Project – Digital Energy Meter

Digital energy meter overview. Functional description. PC interface. Software development.

Introduction

An energy meter (also colloquially known as a kWh meter) is a device that measures the total energy consumption, over a certain time interval, on an electricity service. A digital energy meter (DEM) is a device that implements this function with an embedded system – they are better than an electromechanical meter because they are more accurate, are able to have a completely customisable *tariff* (i.e. a cost of each unit of electricity at a certain time) and are able to be set and interrogated remotely by a Supervisory Control and Data Acquisition (SCADA) system.

Average power in a single-phase AC system is given by $P = VI\cos\phi$, where V is the RMS voltage, I is the RMS current, and ϕ is the phase angle between the voltage and current. One way to determine the energy is by sampling the voltage and current waveforms simultaneously and computing the instantaneous power directly with $p = vi$. You can then determine the energy consumed over one period by $E = \int_0^{T_0} p\,dt \approx \sum_{k=0}^{n} p_k T_s$. This energy figure for one period is then accumulated. The cost of the electricity is also computed on a "per period" basis, based on an appropriate tariff, and accumulated.

DEM Overview

The DEM is a real-time system. It needs to retrieve waveform amplitude information ("samples") from an analog-to-digital converter at precise intervals, do calculations with those samples, and perform timing operations. It needs to start operating virtually instantly upon startup, has to be cheap since there are millions required, and be robust – thus a PC is unsuitable. It is relatively easy to develop such a stand-alone real-time embedded system.

We will therefore exploit the GUI of Windows® and the real-time capability of a microcontroller to develop the DEM, as shown below:

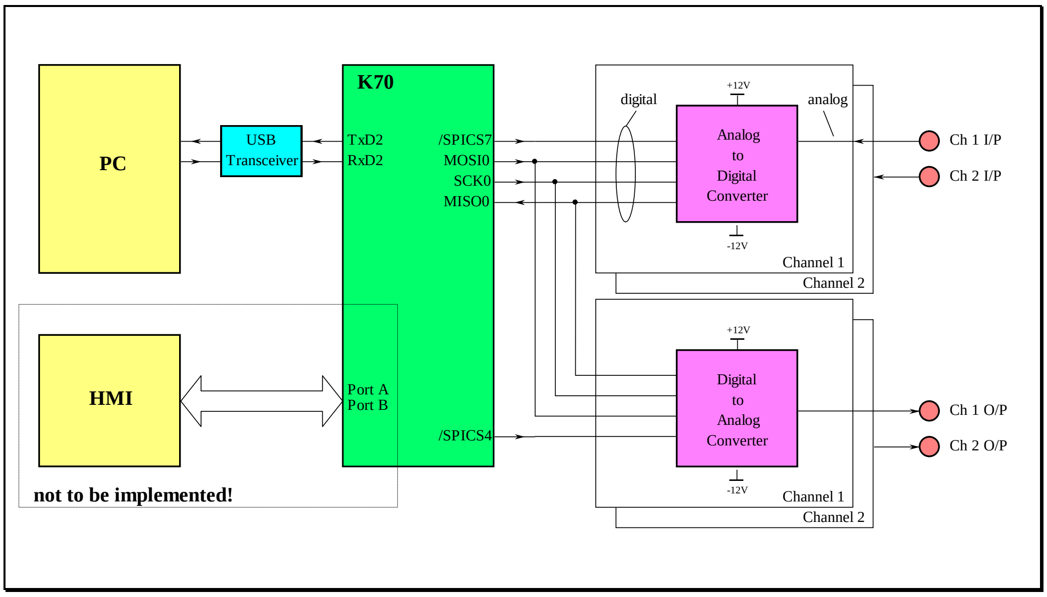

Figure P.1 – DEM Block Diagram

The PC is responsible for providing a user interface for the DEM.

A USB interface is the primary means of communication with the DEM. One of the six available UARTs on the μC is used for this purpose (UART2). The operation of the DEM is controlled and monitored from a PC using the Tower protocol.

The μC is responsible for communications with the PC and for retrieving information about 2 independent voltages from the external analog-to-digital converter (ADC) via a Serial Peripheral Interface (SPI). The input voltages are assumed to have been derived from the voltage and current in the electricity service by additional signal conditioning circuitry which is not shown.

