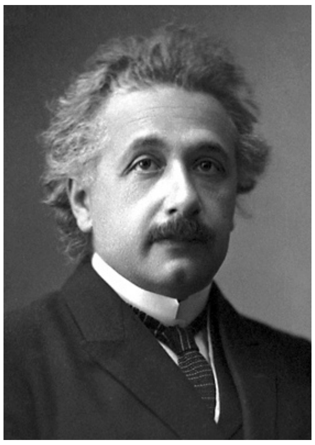
# An 8-bit AVR EMBEDDED SYSTEM

## *Features*

- Six independent pins automation through IR remote control, keyboard or IoT.
- Six independent programmable pins controlled by temperature, battery charger sensors and timers
- Real time clock with alarm
- Calendar
- Default and Scientific Calculator
- 20x4 LCD display
- Two Battery Charger
- Bilingual language support (English/Portuguese)
- 16 MHz MCU clock speed using ATMEGA 328pu microcontroller
- Full IR control available through ProntoHEX code allowing access through IR devices.
- Automotive, Home and Industry versions available in a single hardware
- Low cost components

*Author: Fábio Pereira da Silva*
*03/29/2015*

*Rev.: 2*



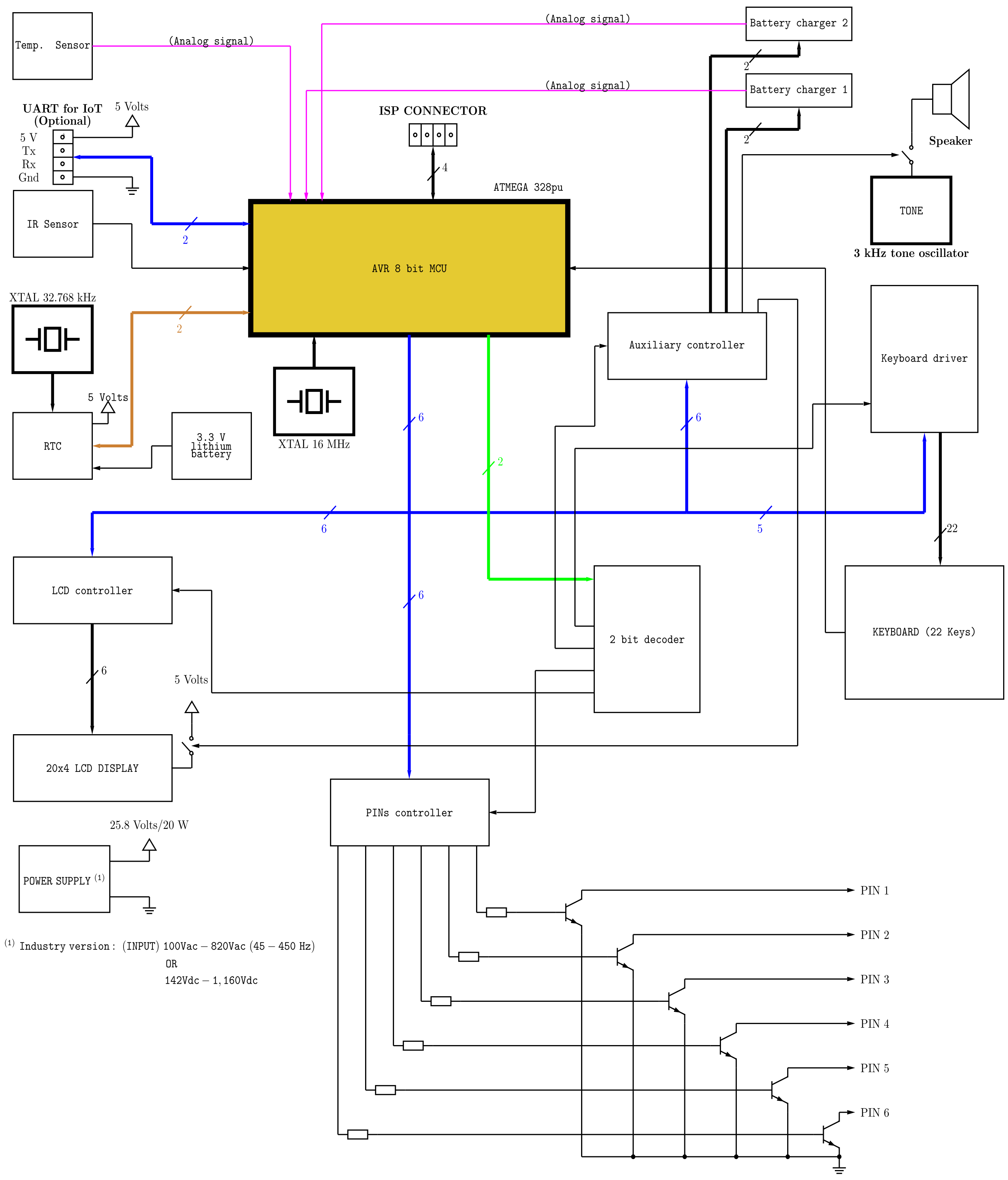
"Try not to become a man of success, but rather try to become a man of value

Albert Einstein

# Overview

This Embedded System was developed with ATMEGA 8-bit family microcontroller. Due to its capacity of processing it's possible to do amazing stuffs. This Embed System is an automated controller that you can control 6 (six) home/industry appliances individually easily by temperature, timer and by user through infrared (remote control or cellphone) or even IoT (optional). In Figure 1 is seen the basic block using ATMEGA 328pu 8-bit microcontroller.

**Figure 1: Basic Block Diagram**



3

# *Hardware interface and connections*

Before turn on this Embedded System you should know this hardware interface and basic connections. Figure 2 shows this 8-bit Embedded System Hardware.

**Figure 2: 8-bit hardware AVR architecture**

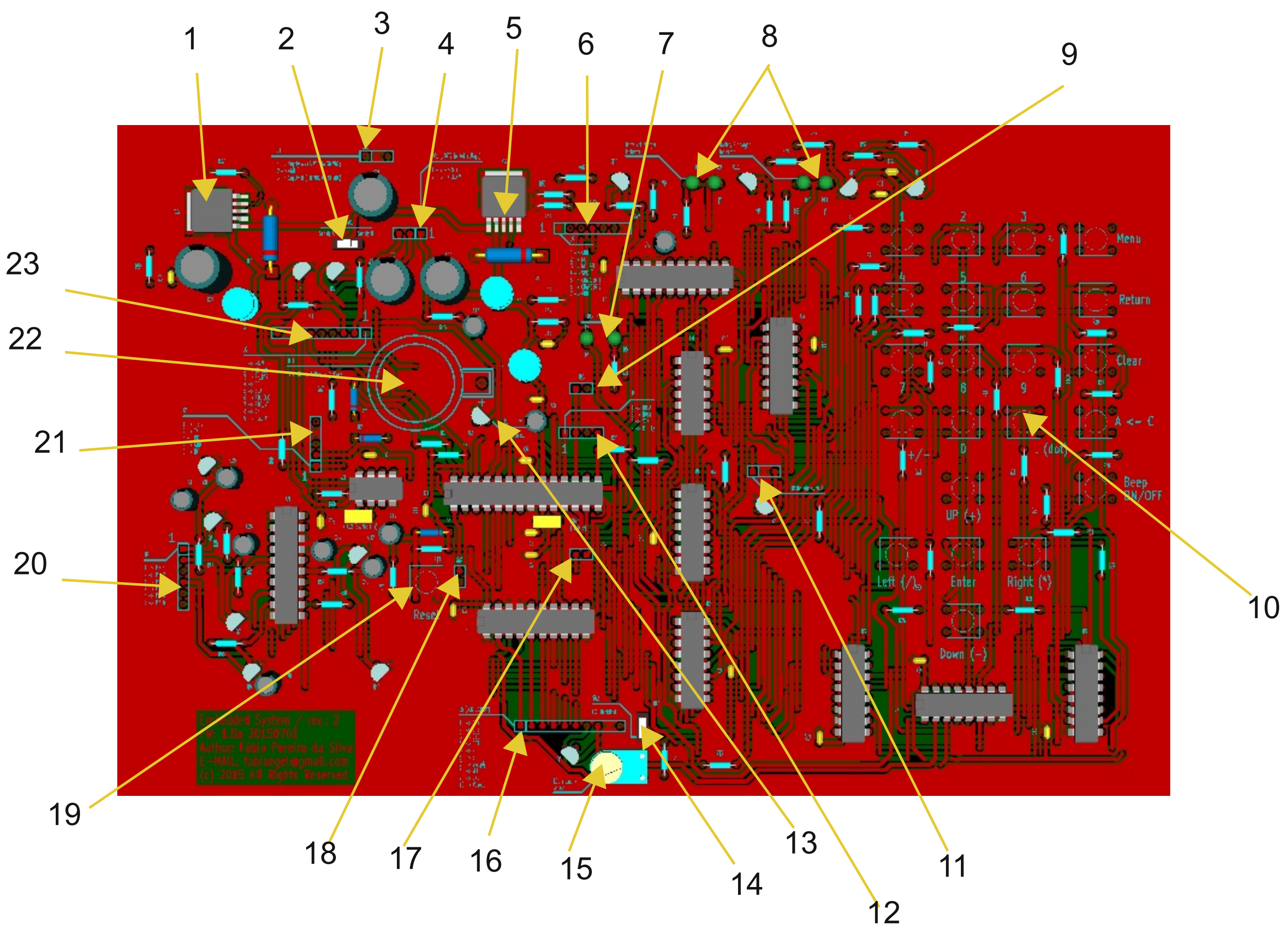


Table 1 shows detailed connections name and descriptions of the Figure 2.

## *Table 1: Connections name and descriptions*

| | ***Name*** | ***Description*** |
|---|---|---|
| **1** | 3.3 V regulator | Provides 3.3 V (2 A) to external devices |
| **2** | SW1 - Input power selector | Selects one of two inputs power supply (A or B) |
| **3** | 2 (two) Input power | Input A and B => Power supply voltage 8 Vdc to 38 Vdc |
| **4** | JP4 (UART2 for IoT Voltage) | Selects operation voltage (3.3 V or 5V) for UART2 for IoT |
| **5** | 5 V regulator | Provides 5 V (1.5 A) for internal and external circuit devices |
| **6** | J8 output interface | Output interface (Battery 1 and 2 chargers and Speaker) |
| **7** | RX and TX indicator leds | Indicates Rx and Tx transmitting data |
| **8** | Battery 1 and 2 indicator leds | Indicates Battery 1 and 2 status |
| **9** | JP3 | Normal operation= (short) / ISP programming = (open) |
| **10** | Keyboard | Numerical and functions keyboard |
| **11** | IR (Infrared Sensor) | For remote control access |
| **12** | J3 (ISP interface) | For ISP interface (firmware update/debug/programming) |
| **13** | T1 | External temperature sensor |
| **14** | SW2 | LCD Backlight Switch |
| **15** | RV1 | LCD intensity adjust |
| **16** | J4 | Output LCD interface |
| **17** | JP1 | Normal operation= (short) / ISP programming = (open) |
| **18** | JP2 | Normal operation= (short) / ISP programming = (open) |
| **19** | Reset button | Resets microcontroller |
| **20** | J6 | PIN's output interface |
| **21** | J2 | UART for IoT interface |
| **22** | Real time clock backup | Lithium Battery (optional) for Real Time Clock (RTC) |
| **23** | J5 | Input/Output interface |

## *Hardware settings*

You need configure hardware to interface other devices. This hardware also provides two post regulated power supply 3.3 V (2 A) and 5 V (1.5 A) to others devices such as Arduino, Raspberry PI, Beagle Bone and many others from 8 Vdc ~ 38 Vdc (automotive application) and 100 Vac ~ 820 Vac (45-450 Hz) or even 142 Vdc to 1160 Vdc (Home and Industrial application) using a special high range SMPS (Switching Mode Power Supply) designed by me.

