Home Automation Board C & Assembly Strategy

Introduction

This document will suggest basic strategies for creating 'C' and Assembly code for the Home Automation E-Block. Because this E-Block can be used with a variety of upstream boards (e.g. PICmicro multi-programmer board), this document will not provide all of the information required. See the "further reading" section for more complete reference information.

Home automation has become a big topic in recent years, and the major standard to emerge is X10®. With this standard, communication signals are sent and received by the electrical wiring system within a building. This means that no wires are needed for 2 devices to 'talk' to each other.

The Home Automation E-Block has 2 connectors - a standard 9-way D-type socket which allows the board to be connected to any upstream E-Block, and a RJ-11 socket which must be attached to a XM10U (UK), XM10E (Europe) or TW523 (USA) 2-way TTL interface module which plugs into the electrical system of the building.

This E-Block requires 3 connections to the upstream board: Zero-crossing, Transmit and Receive. The Zero-crossing pin is an output from this E-Block (i.e. it should be connected to a microcontroller input pin) and this pin changes state each time the mains AC waveform crosses zero volts. This is important because all X10® data bits must begin transmission within a 200 microseconds of the AC voltage crossing zero. The XM10U (or equivalent) has delays of its own (up to 150us), so you should make sure that you react to this zero-crossing signal within 50us.

Each binary bit "1" is a 1ms modulated burst of 120kHz just after the zero-crossing. This modulation is performed by the XM10U, so a simple 1ms pulse should be sent by the firmware instead.

A typical X10® message consists of 22 bits of information, and so takes 11 AC power cycles to complete transmission (each bit of information is sent immediately after a zero-crossing). A message always begins with a start code (4 bits, always "1110") and is followed by a "house code" (4 bits, but is sent as 8 bits) and a "key code" or "function code" (5 bits, but sent as 10 bits). The house codes range from "A" to "P" and key codes from "1" to "16" - this means that there are 16 × 16 or 256 unique addresses.

All information except the start code is transmitted in "true compliment form" - this gives the protocol a certain amount of robustness. For example, the house code "A" is represented by the 4 byte code "0110". But this is transmitted as "01101001" (the first "01" represents the first bit "0" in the house code, then there is a "10" representing a "1", then another "10" for the next "1" and finally a "01" which represents the final "0" in the actual home code).

Each 22-bit message is actually sent twice to provide further robustness. This does mean that this protocol is quite slow, but this is generally not a problem in the applications where X10® is used.

Implementing a strategy

The following strategy is specific for a PICmicro microcontroller, but should be adaptable to any upstream device.

Waiting for a zero-crossing

To make sure each bit is sent straight after a zero-crossing when transmitting a message, your code should wait until a change of state of the zero-crossing input. You also need to do this when receiving a message. The following pseudo-code shows how to do this:

- read current state of zero-crossing input
- if it is high, then wait until it becomes low
- otherwise, wait until is becomes high

Message transmission

Important note: The transmitter circuit of this E-Block actually inverts the signal coming out of the PICmicro, so your firmware will need to set the transmit pin LOW when sending an logic "1", and set it HIGH when sending a logic "0". The transmit pin should stay HIGH when it is not transmitting a message.

As an example, this document will explain the techniques to send a message to switch on an appliance at address "B4". To do this, you must actually send 2 messages: "B4" followed by "B on". Also, remember that each message needs to be sent twice. The codes house, key and function codes for "B", "4" and "on" are "1110", "10100" and "00101" respectively.

Each message will start with the unique start code "1110" and the subsequent data bits will be send in "true compliment form". Each sent data bit needs to be sent within 50us of a zero-crossing.

The complete 22-bit message for "B4" is as follows:

```
1 1 1 0 1 0 1 0 1 0 0 1 1 0 0 1 1 0 0 1 0 1
(start)  1   1   1   0   1   0   1   0   0
```

Similarly, "B on" is:

```
1 1 1 0 1 0 1 0 1 0 0 1 0 1 0 1 1 0 0 1 1 0
(start)  1   1   1   0   0   0   1   0   1
```

After each message is sent, 6 zeros must be sent.

You may want to define each of these codes within your firmware,
e.g. `#define HouseB b'10101001'`

Or you could implement a look-up table or define an array of values of these codes.

A single bit "1" must be 1ms in length, and here's some pseudo-code to show how to send a single bit:

- Wait for a zero-crossing
- If the bit is "1", set "transmit" pin low
- Wait 1ms
- Set "transmit" pin high

When sending a single bit "0", wait for the zero-crossing and then maintain the "transmit" pin high.

Message reception

To receive a message, the firmware needs to continually monitor the "receive" pin. When this pin goes high, a message has been detected and needs to be read in. Remember that each bit will be read in as a 1ms-long pulse (or lack of for a logic zero!) which begins shortly after the zero-crossing. For each bit read, it is probably worth waiting for around 600us after the zero-crossing.

Once that first logic '1' has been detected, a further 3 bits must be read and the 4-bit result will be "1110" to indicate the start of a message. If it is not this value, then there has been an error reading the value.

If this start code has been read successfully, then the next 8 bits will represent the house code (in " true compliment form"). The following 10 bits will represent the house or function code (again, in " true compliment form").

Once a complete message has been read and decoded, the firmware needs to respond appropriately to that message.

Further reading

X10® is a well published communications protocol and more information about this standard can be found by searching the Internet. A good explanation can be found on the following webpage:

- http://www.x10.com/support/technology1.htm