In the latest generation of home automation systems, appliances can exchange information by transmitting data over the domestic mains wiring. As a result there is no need to install extra control cables and appliances can be connected to the "network" simply by plugging them into the nearest wall socket. Apart from the obvious saving in installation cost, this virtual network also makes modification and enhancement very simple since new devices just have a wall socket to be instantly connected to the network. X-10 is a communication protocol designed for sending signals over 230V AC wiring. Microcontrollers can easily be used in conjunction with X-10 technology to create home automation applications. The PIC16F877A can be selected for this application because of its versatility as a general purpose microcontroller, its FLASH program memory (for ease of development), data EEPROM, and ample I/O. Installation is simple, a transmitter plugs (or wires) in at one location in the home and sends its control signal (on, off, dim, bright, etc.) to a receiver which plugs (or wires) into another location in the home.
INTRODUCTION

Power line communication, also known as PLC, uses existing power distribution wires to communicate data. There are two main applications for power line communication - one for broadband Internet access to the home and the other for home and office networking. This work focuses on using power lines for home networking. Home networks typically use Ethernet or wireless devices. Ethernet provides high speed networking, but requires dedicated category 5 (CAT5) cabling which would need to be installed in the home. Wireless devices are now becoming more popular and work quite well, but provide speeds that are excessive for simple applications. Performance of wireless networks is also affected by line of sight obstructions such as walls. One major attraction of power line communication is the high availability of power outlets. “as long as there is a power socket, there is a connection to the network”. The high node availability is why this technology has tremendous market potential. Power line communication technology has been slow to evolve because the lines were designed solely for the purpose of 50Hz main power distribution. But after development of X-10 protocol for convenient transmission over power line, it became easy.
MOTIVATION FOR THIS WORK

In the past twenty years, data networks have gone from being an experimental technology to becoming a key tool for business and entertainment used by companies and homes worldwide. Home users, who often have more than one computer, are looking to data communication networks to share information between computers. They are also looking to networks for the ‘automation’ of their home – including applications such as security systems, network gaming, and controlling heating, air conditioning and other household appliances. The design of a network should consider several factors, of which the two most important are predicted network traffic and installation cost. The various types of traffic can have different throughput, data integrity, latency, and other requirements. A simple control system network that performs functions such as turning lights on and off, opening and closing the garage door, and controlling the air conditioner does not require high speeds. A high speed network would be much better utilized by a multiple computer network where there is a large amount of file and application sharing or video. The cost factor refers to the installation cost of a network. High speed networks often require more expensive equipment than low speed networks, so for low speed networks it is not economically smart to install high speed equipment. Installation cost is also affected by the actual setup of the network. Wireless equipment is becoming popular because it is simple to set up and provides high speed and high mobility (computers can access the network as long as they are within a certain distance of the access point). However, the wireless equipment may be too costly for low to medium speed applications. Another solution is to use dedicated network cabling but this is also a high cost solution because retrofitting a home with the required cabling becomes a time consuming and
expensive job. Also, once network cabling is installed in a home or office, it does not lend itself easily to reconfiguration – resulting in down time when location of network entities changes. What is missing is a medium speed technology that is low cost and allows for easy and ubiquitous network access. This project addresses a possible solution to the problem of mobility, ease of installation, and cost of networks by using the in-building power distribution system.

**OBJECTIVE**

1) To review previous studies on the power line as a transmission medium and previous and current methods of communicating data on the power line.
2) To take advantage of this information by studying a protocol.
3) To design and implement one of the application of power line communication named ‘DEVICE CONTROL USING X-10 PROTOCOL’

**BLOCK DIAGRAM**

1) **CONTROL TRANSMITTER**

![Control Transmitter Diagram](image)

---

IV
The above block diagram shows the master transmitter system in this project. It is used to control all the devices connected to it. The 4 operation switches connected is used to select house, device, function code and sending the data through powerline. The zero crossing detector will detect the zero crossing of 50Hz a.c. The microcontroller, PIC16F877, is programmed to perform all the transmitting functions. The LCD will display the functions selected by the operation switches. The PWM o/p will generate the 120 KHz carrier signal for data transmission.

2) RECEIVER

![Diagram](image)

Figure B

The above block diagram shows the output receiver system. It is used to receive the commands from the master transmitter. The zero crossing detector will detect the zero crossing of 50Hz a.c. The microcontroller, PIC16F877, is programmed to perform all the receiving functions. The envelope detector will detect the transmitted data.
HOW DOES THE X-10 PROTOCOL WORK?

The method used by X-10 is based on a simple data frame with eight data bits (one byte) preceded by a predetermined start code. The complicated part of this technology was not the system of binary data, but the method in which it was transmitted from one device (the transmitter) to another device (the receiver). The key was for every device to have an integral "zero crossing" detector so that all of them were synchronized together (figure 1). A receiver opens its receive "window" twice each sine wave (figure 2), that is 120 times each second or 7,200 times each minute.

Since these devices would not have any direct wiring between them, it was necessary to devise a way of sending the data over the existing electrical wiring. The actual binary data is transmitted by sending 1ms bursts of 120kHz just past the zero crossing of the 50Hz power. It was also obvious that complementary bit pairs were necessary. Therefore, a binary "1" was defined as the presence of a pulse, immediately followed by the absence of a
pulse. A binary "0" was defined as the absence of a pulse, immediately followed by the presence of a pulse (figure 3).