This devices offers 9 connectors to interface others components or devices

Tables 2, 3, 4, 5, 6, 7 and 8 shows each input/output interface.

## *Table 2: Input power supply connection (J1)*

| ***J1*** | ***Input type*** | ***Description*** |
|---|---|---|
| 1 | Power supply A | Input voltage 8 Vdc to 38 Vdc |
| 2 | GND | Ground |
| 3 | Power Supply B | Input voltage 8 Vdc to 38 Vdc |

## *Table 3: UART1 for IoT and +5 V supply connection (J2)*

| ***J2 (UART1 for IoT)*** | ***Input/output type*** | ***Description*** |
|---|---|---|
| 1 | +5 V | Output power supply |
| 2 | Tx (Output) | UART1 for IoT transmit |
| 3 | Rx (Input) | UART1 for IoT receive |
| 4 | GND | Ground |
| 5 | NC | No connection |

## *Table 4: ISP connection (J3)*

| ***J3 (ISP Programming)*** | ***Input/output type*** | ***Description*** |
|---|---|---|
| 1 | MISO (output) | Data output |
| 2 | MOSI (Input) | Data input |
| 3 | SCK (Input) | Clock from master from programmer |
| 4 | $\overline{\text{RESET}}$ (Input) | Reset inverted logic from programmer |

## Table 5: LCD display interface connection (J4)

| *J4 (LCD display interface)* | *Output type* | *Description* |
|---|---|---|
| 1 ~ 4 | DB4 ~ DB7 | Four high order bits for HD44780 LCD controller |
| 5 | RS | Select register |
| 6 | E | Data write |
| 7 | $V_{supply}$ | +5 V Power supply for LCD controller |
| 8 | VEE | LCD intensity |
| 9 | GND | Ground |
| 10 | $V_{anode}$ | +5 V power supply for LCD backlight |

## Table 6: Input/Output interface connection (J5)

| *J5 (I/O interface)* | *Input/Output type* | *Description* |
|---|---|---|
| 1 | +5 V (output) | Output power supply. Provides 1.5 A for external devices |
| 2 | $\overline{\text{TX}}$ (output) | Inverted output (UART2 for IoT) |
| 3 | $\overline{\text{RX}}$ (input) | Inverted input (UART2 for IoT) |
| 4 | SCL (output) | Output $I^2C$ clock |
| 5 | SDA (input/output) | Input/output $I^2C$ data |
| 6 | RTC_CLK (output) | Real time clock frequency (32.768 Hz, 1024 Hz, 32 Hz or 1Hz) Refer PCF 8563 for details |
| 7 | $\overline{\text{RTC\_INT}}$ (output open drain) | Inverted real time clock interrupt. Refer PCF 8563 for details |
| 8 | GND | Ground |
| 9 | +3.3 V (output) | Output power supply. Provides 2 A for external devices. |

### *Table 7: Output interface connection (J6)*

| ***J6 (PINs output)*** | ***Output type*** | ***Description*** |
|---|---|---|
| 1~6 | PIN1~PIN6 (output open collector) | Provides 90 mA (each pin) output driver. Maximum voltage is 40 V |

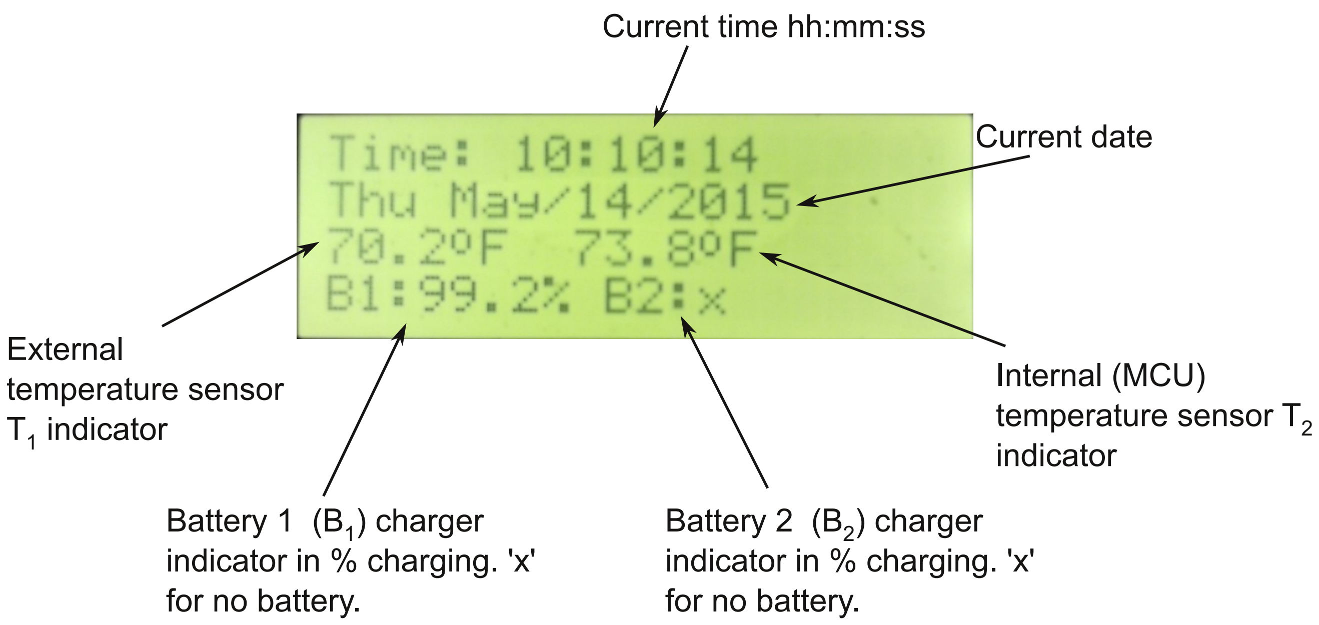
### *Table 8: Output interface connection (J7)*

| ***J7 (Output interface)*** | ***Output type*** | ***Description*** |
|---|---|---|
| 1 | GND | Ground |
| 2 | NC | No connection |
| 3 | NC | No connection |
| 4 | SPK | Speaker |
| 5 | CHARGER 1 | Battery Charger 1 output (47 mA inicial charger current) |
| 6 | CHARGER 2 | Battery Charger 2 output (47 mA inicial charger current) |
| 7 | GND | Ground |

## Turning on this Embedded System

After turn on the system will shown in LCD display the following message (Figure 3).

**Figure 3: Embedded System Display**



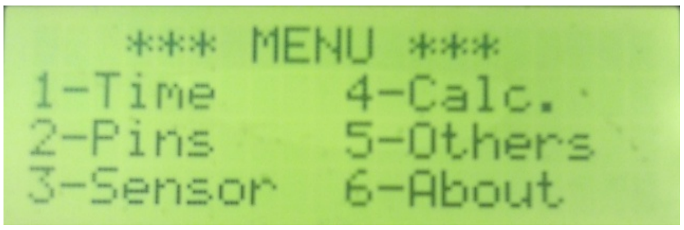
```
Time: 10:10:14
Thu May/14/2015
70.2ºF  73.8ºF
B1:99.2% B2:x
```

Current time hh:mm:ss
Current date
External temperature sensor $T_1$ indicator
Internal (MCU) temperature sensor $T_2$ indicator
Battery 1 ($B_1$) charger indicator in % charging. 'x' for no battery.
Battery 2 ($B_2$) charger indicator in % charging. 'x' for no battery.

## Accessing Menu for settings

When pressing <MENU> button will shown 6 (six) options. Press 1 to access Time adjust, weekday (calendar) or alarm, 2 for pins configuration, 3 for detailed sensor measures, 4 for default or scientific calculator, 5 for (others) language support, temperature sensor, and configure UART for IoT, security, UART parity and user password. Finally press 6 for about firmware and hardware information. Full access at UART is limited. Only ROOT has full access by keyboard or remote control.

**Figure 4: Embedded System inicial MENU**

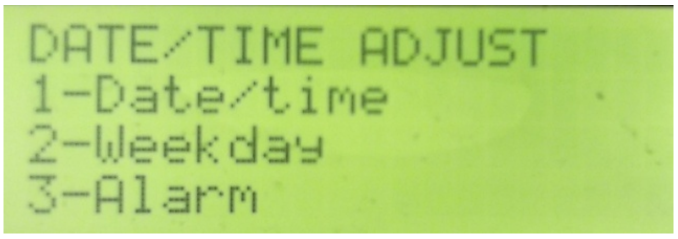


```
   *** MENU ***
1-Time     4-Calc.
2-Pins     5-Others
3-Sensor   6-About
```

## Setting date and time

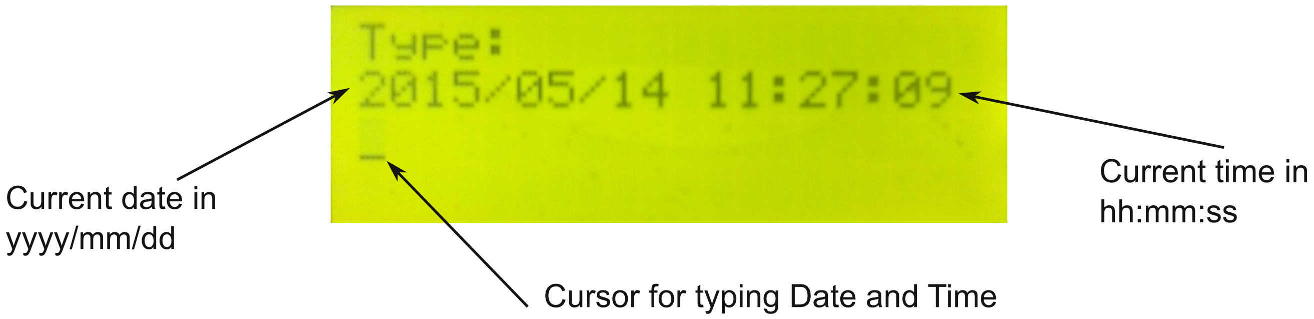
To set date/time press <MENU> buttom and press 1 (one) for Time and 1 (one) again to set Date/Time adjust.

**Figure 5: DATE and TIME ADJUST**



DATE/TIME ADJUST
1-Date/time
2-Weekday
3-Alarm

**Figure 6: Date and time adjust example**

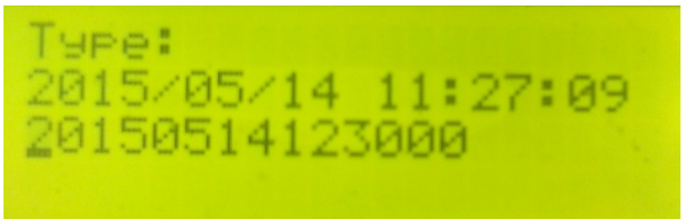


Type:
2015/05/14 11:27:09
_

## Example for Date/Time Adjust

If you want to set Time and Date to May, 14/2015 12:30:00 you must to type 20150514123000 and press <OK> button. You must type yyyymmddhhmmss where yyyy is current year, mm is current month, dd is current day, hh is current hour, mm is current minute and finally ss is current second. If you type anything wrong just press <CLEAR> button to clean Date/Time Adust. For year range is 1900 to 2099.

