* kit and open *Eta32mini* software tool. *Eta32mini* tool starts to search COM ports and detect kit automatically. If there’s a problem in connection, please review USB cable connection to kit and click “Re-Connect” button to try again.

**Step 4**
Now you are ready to upload your hex code and enjoy working with *Eta32mini* kit. Refer to Programming Using Eta32mini Tool for more details about using Eta32mini IDE.

If it is the first time to use *Eta32mini* kit, you should perform some test operations on kit before start working on it. The CD included with package contains the firmware code required for testing all modules in kit. So, it's recommended to upload this test code before going to your own application firmware to ensure correct functioning.

Upon burning “*Eta32mini_Test.hex*” code user could test

- Keypad in matrix mode.
- 2X16 LCD.
- Outputs unit (Relays, LEDs and Buzzer).
- 7segments.
- UART Serial operation.
- Analog Input.

- Enable DIP switches of all modules on kit.
- Set keypad to matrix mode (Set KB Mode to Matrix position).
- Plug in USB cable.
- Turn on power switch.
- Open *Eta32mini* programming tool.
- Click “LOAD” button to load the test code included in CD (*Eta32mini_Test.hex*).
- Click Write button.
- After programming is completed the microcontroller is reset automatically.
After power on or reset operation the test sequence is

1 - Serial module transmits this message to serial port
   "FARESPCB Co."
   "ATMEL AVR Development kit."
   "Eta32mini kit."
   "Arduino IDE compatible."

User may receive this statement by HyperTerminal program or any other serial monitor software.

2 - LCD displays the following messages one by one.

![FARESPCB Co.
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=
=...
Some switches perform a test operation for specified modules in kit. These tests are listed in table below.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Serial port / LCD (line2) message</th>
<th>Function</th>
<th>Test operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW1</td>
<td>&quot;SW 1 is pressed.&quot;</td>
<td></td>
<td>Toggle Relay1 status</td>
</tr>
<tr>
<td>SW2</td>
<td>&quot;SW 2 is pressed.&quot;</td>
<td></td>
<td>Toggle Relay2 status</td>
</tr>
<tr>
<td>SW3</td>
<td>&quot;SW 3 is pressed.&quot;</td>
<td></td>
<td>Toggle Blue LED</td>
</tr>
<tr>
<td>SW4</td>
<td>&quot;SW 4 is pressed.&quot;</td>
<td></td>
<td>Toggle Green LED</td>
</tr>
<tr>
<td>SW5</td>
<td>&quot;SW 5 is pressed.&quot;</td>
<td></td>
<td>Toggle Red LED</td>
</tr>
<tr>
<td>SW6</td>
<td>&quot;SW 6 is pressed.&quot;</td>
<td></td>
<td>Digits counts from 00 to 99 and perform auto reset</td>
</tr>
<tr>
<td>SW7</td>
<td>&quot;SW 7 is pressed.&quot;</td>
<td></td>
<td>LCD displays the analog volt measured on port pin (PA0)</td>
</tr>
<tr>
<td>SW8</td>
<td>&quot;SW 8 is pressed.&quot;</td>
<td></td>
<td>LCD displays any received text from the serial port in the second line.</td>
</tr>
<tr>
<td>SW9</td>
<td>&quot;SW 9 is pressed.&quot;</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>SW10</td>
<td>&quot;SW10 is pressed.&quot;</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>SW11</td>
<td>&quot;SW11 is pressed.&quot;</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>SW12</td>
<td>&quot;SW12 is pressed.&quot;</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>SW13</td>
<td>&quot;SW13 is pressed.&quot;</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>SW14</td>
<td>&quot;SW14 is pressed.&quot;</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>SW15</td>
<td>&quot;SW15 is pressed.&quot;</td>
<td></td>
<td>---</td>
</tr>
<tr>
<td>SW16</td>
<td>&quot;SW16 is pressed.&quot;</td>
<td></td>
<td>---</td>
</tr>
</tbody>
</table>
Special Thanks to:
Eng. / Ahmed Ibrahim Ahmed
ah.ahmed@nu.edu.eg

Copyright © 2019 by FARESPCB™

For our full range of products see our website at http://www.farespcb-eg.com
If you have any technical questions about our products, e-mail us at
www.support@farespcb-eg.com

FARESPCB co. (Headquarters)
17 Yossif elgendy st.
Bab ellouq , Tahreer , Cairo
Egypt.
Tel: 02-23904484
Mob: 01000652977

FARESPCB Co reserves the right to make changes in circuit design, software and/or
specifications at any time without prior notification. For the most up-to-date information,
please visit our web site at http://www.farespcb-eg.com

Information furnished by FARESPCB is believed to be accurate and reliable. However,
FARESPCB assumes no responsibility arising from the use of the specifications described.

Warrantee: FARESPCB™ warrants its products against defects in materials and
workmanship for a period of 30 days. If you discover a defect, we will, at our option,
repair or replace your product or refund your purchase price. This warrantee does not
cover products that have been physically abused or misused in any way.

Distributor:
RAM Electronics
32 El Falaky St. Bab El Louk
Tahreer, Cairo
Egypt.
Tel: 02-27960551
www.ram.com.eg
Sales@ram-electronics.com