The μC is also responsible for generating voltage and current waveforms that can be used for self-test and calibration. There are two outputs for this purpose which are generated by the digital-to-analog converter (DAC) via the SPI.

The ADC circuitry converts the voltages appearing on the external connectors from an analog signal into a digital value. The DAC circuitry converts a digital output value into an analog voltage which appears on an external connector. **Both the ADC and the DAC are 16-bit, bipolar, with a range of ±10 V.**

Functional Description

Functional Requirements

The DEM can be programmed to support different tariffs (cost/unit of electricity), and should provide usage statistics of metering time, total energy, average power and total energy cost. The DEM also communicates with a PC, which allows the updating of tariffs and monitoring of the DEM.

The DEM must meet the following functional requirements:

Specification	**Value**
Tariffs	Settable as per Table P.2
Frequency Range	47.5 Hz to 52.5 Hz
Voltage Range	200-250 V RMS
Current Range	5 A RMS
Measurements	<u>Basic</u> – Total time, average power, total energy, total cost <u>Intermediate</u> – frequency, RMS voltage, RMS current, power factor.
Accuracy	5% (energy, power, cost and time)
Sample Period	16 samples per cycle minimum

Table P.1 – DEM Specifications

The μC needs to reliably take a sample value from the ADC every sample period. To do this, it needs to take into consideration any suspected interruptions such as PC communication and overhead such as SPI interfacing. You can ensure reliable sampling periodicity by setting up interrupt priorities appropriately.

Sample Timing

The time between samples is known as the sample period, and is usually denoted T_s. The value of T_s should be chosen so that there are at least 16 samples per cycle of the power system's voltage, which has a nominal frequency of 50 Hz, but which can vary between 47.5 Hz and 52.5 Hz.

Arithmetic Calculations

You are welcome to use floating-point calculations in your implementation, however they may not be the fastest implementation (it depends on what operations you are doing). For this application (and many others) they are not necessary with careful planning. The topic notes on fixed-point calculations should be studied carefully if you do not use floating-point calculations.

Note: Code with floating-point operations may not be optimal if speed of execution is part of the specification (and marking criteria).

Tariffs

The DEM should store the tariff in non-volatile memory.

The electricity tariffs are:

Tariff #	**Rates**			
	Non-ToU	Time of Use (ToU)		
	¢ / kWh	Peak ¢ / kWh	Shoulder ¢ / kWh	Off Peak ¢ / kWh
1		22.235	4.400	2.109
2	1.713			
3	4.100			

Table P.2 – Electricity Tariffs

<u>Tariff Notes:</u>

Peak period is from 14:00-20:00.

Shoulder period is from 07:00-14:00 and 20:00-22:00.

Off-Peak period is at all other times.

Input Signal Conditioning Circuitry

The input signals (which are all voltages) are to be applied to Channels 1 (voltage) and 2 (current) of the Tower inputs, which connect to the analog board’s analog-to-digital converter.

We may assume that the input signal conditioning circuitry is such that:

Channel	Use	Raw input	Output to ADC
1	Voltage	230 V RMS	2.30 V RMS
2	Current	1 A RMS	1 V RMS

Table P.3 – Input Signal Conditioning

Self-Test and Calibration

The output channels are used in the following manner:

Channel	Use
1	Voltage
2	Current

Table P.4 – Outputs for Self-Test and Calibration

These outputs can be fed directly into the Tower inputs, bypassing the input signal conditioning circuitry. The DEM should be able to generate self-test waveforms – e.g. a 2.3V RMS voltage (nominally at 50 Hz) as a voltage reference on Channel 1, and a variable amplitude and phase voltage waveform on Channel 2 to simulate current.

As a self-test feature, the DEM is able to operate the metering function with “accelerated time”, where 1 second is equivalent to 1 hour. This is useful to test the metering function with a Time of Use tariff.