**Figure 7: EXAMPLE: Setting time and date to 12:30:00 May 14, 2015**

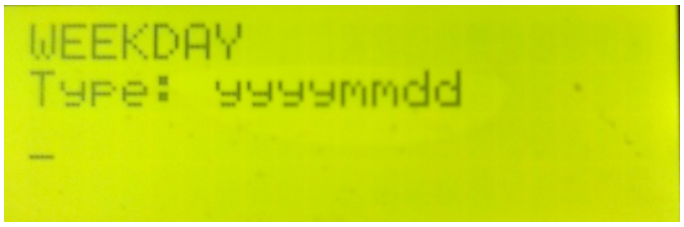


Type:
2015/05/14 11:27:09
20150514123000

## Weekday (calendar)

To access Weekday press <MENU> -> press <1> (Time) -> press <2> (Weekday). Display will show a cursor for Weekday command (figure 7 below).

**Figure 7: Calendar (weekday)**



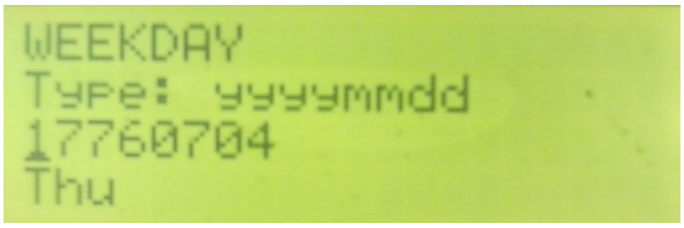
```
WEEKDAY
Type: yyyymmdd
_
```

You must type yyyymmdd for yyyy year mm for month and dd for day

<u>Example:</u>

Which day was July, 4 1776? If you want to know, you must type 17760704 and press <OK> button. The answer is (Thu). It was on Thuesday.

**Figure 8: Calendar example date (July, 6 1776)**



```
WEEKDAY
Type: yyyymmdd
17760704
Thu
```

## Alarm Setup

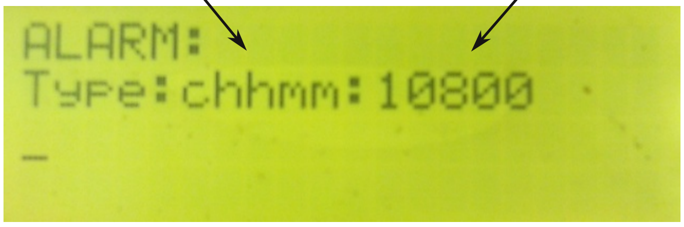
To configure alarm just press <MENU> -> 1 (Time) -> and finally press 3 (alarm) the following message will appear at  LCD screen.

**Figure 9: Alarm setup**

*Command type: chhmm:*

Where:
*c = 0* (alarm disabled)*; c=1* (alarm enabled)
*hh* for hour and *mm* for minutes

Last alarm in EEPROM memory

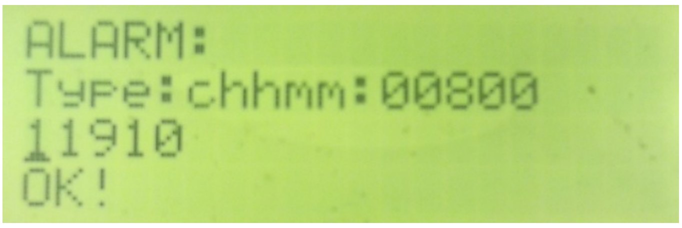


```
ALARM:
Type:chhmm:10800
_
```

Example 1:

If you type 11910 the alarm will set for 19:10.

**Figure 10: EXAMPLE: Setting alarm**

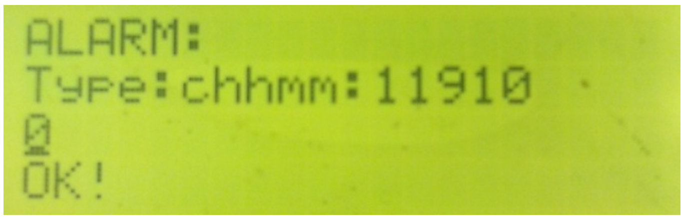


```
ALARM:
Type:chhmm:00800
11910
OK!
```

Example 2:

To turn off alarm just type 0 and press <OK>.

**Figure 11: EXAMPLE: Reset alarm**

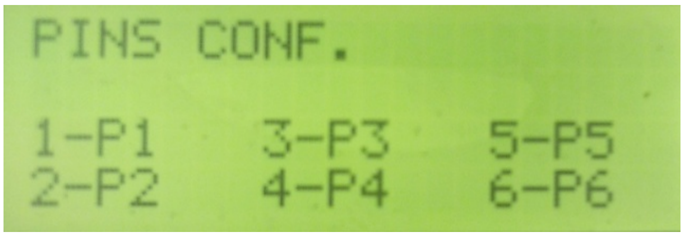


```
ALARM:
Type:chhmm:11910
0
OK!
```

## *Configuring PIN's*

This Embedded System has 6 (six) PIN's that can be controlled individually by remote control using Pronto HEX code, remotely by IoT using commands from UART for IoT, by onboard keyboard, by sensors (4 sensors) or time. Each output PIN has two state (ON/OFF) and is open collector port that supports maximum voltage of 40 V and can drive maximum input current of 90 mA.

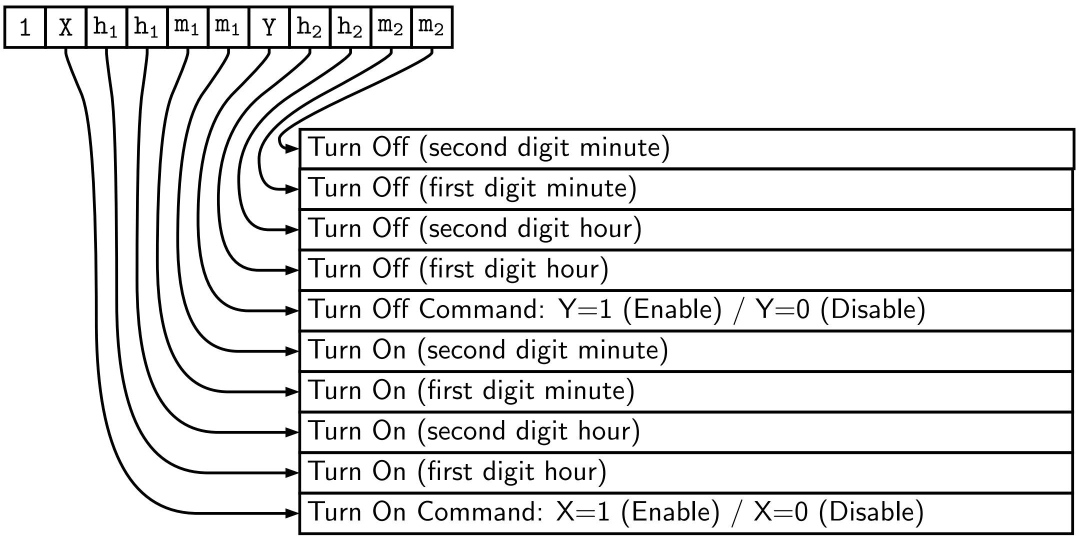
**Figure 12: PIN's configuration menu**



```
PINS CONF.

1-P1     3-P3     5-P5
2-P2     4-P4     6-P6
```

• ***Timer command:***

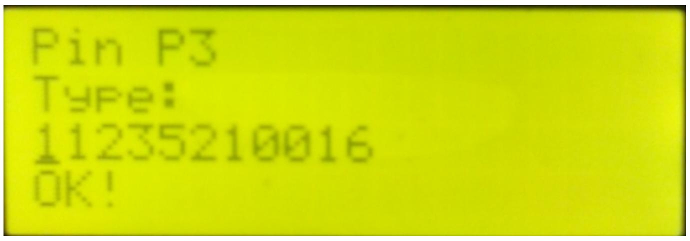
To configure any pins to turn ON and/or OFF by timer you must type the following command *1 X $h_1$ $h_1$ $m_1$ $m_1$ Y $h_2$ $h_2$ $m_2$ $m_2$* for desired pin(s), where:


1
X
$h_1$
$h_1$
$m_1$
$m_1$
Y
$h_2$
$h_2$
$m_2$
$m_2$
Turn Off (second digit minute)
Turn Off (first digit minute)
Turn Off (second digit hour)
Turn Off (first digit hour)
Turn Off Command: Y=1 (Enable) / Y=0 (Disable)
Turn On (second digit minute)
Turn On (first digit minute)
Turn On (second digit hour)
Turn On (first digit hour)
Turn On Command: X=1 (Enable) / X=0 (Disable)

<u>Example 1 for timer command:</u>

If you type 11235210016 and press <OK> button, the pin (in this example PIN 3 in figure 13 below) will always turn ON at 23:52 and always turn OFF at 00:16.

**Figure 13: Timer command Example 1**

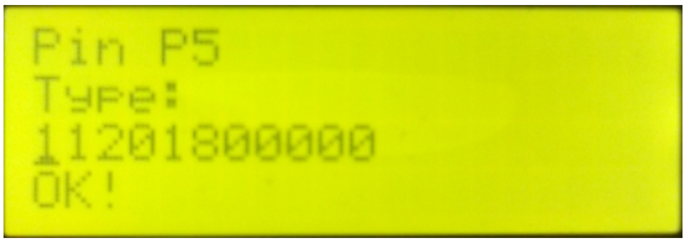


```
Pin P3
Type:
11235210016
OK!
```

*<u>Example 2 for timer command:</u>*

If you type 11201800000 and press <OK> button, the pin (in this example PIN 5 in figure 14 below) will always turn ON at 20:18 (only). Y=0 (turn OFF command is DISABLED)

**Figure 14: Timer command Example 2**

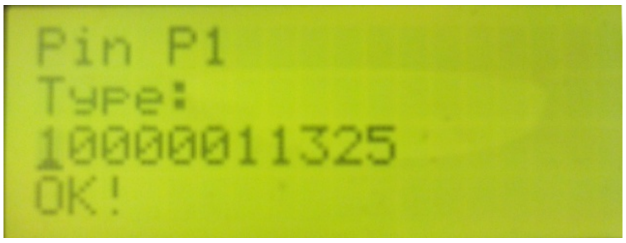


```
Pin P5
Type:
11201800000
OK!
```

*Example 3 for timer command:*

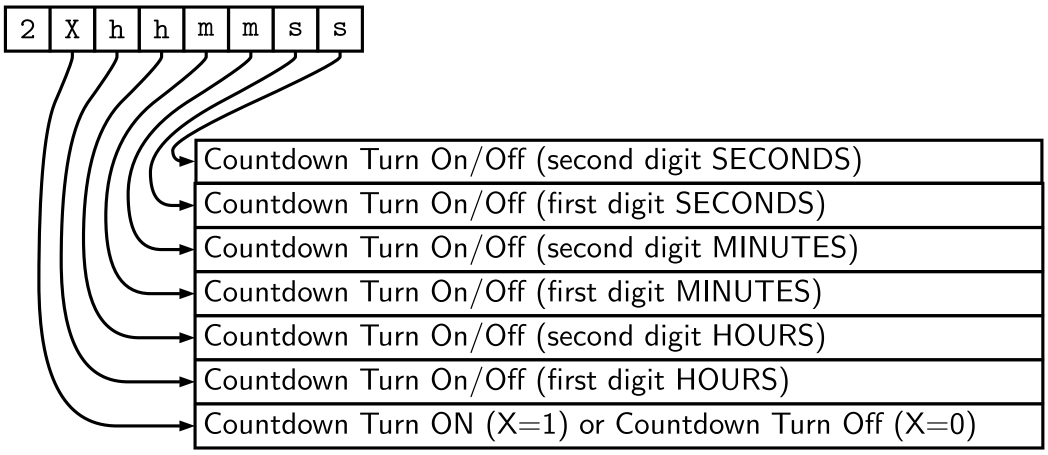
If you type 10000011325 and press <OK> button, the pin (in this example PIN 1 in figure 15) will always turn OFF at 13:25 (only). X=0 (turn ON command is DISABLED).

