emonFrontEndCM.ino

This sketch can only receive data by radio if the data is sent using the “RFM69 Native Format”. Therefore all sensor nodes (e.g. emonTx, emonTH) that send data to the emonPiCM must use a sketch that transmits in this format. Those sketches include “rfm69n” in the name.

The emonPiCM front-end sketch - description

The sketch that runs in the Atmel ATMega 328P primarily provides the energy monitoring function and handles the radio traffic coming in from other sensor nodes.

It uses emonLibCM, set up for two current channels, to obtain and process current and voltage readings and, from external sensors, temperature readings and the pulse input.

The radio traffic is almost completely processed within the RFM69CW radio module, all the sketch needs to do is retrieve the complete message when it becomes available. It uses the SPI driver spi.h to interface to a patched version of the JeeLabs driver rf69.h. When a complete message is available, a call to rf.receive( ) will return a non-zero length, and the message is formatted and sent via the serial connection to the Raspberry Pi by print_frame( ).

Likewise, when a set of readings is available from emonLibCM, EmonLibCM_Ready( ) returns true and the power values (or assumed apparent powers calculated from the nominal voltage and measured current) are loaded into the emonpi structure and similarly passed to the Raspberry Pi by send_emonpi_serial().

It is now possible to fully calibrate the emonPi. Calibration commands come in via the same serial interface from the Raspberry Pi, and are handled by getCalibration( ) (in a separate source file).

It’s also possible to transmit a message – that function is handled by the outbound r.f. data handler.

Configuration, calibration and the latest energy values are stored in the EEPROM memory of the ATMega 328P. The task of saving the data to, and retrieving the data from, the EEPROM is handled in the configuration source file by calling functions from the oemEProm library.

Configuration & on-line calibration

On-line configuration is locked for the first 3 minutes after the front-end sketch has started in order to allow the Raspberry Pi to start, without any data emitted to the serial port affecting the configuration or calibration.

If the emonPi does not appear in emonCMS the very first time it is powered up (following setting up the Wi-Fi parameters etc), restart the energy monitoring front end with a **short** press of the reset button adjacent to the CT1 connector.

The following setup/configuration commands are available:

```
l          - list the config values (terse)
L          - list the config values (verbose)
r          - restore sketch defaults & restart
s          - save config to EEPROM
v          - show firmware version
V          - verbose mode - changes etc echoed to serial output
b<n>       - set r.f. band n = a single numeral: 4 = 433MHz, 8 = 868MHz,
               9 = 915MHz (may require hardware change)
p<nn>      - set the r.f. power. nn - an integer 0 - 31 representing
               -18 dBm to +13 dBm. Default: 25 (+7 dBm)
g<nnn>     - set Network Group nnn - an integer (OEM default = 210)
n<nn>      - set node ID n= an integer (standard node ids are 1..30)
c<n>       - enable current & power factor values to serial output for
               calibration. n = 0 for OFF, n = 1 for ON,
d<xx.x>    - xx.x = a floating point number for the datalogging period
k<x> <yy.y> <zz.z>
           - Calibrate an analogue input channel:
              x = a single numeral: 0 = voltage calibration, 1 = ct1
               calibration, 2 = ct2 calibration, etc
              yy.y = a floating point number for the voltage/current
               calibration constant
              zz.z = a floating point number for the phase calibration for
               this c.t. (z is not needed, or ignored if supplied, when x =
               0)
                e.g. k0 256.8
                     k1 90.9 2.00
f<xx>      - xx = the line frequency in Hz: normally either 50 or 60
a<xx.x>    - xx.x = a floating point number for the assumed voltage if no
               a.c. is detected
m<x> <yy> - meter pulse counting:
                x = 0 for all pulses OFF, x = 1 for counter 1 ON, x = 2 for
               counter 2 ON, <yy> = an integer for the pulse minimum period
               in ms. (y is not needed, or ignored when x = 0)
t0 <y>     - turn temperature measurement on or off:
                y = 0 for OFF, y = 1 for ON, y = 2 ON & search for sensors
t<x> <yy> <yy> <yy> <yy> <yy> <yy> <yy> <yy>
                change a temperature sensor's address or position:
                x = a single numeral: the position of the sensor in the
               list (1-based)
                yy = 8 hexadecimal bytes representing the sensor's address
                  e.g.  28 81 43 31 07 00 00 D9
                  N.B. Sensors CANNOT be added beyond the array size.
T<ccc>\n   - transmit a string of alpha-numeric characters.
w<n>       - set the radio on or off: n = 0 for OFF, n = 1 for Receive ON,
               n = 2 for Transmit ON, n = 3 for both ON
z          - set the energy values and pulse count(s) to zero
?          - show this text again
```

The command will be acknowledged, and confirmation given when saving the settings, only after verbose mode has been set. When you change one or more of the settings, the change will take effect immediately and you might see the displayed values change. (See below for the temperature sensors, which are different.)

Option (‘s’) will save all the changes. If you do not do this, the settings will revert to the previous values at the next restart. After you save (‘s’) the changes, the new settings will be used forever, or until changed again.

If you restore the sketch default values (‘r’), all the EEPROM data is ignored and the sketch restarts immediately, using the values set in the sketch. There is then no means of recovering the EEPROM data.

If the EEPROM has been used previously and has had non-compliant values written, the EEPROM content will be ignored and the sketch will start using its own default values. Saving the configuration will format the EEPROM and the sketch’s set values will be stored.

Adding or moving Temperature Sensors

Sensors can be added, or moved around, without stopping continuous monitoring of power and energy, but temperature readings must be disabled. If you are indifferent as to the order of the sensors, then all you need to do is stop temperature monitoring with ‘t0 0‘ followed by ‘t0 1’ to restart it. The sensors will be detected and read. If you need to preserve the order which they appear, save the configuration to EEPROM with ‘s’.

If you need to preserve the order of existing sensors, or put new sensors into a specific order, then the procedure is:

> Make a note of the addresses (serial numbers) and the order of any existing sensors with the verbose ‘V’ then list ‘L’ commands.
>
> Stop temperature monitoring with ‘t0 0’
>
> Connect the new/replacement sensor(s) and enable temperature monitoring with ‘t0 1’, list the sensors again.
>
> You will probably find the order of sensors is not what you want. You can put the sensors into the order you need by typing in the address of each. Stop temperature monitoring with ‘t0 0’, then type each sensor address into the position it needs to occupy, e.g:
>
> ‘t1 28 81 43 31 07 00 00 D9’
>
> ‘t2 28 8D A5 C7 05 00 00 D5’
>
> (‘28’ in the first byte is the signature of a DS18B20.) Enable temperature monitoring with ‘t0 1’, list the sensors and check that all is correct and that they are reading as expected.
>
> Save the configuration to EEPROM with ‘s’