In self-test mode, the user (via the PC interface), shall be able to change the voltage and current **amplitudes** and phase according to the following table:

	Number of Steps	**Step Size**	**Range**
Voltage	2 317	30.52 mV	282.8 V – 353.5 V
Current	23 170	305.2 µA	0 A – 7.072 A
Phase	32	5.625 °	-90 ° to +90 °

Human-Machine Interface

A single push button lets the user cycle through various displays of quantities stored in the DEM: metering time, average power, total energy and total cost.

The display cycle is shown below:

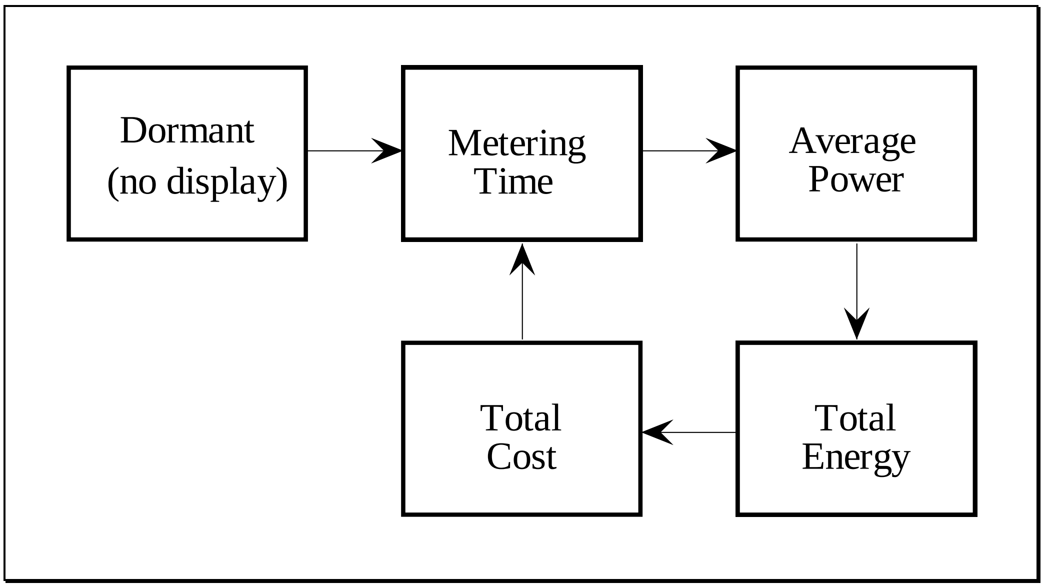

Figure P.2 – Display Cycling

Display States

There are 5 display states of the LCD display. The display states are cyclical, as shown in Figure P.2, and are advanced by a press of a push button, SW1.

The display is a real-time display, showing current values, and is updated once every second. If the user does not push a button within 15 seconds, then the device returns to the initial state of "dormant", with the display turning off.

State Name	Description
Dormant	The default startup state. The display is off.
Time	Shows the metering time in the format dd:hh:mm:ss. If the days exceed 99, then `xx:xx:xx:xx` is displayed.
Power	Shows the average power in kW in the format PPP.ppp.
Energy	Shows the total energy in kWh in the format EEE.eee. If the kWh exceed 999, then `xxx.xxx` is displayed.
Cost	Shows the total cost in the format $$$$.¢¢. If the cost exceeds $9999.99, then `xxxx.xx` is displayed.

Table P.5 – Display States

Note: We will not be implementing the HMI this session!

You can simply write the strings to UART2 where they will be picked up by the TowerPC interface, on the "Terminal" tab.

PC Interface

The remote interface to the DEM is via a PC running the Tower interface program under the Microsoft Windows® operating system. You are welcome to expand the Tower protocol for your own purposes.

Setting the DEM

The load tariffs used by the DEM should be able to be set via the PC interface. The DEM should store the tariffs in non-volatile memory. The PC should be able to set the time. The PC can also put the DEM into a self-test mode, where time is accelerated.

Interrogating the DEM

The PC should be able to interrogate the DEM for various quantities and settings, depending on the level of functionality.