**Figure 15: Timer command Example 3**


```
Pin P1
Type:
10000011325
OK!
```

• ***Countdown timer command:***

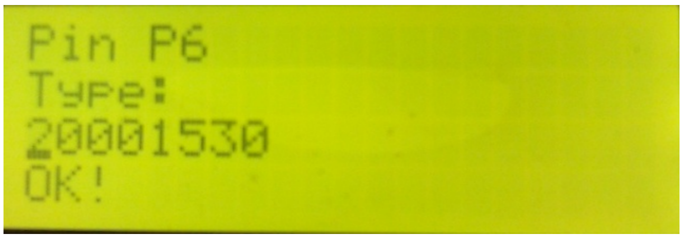
To configure any pins to turn ON and/or OFF by countdown just type the following command *2 X h h m m s s* for desired pin(s), where:


2
X
h
h
m
m
s
s
Countdown Turn On/Off (second digit SECONDS)
Countdown Turn On/Off (first digit SECONDS)
Countdown Turn On/Off (second digit MINUTES)
Countdown Turn On/Off (first digit MINUTES)
Countdown Turn On/Off (second digit HOURS)
Countdown Turn On/Off (first digit HOURS)
Countdown Turn ON (X=1) or Countdown Turn Off (X=0)

*Example 1 for countdown timer command:*

If you type 20001530 and press <OK> button, the pin (in this example PIN 6 in figure 16) will turn off after 15 minutes and 30 seconds (countdown timer).

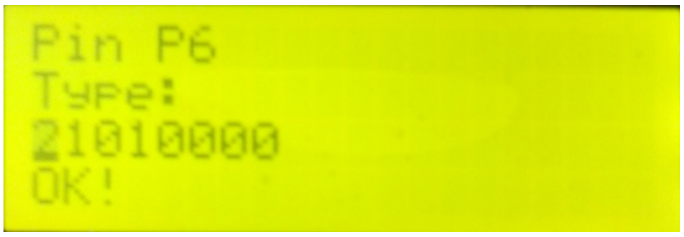
**Figure 16: Countdown timer command Example 1**


```
Pin P6
Type:
20001530
OK!
```

*Example 2 for countdown timer command:*

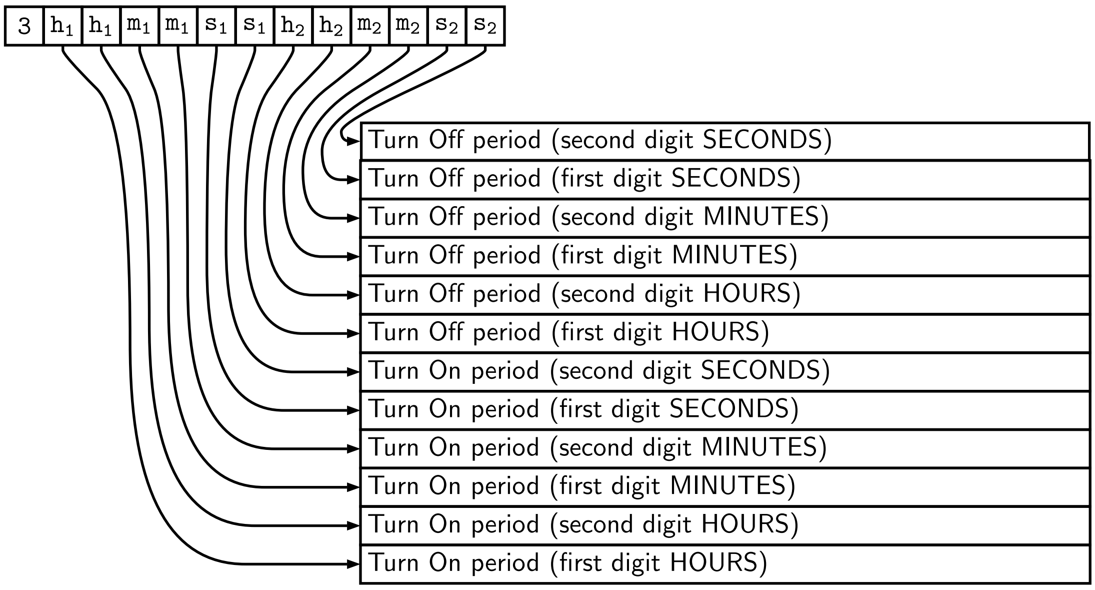
If you type 21010000 and press <OK> button, the pin (in this example PIN 6 in figure 17) will turn on after 1 hour (countdown timer). Maximun countdown timer operation is also programmable for 17h59m59s.

**Figure 17: Countdown timer command Example 2**


Pin P6
Type:
21010000
OK!

• ***Ciclic (periodic) command:***

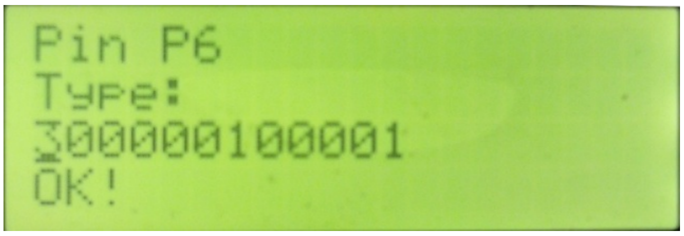
To configure any pins to turn ON and OFF periodicly, just type the following command $3\ h_1\ h_1\ m_1\ m_1\ s_1\ s_1\ h_2\ h_2\ m_2\ m_2\ s_2\ s_2$ for desired pin(s), where:


3
$h_1$
$h_1$
$m_1$
$m_1$
$s_1$
$s_1$
$h_2$
$h_2$
$m_2$
$m_2$
$s_2$
$s_2$
Turn Off period (second digit SECONDS)
Turn Off period (first digit SECONDS)
Turn Off period (second digit MINUTES)
Turn Off period (first digit MINUTES)
Turn Off period (second digit HOURS)
Turn Off period (first digit HOURS)
Turn On period (second digit SECONDS)
Turn On period (first digit SECONDS)
Turn On period (second digit MINUTES)
Turn On period (first digit MINUTES)
Turn On period (second digit HOURS)
Turn On period (first digit HOURS)

*Example:*

If you type 3000001000001 and press <OK> button, the pin (in this example PIN 6 in figure 18) will keep ON each 1 (one) second and will keep OFF each 1 (one) second. In other words, PIN 6 in this configuration oscillates at 1/2 Hertz. Maximun ciclic operation is also programmable for 17h59m59s.

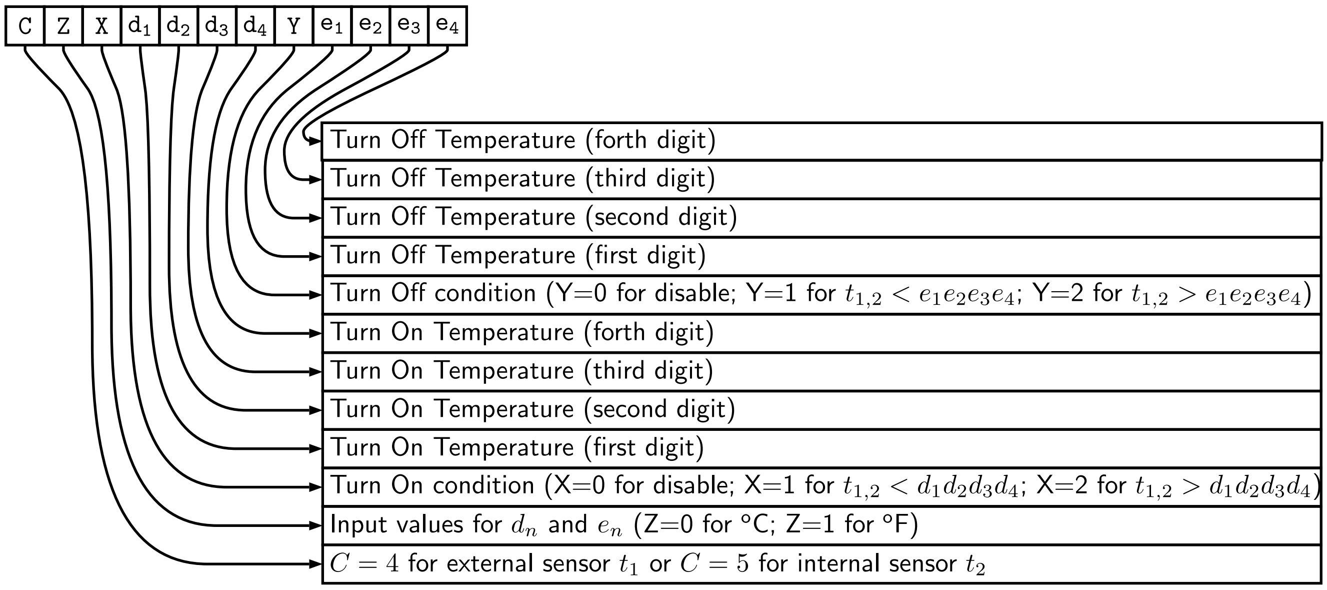
**Figure 18: Ciclic command example**



```
Pin P6
Type:
300000100001
OK!
```

## *Temperature sensor command ( $T_1$ and $T_2$ ):*

To configure any pins to turn ON and OFF by external temperature sensor ( $T_1$ ) or internal MCU temperature sensor ( $T_2$ ), just type the following command *C Z X* $d_1$ $d_2$ $d_3$ $d_4$ *Y* $e_1$ $e_2$ $e_3$ $e_4$, for desired pin(s), where:

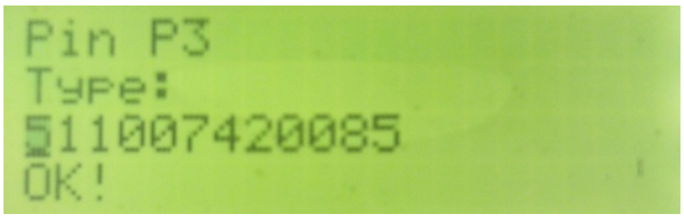


Turn Off Temperature (forth digit)
Turn Off Temperature (third digit)
Turn Off Temperature (second digit)
Turn Off Temperature (first digit)
Turn Off condition (Y=0 for disable; Y=1 for $t_{1,2} < e_1e_2e_3e_4$; Y=2 for $t_{1,2} > e_1e_2e_3e_4$)
Turn On Temperature (forth digit)
Turn On Temperature (third digit)
Turn On Temperature (second digit)
Turn On Temperature (first digit)
Turn On condition (X=0 for disable; X=1 for $t_{1,2} < d_1d_2d_3d_4$; X=2 for $t_{1,2} > d_1d_2d_3d_4$)
Input values for $d_n$ and $e_n$ (Z=0 for °C; Z=1 for °F)
$C = 4$ for external sensor $t_1$ or $C = 5$ for internal sensor $t_2$

*Example 1 for temperature command:*

If you type 511007420085 and press <OK> button, the pin (in this example PIN 3 in figure 19) will turn on if $T_2$ is less than 74 °F and will turn off if $T_2$ is greater than 85 °F.

**Figure 19: Temperature command Example 1**

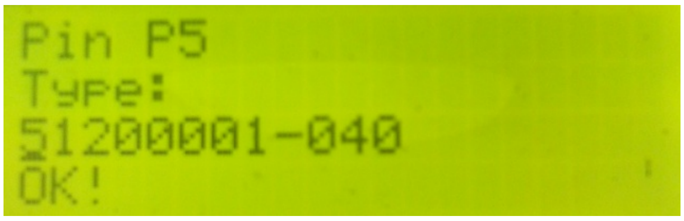


```
Pin P3
Type:
511007420085
OK!
```

*Example 2 for temperature command:*

If you type 51200001-040 and press <OK> button, the pin (in this example PIN 5 in figure 20) will turn on if $T_2$ is greater than 0 °F and will turn off if $T_2$ is less than -40 °F.

**Figure 20: Temperature command Example 2**

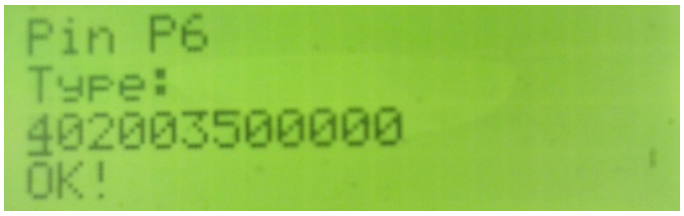


```
Pin P5
Type:
51200001-040
OK!
```

*Example 3 for temperature command:*

If you type 402003500000 and press <OK> button, the pin (in this example PIN 6 in figure 21) will turn on if $T_1$ is greater than 35 °C. NOTE: turn off condition (Y=0) is disable. Then turn on (only) is enabled.

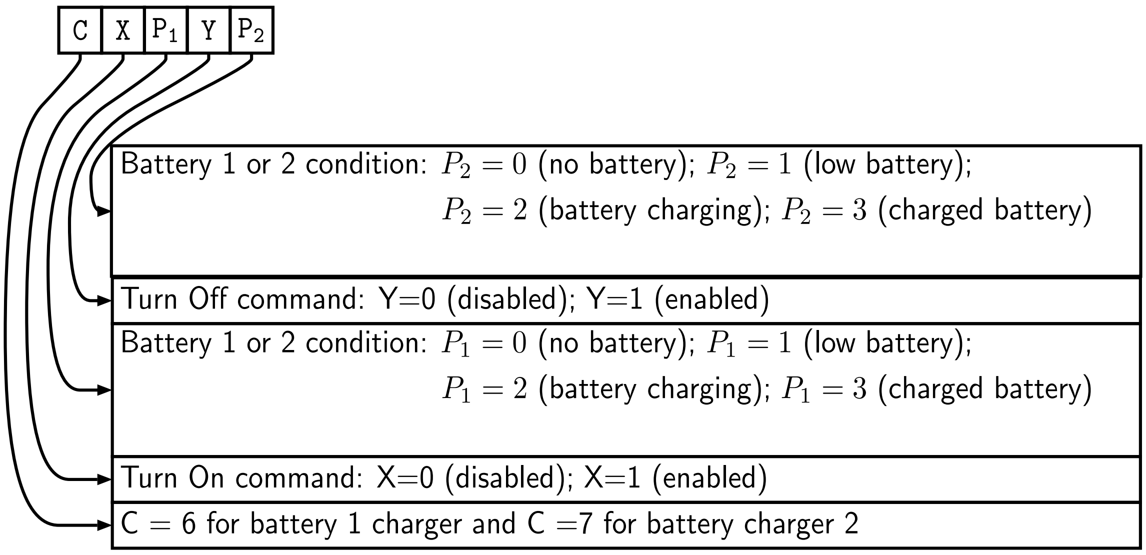
**Figure 21: Temperature command Example 3**



```
Pin P6
Type:
402003500000
OK!
```

## ***Battery charger sensor command ( $B_1$ and $B_2$ ):***

To configure any pins to turn ON and/or OFF by battery sensor $B_1$ or $B_2$, just type the following command.



| C | X | $P_1$ | Y | $P_2$ |
|---|---|---|---|---|

Battery 1 or 2 condition: $P_2 = 0$ (no battery); $P_2 = 1$ (low battery); $P_2 = 2$ (battery charging); $P_2 = 3$ (charged battery)

Turn Off command: Y=0 (disabled); Y=1 (enabled)

Battery 1 or 2 condition: $P_1 = 0$ (no battery); $P_1 = 1$ (low battery); $P_1 = 2$ (battery charging); $P_1 = 3$ (charged battery)

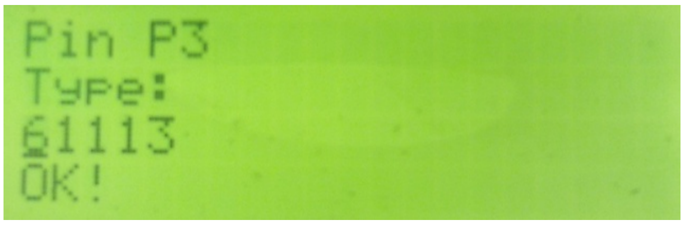
Turn On command: X=0 (disabled); X=1 (enabled)

C = 6 for battery 1 charger and C =7 for battery charger 2

*Example 1 for battery charge command:*

If you type 61113 and press <OK> button, the pin (in this example PIN 3 in figure 22) will turn on if battery in charger 1 is uncharged and will turn off if battery in charger 1 is charged.

**Figure 22: Battery charger example 1**

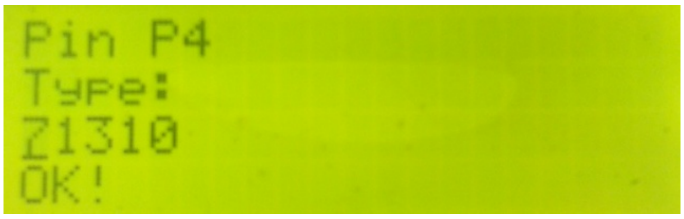


```
Pin P3
Type:
61113
OK!
```

*Example 2 for battery charger:*

If you type 71310 and press <OK> button, the pin (in this example PIN 4 in figure 23) will turn on if battery in charger 2 is charged and will turn off if battery in charger 2 is released.

**Figure 23: Battery charger example 2**

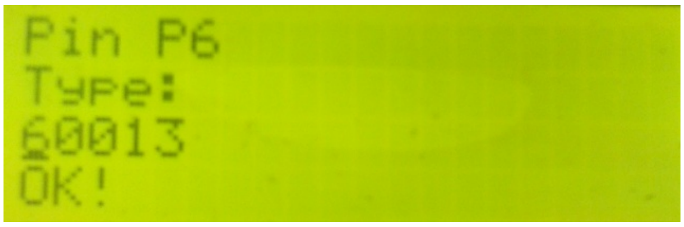


```
Pin P4
Type:
71310
OK!
```

*Example 3 for battery charger:*

If you type 60013 and press <OK> button, the pin (in this example PIN 6 in example 24) will turn off if battery in charger 1 is charged.

**Figure 24: Battery charger example 3**



```
Pin P6
Type:
60013
OK!
```

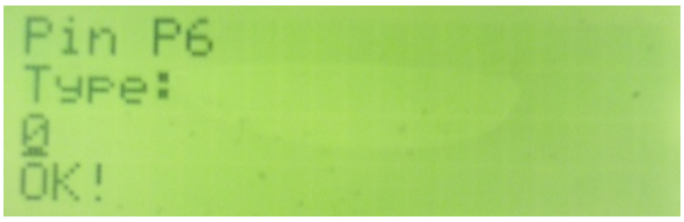
## *Resetting pin(s)*

If you want to reset any pin(s), just type 0 and press <OK>. Now pin will be controlled only by user from IoT, Infrared or Keyboard.

*Example:*

If you type 0 and press <OK>. The pin (in this example PIN 6 in figure 25) will be controlled by user via IoT, remote control or keyboard.

**Figure 25: Resetting pin(s)**

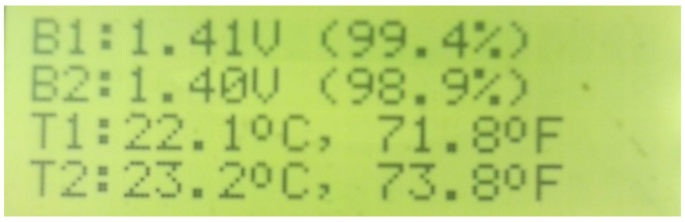


```
Pin P6
Type:
0
OK!
```

## *Sensor panel*

To access real time sensors press <MENU> and press <3> (Sensor). The system will provide measurements of four sensors (Battery 1 and 2 charger voltage, internal (MCU) and external temperature T1 and T2 respectively).

**Figure 26: Sensor panel (real time measuring)**



```
B1:1.41V (99.4%)
B2:1.40V (98.9%)
T1:22.1ºC, 71.8ºF
T2:23.2ºC, 73.8ºF
```

## *Calculator menu*

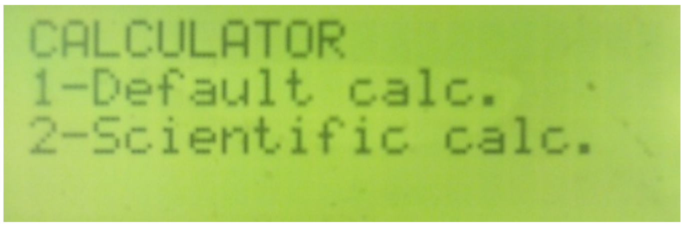
To access calculator option just type <MENU> -> <4> (Calc.). Default and Scientific calculator provides 3 (three) variables A (or 'x'), B and C where A (or 'x') is first value, B is second value and C is result of operation. Input and result values are in float point unit is (32 bits) or 7 digit precision.

## *Default calculator*

To access default calculator just press <MENU> -> <4> (Calc.) -> <1> (Default calculator).

In default calculator is possible perform 4 (four) math basic operations (+) addition, (-) subtraction, (*) multiply and (/) divide.

**Figure 27: Calculator menu**

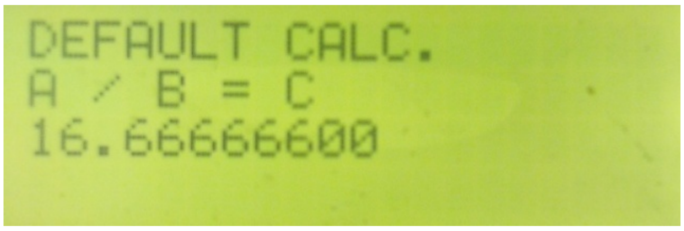


```
CALCULATOR
1-Default calc.
2-Scientific calc.
```

*Example*:

If you want divide 50 by 3 you type <5> -> <0> -> <enter> and then press <divide> and then press <3> and <enter> result is shown in figure 28.