Basic Protocol Extension for the DEM

Command Name	Command ID	Parameter 1	Parameter 2 (and 3 if needed)
Test mode	0x10	0 = disabled, 1 = enabled	not used (0x00)
Tariff	0x11	tariff (1-3)	not used (0x00)
Time1	0x12	seconds	minutes
Time2	0x13	hours	days
Power	0x14	power (W) – low byte	power (W) – high byte
Energy	0x15	$\text{kWh} \times 1000$ – low byte	$\text{kWh} \times 1000$ – high bytes
Cost	0x16	cost – cents	cost – dollars

Intermediate Protocol Extension for the DEM

Command Name	Command ID	Parameter 1	Parameter 2
Frequency	0x17	$\text{Hz} \times 10$ – low byte	$\text{Hz} \times 10$ – high byte
Voltage (RMS)	0x18	low byte	high byte
Current (RMS)	0x19	mA – low byte	mA – high byte
Power factor	0x1A	$\text{p.f.} \times 1000$ – low byte	$\text{p.f.} \times 1000$ – high byte

Software Development

There are two approaches to take for this simple embedded system – use a foreground / background approach, or use a real-time operating system (RTOS). The advantage of the foreground / background approach, for simple systems, is that it is easy to implement. For more complex systems, a real-time operating system simplifies the software design.

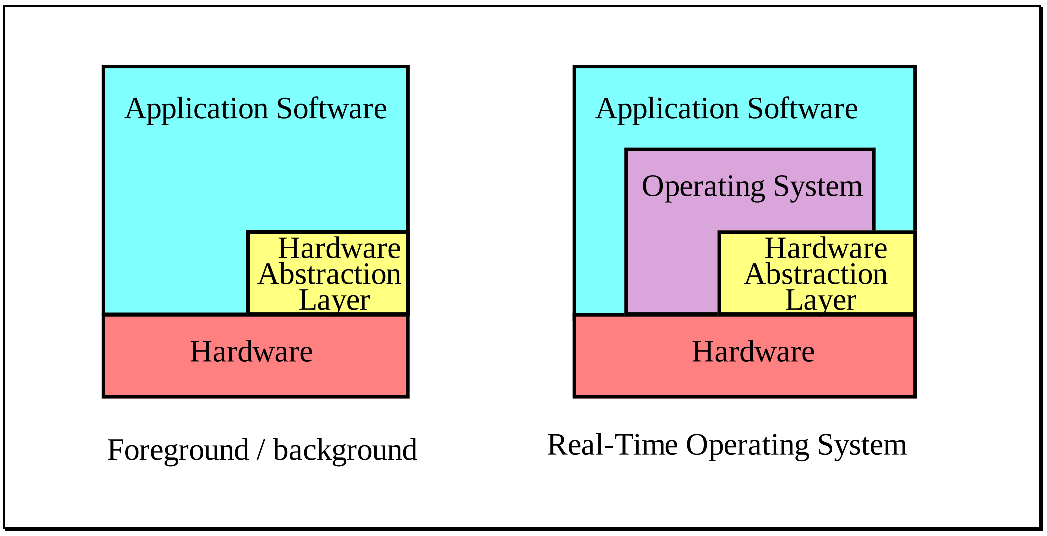

Figure P.3 – Two software approaches for an embedded system

For example, there are two inputs to the DEM. With a real-time operating system, only one piece of code (a program) needs to be written and tested – it is then used by two independent "threads". Management of time also becomes a lot easier with an RTOS. You will have the opportunity to use a simple RTOS.

RTOS

The RTOS is supplied as a library of object code with header files (no C source code is supplied – a typical scenario in industry). The documentation for the RTOS is supplied in a separate document.

Application

The DEM application code should exploit the RTOS capabilities. For example, threads could be created for: each of the input channels; communication with the PC interface program; the metering function, etc.

Strategy

The software should be developed in a modular fashion.

It is perfectly acceptable, and even advisable, to first get the DEM going using a foreground / background approach. It is important in many projects to get parts of a system up and running quickly as a proof of concept of overall system design.

Also consider the fact that in the ultimate system, which uses an RTOS, all the shared code needs to be “re-entrant”, and communication via global variables should be kept to a minimum. Any communication between threads via global variables will need to be carefully examined to see if semaphores are needed. Timing and priority of threads will also be an issue.

Consult the Project Marking Scheme so you can manage your time and focus on areas of code that are important from an assessment perspective.