**Figure 28: Basic calculator example**



```
DEFAULT CALC.
A / B = C
16.66666600
```

## *Scientific calculator*

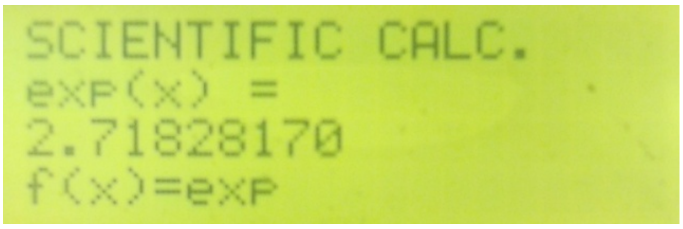
There are 23 math functions: *exp(x), ln(x), ln(x)/C, sqrt(x), cbrt(x), sin(x), cos(x), tan(x), sind(x), cosd(x), tand(x), asin(x), acos(x), atan(x), asind(x), acosd(x), atand(x), C<-A, hyp(C, A), mod(C,A), sinh(x), cosh(x)* and *tanh(x)*.

Some examples: Printing in the screen some basics values.

*Example 1 for scientific calculator:*

If you press <up> or <down> and select exp function and type <1> and press <enter>. Result is an approximation of *e = 2.7182817...*

**Figure 29: Scientific calculator example 1**

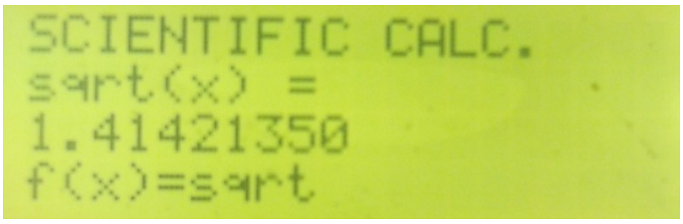


```
SCIENTIFIC CALC.
exp(x) =
2.71828170
f(x)=exp
```

*Example 2 for scientific calculator*

If you press <up> or <down> and select sqrt function and type <2> and press <enter>. Result is an approximation of $\sqrt{2} \approx 1.4142135...$

**Figure 30: Scientific calculator example 2**

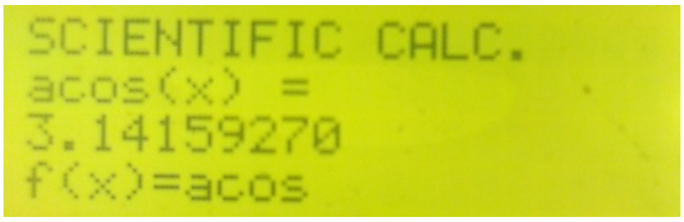


```
SCIENTIFIC CALC.
sqrt(x) =
1.41421350
f(x)=sqrt
```

Example 3 for scientific calculator

Calculating $\pi$ using trigonometric function in this case acos(-1)= $\pi$ = 3.1415927...

**Figure 31: Scientific calculator example 3**



```
SCIENTIFIC CALC.
acos(x) =
3.14159270
f(x)=acos
```

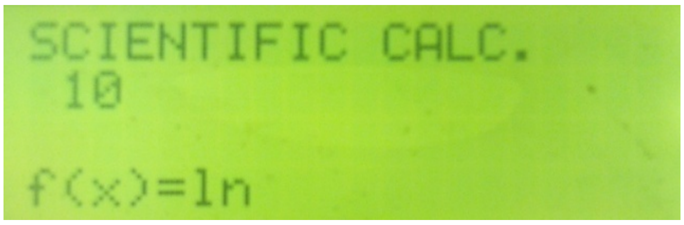
*Example 4 for scientific calculator*

But now, if you want to calculate $log_{10}(2)$? This scientific calculator does not provide such function but you can calculate easily using two steps by using one logarithm property:

step 1: Select *ln(x)* function and type *10* and press <enter>. Result, in this case *ln(10)* is stored in variable *C*.

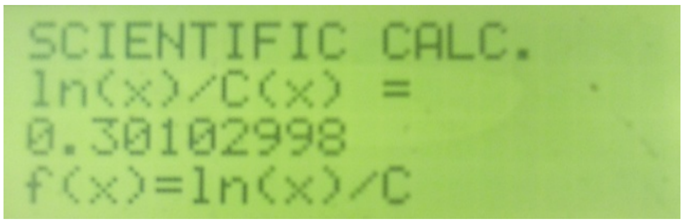
step 2: Select *ln(x)/C* function and type *2* and press <enter>. Result is an approximation of $log_{10}(2)=0.30102998...$

**Figure 32: Scientific calculator example 4 (step 1)**



```
SCIENTIFIC CALC.
 10

f(x)=ln
```

**Figure 33: Scientific calculator example 4 (step 2)**



```
SCIENTIFIC CALC.
ln(x)/C(x) =
0.30102998
f(x)=ln(x)/C
```

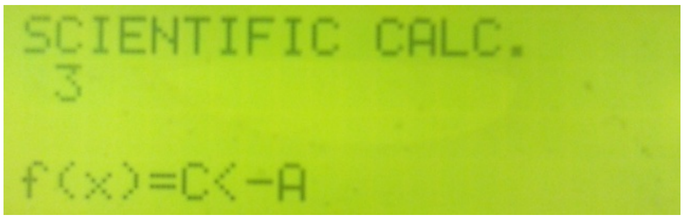
*<u>Example 5 for scientific calculator</u>*

But now, if you want to calculate hypotenuse of *3* and *6*? To calculate it you must call function *C<-A* and call *hyp(C, A)* using two steps because you have to pass two variables (*3*) and (*6*).

<u>step 1:</u> Select *C<-A* function and type *3* and press <enter>. Value *3* is stored in *C* variable.

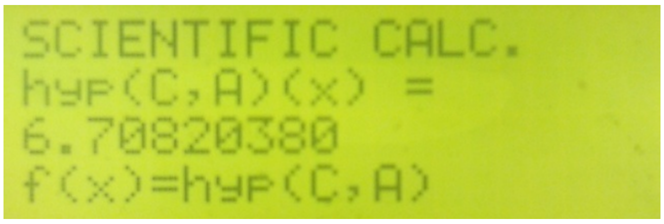
<u>step 2:</u> Select *hyp(C,A)* function and type 6 and press <enter>. Result is an approximation of *hyp(3,6)=6.7082038...*

**Figure 34: Scientific calculator example 5 (step 1)**



```
SCIENTIFIC CALC.
 3

f(x)=C<-A
```

**Figure 35: Scientific calculator example 5 (step 2)**

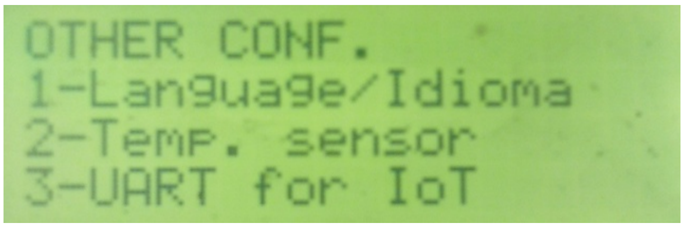


```
SCIENTIFIC CALC.
hyp(C,A)(x) =
6.70820380
f(x)=hyp(C,A)
```

## Other configuration

To access other configuration to have special access like language interface, change unity measurements, configure IoT and security you must type <MENU> -> <5> and then you will see  as figure below (figure 36).

**Figure 36: Other configurations (special options)**

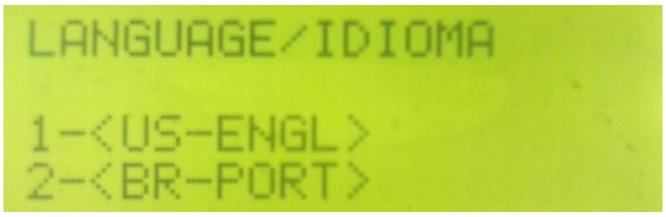


```
OTHER CONF.
1-Language/Idioma
2-Temp. sensor
3-UART for IoT
```

## Changing language system

To change language interface you must first access other functions menu. So you must type <MENU> -> <5> -> <1> and select your available language (US English) or (Brazilian Portuguese). Press <1> for English (US) or press <2> for Portuguese (BR).

**Figure 37: Selecting language interface**

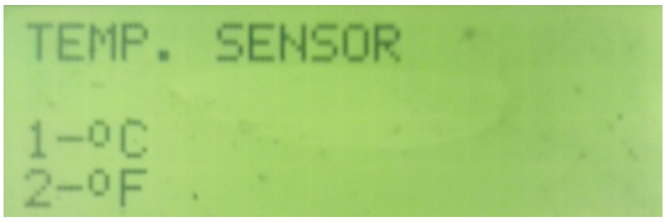


```
LANGUAGE/IDIOMA

1-<US-ENGL>
2-<BR-PORT>
```

## Temperature measurements

You can read temperature sensors T1 (external) and T2 (MCU internal) in (°C) degree Celsius (default) or in (°F) degree Fahrenheit. To modify both temperature indicator type <MENU> -> <5> -> <2>. Then choose 1 (for °C) or (2 for °F).

**Figure 38: Selecting real-time system measurement**



```
TEMP. SENSOR

1-ºC
2-ºF
```

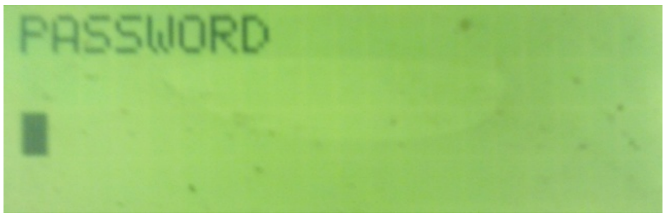
## *UART for IoT*

You can use UART for IoT to access this embedded system to control PINs output, read or write SRAM and EEPROM memories, LCD control, read sensors, reset MCU, configure hardware and read or set date and time remotely. To have entire or partial control for remote control via IoT you must access UART for IoT option. You can do it only locally (ROOT). For details see (UART for IoT user security).

To have an access to UART for IoT you must type <MENU>, <5> -> <3>.

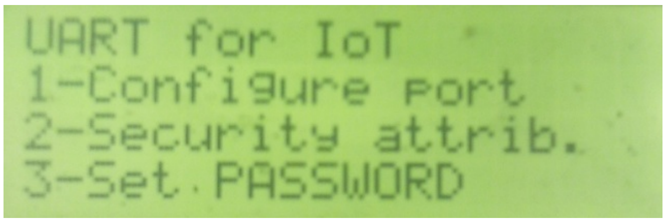
If you set a password then it will be requested every time you access UART for IoT menu. See setting password for details.

**Figure 39: Access UART for IoT (with password)**



```
PASSWORD

█
```

**Figure 40: Access UART for IoT (no password)**



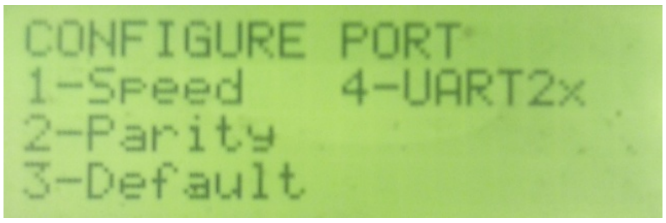
```
UART for IoT
1-Configure port
2-Security attrib.
3-Set.PASSWORD
```

In UART for IoT Menu, there are 3 (three) options: 1 (configure port), 2 (Security attributes), 3 (Set PASSWORD)

## *Configure port menu*

If you want to access UART for IoT you must before configure speed, parity, security attributes and user password (optional)

**Figure 41: Configure port menu**



```
CONFIGURE PORT
1-Speed     4-UART2x
2-Parity
3-Default
```

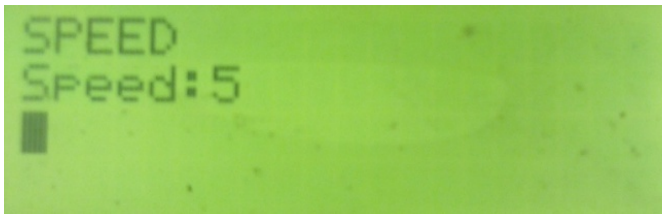
## *Setting speed to UART for IoT*

To set speed (baund rate) just type in UART for IoT menu <1> -> <1>. Default value is 5 (19600 bps). Table 9 below shows 16 values of UART speed available.

***Table 9: UART for IoT speed***

| ***Speed*** | ***Baund rate (bps)*** |
|---|---|
| **0** | 300 |
| **1** | 600 |
| **2** | 1200 |
| **3** | 2400 |
| **4** | 4800 |
| **5** | 9600 (default) |
| **6** | 14400 |
| **7** | 19200 |
| **8** | 28800 |
| **9** | 38400 |
| **10** | 50000 |
| **11** | 76800 |
| **12** | 230400 |
| **13** | 250000 |
| **14** | 500000 |
| **15** | 1000000 |

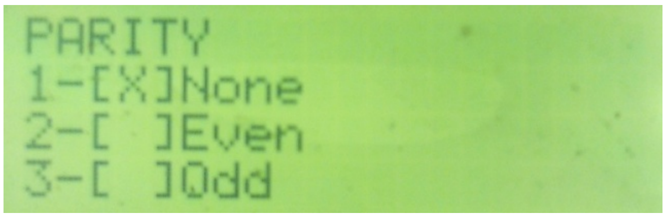
**Figure 42: Setting speed**


```
SPEED
Speed:5
```

## Parity settings

To set parity just type in UART for IoT menu <1> and <2>. There are 3 options available: 1- None (default), 2 - Even and 3 - Odd. User can modify these settings for error detection in UART communication port.

**Figure 43: Setting parity**

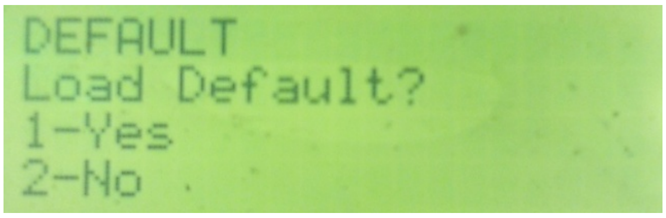


```
PARITY
1-[X]None
2-[ ]Even
3-[ ]Odd
```

## Loading defaut values

To load default values, just type in UART for IoT menu <1> and <3>. A message "Load default?" with two answer available will apear: 1- Yes and 2 - No. If you type 1 (Yes), all the configurations port will set to default (Speed = 5, Parity = None and UART2x=Disabled).

**Figure 44: Load default option**

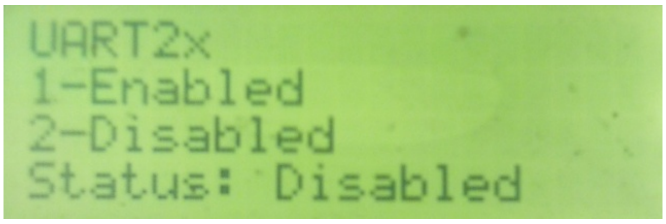


```
DEFAULT
Load Default?
1-Yes
2-No
```

## Enabling/Disabling UART2x

You can simply double (2x) speed communication to UART. To access UART2x press in UART for IoT menu <1> (Configure port) and <4> (UART2x). Press 1 to Enable UART2x or 2 to Disable UART2x. If UART2x is enabled, communication speed in Table 9 will be multiplied by 2 and the new range will be (600 to 2,000,000 bauds/s). Default value to UART2x = Disabled. UART for IoT PORT 2 will not work at highest speed due to hardware limitations. Even in PORT 1 errors may occur in high speed communication.

**Figure 45: Enable/Disable UART2x**



```
UART2x
1-Enabled
2-Disabled
Status: Disabled
```

## *UART for IoT user security*

User can control remote access from read/write and access command using permission configuration. For security access into UART for IoT menu type <2> (Security attributes) <1> (Permission).

You can access FLAG value (16 bits) stored in EEPROM at address 0x3FE (low bits) and 0x3FF (high bits). Each bit(s) has an especial attribute for access, read or write commands for UART for IoT. See Table 10 for permission bits details. User value is present in decimal and hexadecimal see figure 46 below.

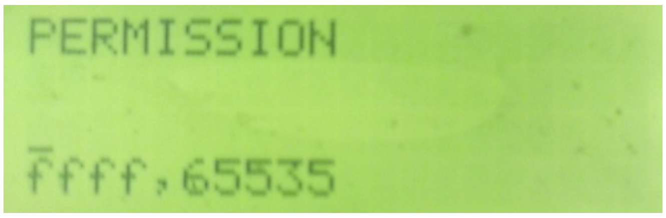
**Figure 46: 16 bit security flags for remote access**

| | 15 | | | | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Flag*** | *N* | *T* | *x* | $S_1$ | $S_0$ | *En* | *F* | *R* | $E_1$ | $E_0$ | $P_1$ | $P_0$ | *x* | *L* | $D_1$ | $D_0$ |
| | **MSB** | | | | | | | | | | | | | | | **LSB** |

### *Table 10: Flag bits function and description*

| ***Function*** | ***Description*** | ***Values*** |
|---|---|---|
| **x** | No function (reserved) | Don't care 0 or 1 |
| **N** | Read sensors | N=0b0 (access denied), N=0x1 (read all sensors) |
| **T** | Two wire access | T=0b0 (access denied), T=0b1 (Two wire read and write access) |
| **$S_1S_0$** | SRAM access | $S_1S_0$=0b00 (access denied), $S_1S_0$=0b01 (read only), $S_1S_0$=0b10 (write only), $S_1S_0$=0b11 (read/write) |
| **En** | UART for IoT access | En=0b0 (UART for IoT) disable, En=0b1 (UART for IoT) enebled. |
| **F** | Program (FLASH) (read only) | F=0b0 (access denied), F=0b1 (FLASH read only enabled) |
| **R** | Reset MCU | R=0b0 (reset MCU disabled), R=0b1 (reset MCU enabled) |
| **$E_1E_0$** | EEPROM access | $E_1E_0$=0b00 (access denied), $E_1E_0$=0b01 (read only), $E_1E_0$=0b10 (write only), $E_1E_0$=0b11 (read/write) |
| **$P_1P_0$** | PIN access | $P_1P_0$=0b00 (access denied), $P_1P_0$=0b01 (read only), $P_1P_0$=0b01 (write only), $P_1P_0$=0b11 (read/write) |
| **L** | LCD access/control | L=0b0 (access denied), L=0b1 (LCD control access/control) |
| **$D_1D_0$** | Date and time access | $D_1D_0$=0b00 (access denied), $D_1D_0$=0b01 (read only), $D_1D_0$=0b10 (write only), $D_1D_0$=0b11 (read write) |

Figure 47: Setting IoT security



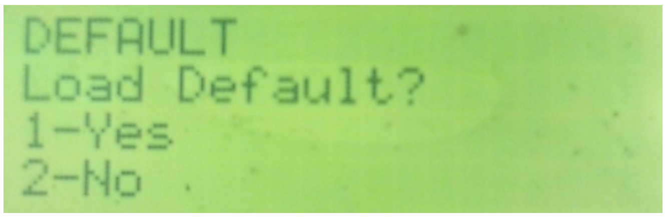
```
PERMISSION

ffff,65535
```

## *UART for IoT user security (load default)*

For default values of security attributes type into UART for IoT menu 2-> (Security attributes) 2- Default. Will apear "Load default?" question. Then type 1 for Yes (Load default) or 2 (No) to keep user security settings.

Default value is FLAG=33873 (0x8451). This means UART for IoT is enabled for read only Pins, Date-time, EEPROM memory and user can access all sensors.

Figure 48: Load default option

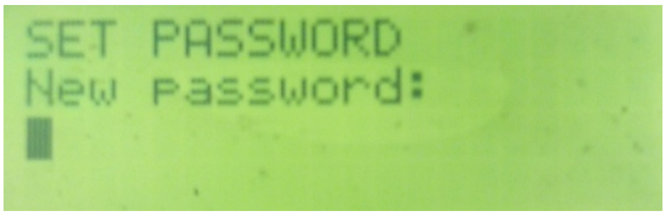


```
DEFAULT
Load Default?
1-Yes
2-No
```

## *Setting user PASSWORD*

For managing (set, modify, clear) user password type into UART for IoT menu 3-> Set PASSWORD.

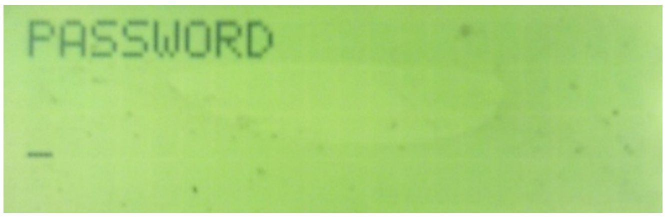
Figure 49: Setting user PASSWORD



```
SET PASSWORD
New password:
```

If there is no Password or it's your first time setting password then will appear a message below (Figure 50). And then you will need to type the old password.

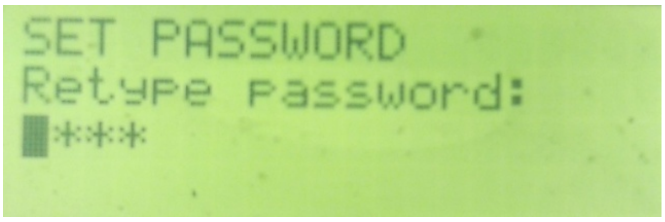
Figure 50: Setting a new PASSWORD



```
PASSWORD

_
```

Type your new password and press <ENTER>. A confirm message password will appear and type <ENTER>. User password now is set. Now, you will need password to access UART for IoT.

Figure 51: Retyping a new PASSWORD

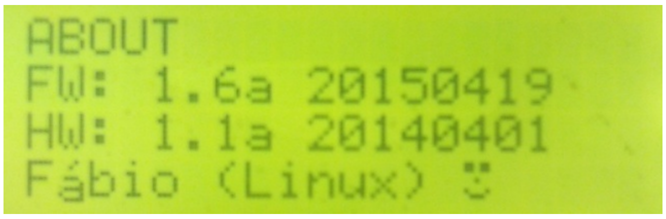


```
SET PASSWORD
Retype password:
█***
```

## *Information about firmware and hardware*

If you type <MENU> -> <6> you have access to information about current firmware and hardware date and version. Firmware is upgradable by ISP (In Serial Programming) interface.

Figure 52: About information



```
ABOUT
FW: 1.6a 20150419
HW: 1.1a 20140401
Fábio (Linux) ☺
```

## *Characteristics*

**CPU architecture:** *AVR 8-bit RISC*

**CPU speed:** *16 MHz*

**Total memory:** *32 kB (Flash)*

**Total SRAM memory:** *2 kB*

**Total  EEPROM memory:** *1 kB*

**Interrupts:** *6 interrupts (3 external + 3 internal)*

**Hardware Version:** *1.1a (20140401)*

**Firmware Version:** *1.5a (20150419) (upgradable via ISP interface)*

**Languages support:** *English (USA) / Portuguese (Brazil)*

**Temperature unity measurements:** *°F (degree Fahrenheit) or °C (degree Celsius)*

**Operation voltages:**

*8 to 38 Vdc*

*Lithium Battery 3.3 V (optional) for Real Time Clock*

*Industry  and Home Version*:100Vac - 820Vac (45-450Hz) or 142Vdc - 1160Vdc*

**A Very High Voltage Range Switching Mode Power Supply for Embedded System. See documentation for details. This power supply uses only 3 (three) transistors e no integrated circuits.*

| Code IR | Pronto Hex code |
|---|---|
| 0 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001<br>002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c |
| 1 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 0030 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| 2 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017<br>0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| 3 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017<br>0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| 4 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001<br>002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| 5 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017<br>0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| 6 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| 7 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017<br>0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| 8 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| 9 | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| .<br>(dot) | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c |
| +/- | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017<br>0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| <enter> | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001<br>002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c |
| Up<br>(+) | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c |
| Down<br>(-) | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001<br>002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| Left<br>(/) | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001<br>002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| Right<br>(*) | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017<br>0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c<br>0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001<br>002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |

| Code IR | Pronto Hex code (continued) |
|---|---|
| Menu | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c |
| Return | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c |
| Clear | |
| A <- C | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |
| Beep On-Off | 0000 0066 0000 006b 0168 00b3 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0001 002c 0017 0017 0017 0017 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c 0017 0017 0001 002c |

## UART for IoT (Summary)

### Keys commands

| Command | Description | Code (HEX) | Response (on success) | Response (on error) |
|---|---|---|---|---|
| Send '0' | Sends '0' key | [0x30] | [ACK] | [NACK] |
| Send '1' | Sends '1' key | [0x31] | [ACK] | [NACK] |
| Send '2' | Sends '2' key | [0x32] | [ACK] | [NACK] |
| Send '3' | Sends '3' key | [0x33] | [ACK] | [NACK] |
| Send '4' | Sends '4' key | [0x34] | [ACK] | [NACK] |
| Send '5' | Sends '5' key | [0x35] | [ACK] | [NACK] |
| Send '6' | Sends '6' key | [0x36] | [ACK] | [NACK] |
| Send '7' | Sends '7' key | [0x37] | [ACK] | [NACK] |
| Send '8' | Sends '8' key | [0x38] | [ACK] | [NACK] |
| Send '9' | Sends '9' key | [0x39] | [ACK] | [NACK] |
| Send [+/-] | Sends [+/-] key | [0x3A] | [ACK] | [NACK] |
| Send [.] | Sends [.] (dot) key | [0x3B] | [ACK] | [NACK] |
| Send [UP or +] | Sends [UP or +] key | [0x3C] | [ACK] | [NACK] |
| Send [Down or -] | Sends [Down or -] key | [0x3D] | [ACK] | [NACK] |
| Send [Right or *] | Sends [Right or *] key | [0x3E] | [ACK] | [NACK] |
| Send [Left or /] | Sends [Left or /] key | [0x3F] | [ACK] | [NACK] |
| Send [ENTER] | Sends [ENTER] key | [0x40] | [ACK] | [NACK] |
| Send [MENU] | Sends [MENU] key | [0x41] | [ACK] | [NACK] |
| Send [RETURN] | Sends [RETURN] key | [0x42] | [ACK] | [NACK] |
| Send [CLEAR] | Sends [CLEAR] key | [0x43] | [ACK] | [NACK] |
| Send [A<-C] | Sends [A<-C] key | [0x44] | [ACK] | [NACK] |
| Send [BEEP/OFF] | Sends [BEEP ON/OFF] key | [0x45] | [ACK] | [NACK] |

## Hardware/Peripheral information and control

| Command | Description | Code (HEX) | Response (on  success) | Response (on error) |
|---|---|---|---|---|
| AUTHOR E-MAIL | Sends author e-mail | [0x6C] | [ACK] [AUTHOR E-MAIL] | [NACK] |
| AUTHOR NAME | Sends author name | [0x68] | [ACK] [AUTHOR NAME] | [NACK] |
| AUTHOR USER | Sends author user | [0x53] | [ACK] [AUTHOR USER] | [NACK] |
| BEEP TEST | Beep Test | [0x62] | [ACK] | [NACK] |
| FIRMWARE VER. | Sends the version of Firmware | [0x56] | [ACK] [FIRMWARE VER.] | [NACK] |
| HARDWARE VER. | Sends the version of Hardware | [0x57] | [ACK] [HARDWARE VER.] | [NACK] |
| READ PIN(S) | Read(s) pin(s) information | [0x70] | [ACK] [VALUE] | [NACK] [ERROR MSG] |
| READ SENSOR | Read(s) sensor(s) information (1) | [0x4E] [SENSOR No.] | [ACK] [VALUE] | [NACK] |
| RESET MCU | Reset the microcontroller | [0x7A] | [ACK] | [NACK] |
| WRITE PIN(S) | Write(s) pin(s) data (2) | [0x50] [VALUE] | [ACK] | [NACK] [ERROR MSG] |

(1) Where [SENSOR No.]=0x01 for reading external temperature sensor in °C or °F, [SENSOR No.]=0x02 for reading internal MCU temperature sensor in °C or °F, [SENSOR No.]=0x04 for reading B1 battery voltage (in volts) and [SENSOR No.]=0x08 for readingB2 battery voltage (in volts).

(2) Where [VALUE] = [$P_6P_5P_4P_3P_2P_1$XX], where $P_n$=$Pin_n$ (output pins) and XX are don't care bits.

## Date/Time control

| Command | Description | Code (HEX) | Response (on success) | Response (on error) |
|---|---|---|---|---|
| READ DATE | Sends date in ASCII format | [0x74] | [ACK] [y][y][y][y][m][m][d][d][w][w] | [NACK] |
| READ TIME | Send time in ASCII format | [0x69] | [ACK] [h] [h] [m] [m] [s] [s] | [NACK] |
| WRITE DATE | Writes date in BCD format (3) | [0x46] [y1] [y0] [m] [d] | [ACK] | [NACK] [ERROR MESSAGE] |
| WRITE TIME | Writes time in BCD format (3) | [0x54] [hh] [mm] [ss] | [ACK] | [NACK] [ERROR MESSAGE] |

(3) Year range from (1900 to 2099) => [y1][y0]=[0x1900 .. 0x2099], month range from 1 (january) to 12 (december) => [m]=[0x01 .. 0x12], day range from 1 to 31 [d]=[0x01 .. 0x31], hour range from 00 to 23 => [hh]=[0x00 .. 0x23], minutes range from (0 to 59) [mm]=[0x00 .. 0x59] and finally seconds range from 0 to 59 => [ss]=[0x00 .. 0x59]. All these values are in BCD format. Example: If you want to set date to July 6, 2015 just send the following command: [0x46][0x20][0x15][0x07][0x06].

*I2C communication access*

| ***Command*** | ***Description*** | ***Code (HEX)*** | ***Response (on success)*** | ***Response (on error)*** |
|---|---|---|---|---|
| I²C RECEIVE | Receive I²C data | [0x76] | [ACK] [DATA] | [NACK] [ERROR MSG.] |
| I²C SEND START | Sends START command to I²C | [0x73] | [ACK] | [NACK] [ERROR MSG.] |
| I²C SEND STOP | Sends STOP command to I²C | [0x6F] | [ACK] | [NACK] [ERROR MSG.] |
| I²C TRANSMIT | Transmits I²C data | [0x64] [VALUE] | [ACK] | [NACK] [ERROR MSG.] |

*Memory read/write access*

| ***Command*** | ***Description*** | ***Code (HEX)*** | ***Response (on success)*** | ***Response (on error)*** |
|---|---|---|---|---|
| READ EEPROM | Reads EEPROM memory | [0x72] [ADDR_H] [ADDR_L] | [ACK] [VALUE] | [NACK] [ERROR MSG.] |
| READ FLASH | Reads FLASH memory | [0x4C] [ADDR_H] [ADDR_L] | [ACK] [VALUE] | [NACK] [ERROR MSG.] |
| READ SRAM | Reads SRAM memory | [0x6D] [ADDR_H] [ADDR_L] | [ACK] [VALUE] | [NACK] [ERROR MSG.] |
| WRITE EEPROM | Writes EEPROM value | [0x52] [ADDR_H] [ADDR_L] [VALUE] | [ACK] | [NACK] [ERROR MSG.] |
| WRITE SRAM | Writes SRAM value | [0x4D] [ADDR_H] [ADDR_L] [VALUE] | [ACK] | [NACK] [ERROR MSG.] |

***LCD control***

| ***Command*** | ***Description*** | ***Code (HEX)*** | ***Response (on success)*** | ***Response (on error)*** |
|---|---|---|---|---|
| ENTER LCD MODE | Enter LCD mode | [0x63] | [ACK] [MSG.] | [NACK] [ERROR MSG.] |
| LCD CLEAR LINE | Clear line in LCD | [0x49] [LINE] | [ACK] | [NACK] [ERROR MSG.] |
| LCD CLEAR SCREEN | Clear LCD screen | [0x6E] | [ACK] | [NACK] [ERROR MSG.] |
| LCD CONTROL | Control LCD (4) | [0x75] [CONTROL] | [ACK] | [NACK] [ERROR MSG.] |
| LCD DATA | Sends a data to LCD (4) | [0x61] [DATA] | [ACK] | [NACK] [ERROR MSG.] |
| EXIT LCD MODE | Exits LCD mode | [0x4F] | [ACK] | [NACK] [ERROR MSG.] |
| LCD RETURN HOME | Return HOME | [0x48] | [ACK] | [NACK] [ERROR MSG.] |
| LCD SET CURSOR | Set cursor at position [cccccrrr] (5) | [0x55] [cccccrrr] | [ACK] | [NACK] [ERROR MSG.] |

(4) Refer to HD 44780 (LCD panel) datasheet for data/control for complete details.

(5) [0x55][cccccrrr]Where [rrr] are rows from [1 to 4], and [ccccc] are columns from [1 to 20]. Example. If you want set cursor at (2, 15) just send command: the following command: [0x55][0x7A] where, [0x7A]=[0b01111010].

***Bug fixes (firmware)***

| ***Bug #*** | ***Description*** | ***Fixed date*** |
|---|---|---|
| 1 | READ SENSOR must be 2 digit | 08/26/2015 |
| 2 | READ SENSOR always returns NACK even when data success | 08/26/2015 |
| 3 | WRITE DATE is not working | 08/26/2015 |
| 4 | WRITE TIME is not working and crashes data | 08/26/2015 |
| 5 | RECEIVE DATA sometimes halts microcontroller | 08/26/2015 |
| 6 | Could not read SRAM when access is allowed | 07/23/2015 |
| 7 | Could not read FLASH when access is allowed | 07/23/2015 |
| 8 | Could not load DWORD value from FLASH | 08/26/2014 |
| 9 | Could not load byte from FLASH with index | 08/26/2014 |
| 10 | Function BCD_to_uint8 returns less than zero | 08/27/2015 |
| 11 | Wrong value in function get_month_day | 10/01/2015 |
| 12 | else statement is not compiling. Error in avr-gcc | 10/02/2015 |
| 13 | Turn-on/off setup error when programming | 05/14/2015 |
| 14 | Error when checking user password | 07/27/2015 |
| 15 | (SECURUTY) must digest user password before write | 07/12/2015 |
| 16 | Error in SetCursor function | 07/28/2015 |
| 17 | (SECURITY) error at function is_null_pass | 07/18/2015 |
| 18 | Error in function UART_println | 08/26/2015 |
| 19 | Error in check_uart_error. Old data should be flushed | 08/28/2015 |