While the transmitted pulses were to be a full 1ms in duration, the receivers were designed to open a receive window of only .6ms. That allowed for the loose tolerances of the 1978-era components to "slop" plus/minus 200m sec. In order to provide a predictable start point (figure 4), every data frame would always begin with at least 6 leading clear zero crossings, then a start code of "pulse", "pulse", "pulse", "absence of a pulse" (or 1110). Once the Start Code has been transmitted, the first nibble is sent. (If you are not familiar with the term "nibble", that means 4 bits or half a byte.) In order to make it easier for the consumers to operate the devices, this first 4-bits were given "letter" code designations (figure 5). It was also decided to randomly rearrange the patterns so that the "A", "B", "C" codes,
etc., did not fall in the predicable binary pattern. It is easy to see that in reality, the "M" code is first in the binary progression.

In one contiguous bit stream, the second nibble provides the second half of the address (figure 6). The last bit appears to be a part of the "number" code but in reality it is a function bit. Whenever this function bit is a "0", it designates the preceding nibble as a number code and therefore a part of the address. For purposes of redundancy, reliability and to accommodate line repeaters, the X-10 protocol calls for every frame of data to be transmitted twice (figure 7).
Whenever the data changes from one address to another address, from an address to a command, from one command to another command or from one command to another command (figure 8), the data frames must be separated by at least 6 clear zero crossings (or "000000"). When teaching classes in this stuff, I often say that this gap "gives the receivers a chance to catch their breath". In reality, of course, the sequence of six "zero’s" resets the shift registers.
Once a receiver has processed its address data, it is ready to receive a command. As before, all data frames must begin with a start code. Then the following nibble gives the letter code (figure 9). The next nibble is the command. Since the last bit is the function bit ($b_f = 0 = \text{address number}, b_f = 1 = \text{command}$) all the commands end in a binary 1.

This diagram (figure 10) only shows the six most often used commands. A later graphic will illustrate all the available commands. As before, all X-10 protocol transmitters send their data frames twice (figure 11).
Figure 12 shows that an example transmission of two data frames (A1 A1 A-On A-On, for instance) would take 47 cycles of the 50Hz sine wave. That would equate to 0.7833 seconds, or in practical terms, just under 1 second. Of course, some commands take less time. When sending an "All-Lights-On" command, for example, no address needs to be sent. Therefore the entire two frame sequence takes only one third of a second (actually, 0.3666 seconds, but who’s quibbling). If your receivers react on the first frame, it could take a mere two tenths of a second (0.1833 seconds).
A standard X-10 transmission takes place over a duration of 47 cycles of the AC power.

Each eleven cycles (one start code, one letter code and one function code) is known as a “frame”.

Figure 12

X-10 CODES
X-10 data is usually sent in standard length "frames" each beginning with a start code designated as "1110". The next 4 bits represent a major group category and the following 4 bits represent either a unit designation or a command, the selection of which is determined by the last function bit. Every data frame is always transmitted twice (as illustrated below).

To improve reliability, every bit is sent with its complement (as shown below)

<table>
<thead>
<tr>
<th>Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>M 0000</td>
<td>ALL UNITS OFF</td>
</tr>
<tr>
<td>E 0001</td>
<td>ALL LTS ON</td>
</tr>
<tr>
<td>C 0010</td>
<td>ON</td>
</tr>
<tr>
<td>K 0011</td>
<td>OFF</td>
</tr>
<tr>
<td>O 0100</td>
<td>DIM</td>
</tr>
<tr>
<td>G 0101</td>
<td>BRIGHT</td>
</tr>
<tr>
<td>A 0110</td>
<td>ALL LIGHTS OFF</td>
</tr>
<tr>
<td>I 0111</td>
<td>EXT CODE 1</td>
</tr>
<tr>
<td>N 1000</td>
<td>HAIL REQUEST</td>
</tr>
<tr>
<td>F 1001</td>
<td>MAIL ACK</td>
</tr>
<tr>
<td>D 1010</td>
<td>EXT CODE 3</td>
</tr>
<tr>
<td>L 1011</td>
<td>UNUSED</td>
</tr>
<tr>
<td>P 1100</td>
<td>EXT CODE 2</td>
</tr>
<tr>
<td>H 1101</td>
<td>STATUS ON</td>
</tr>
<tr>
<td>B 1110</td>
<td>STATUS OFF</td>
</tr>
<tr>
<td>J 1111</td>
<td>STATUS REQ</td>
</tr>
</tbody>
</table>

Table 1

- Ext Code0111- Now designated as "Ext Code 1", for data and control
• Preset Dim (1)1010- Now designated as "Ext Code 3", for security messages
• Preset Dim (2)1011- Now designated as "Unused"
• Ext Data1100- Now designated as "Ext Code 2", for meter read and DSM

"Extended Code 1" has a defined frame length which is 31 cycles (62 bits) and is described as:

• **Start Code** = 4 bits,
• **Housecode** = 4 bits,
• **Extended code 1** = 5 bits (01111),
• **Unit code (device code)** = 4 bits,
• **Data** = 8 bits,
• **Command** = 8 bits.

The explanation for not having a defined frame length for the other two is:
"Extended code 2 is variable in length, depending on the type of message. It has its own separate "attention" marker to separate it from all other formats.
Extended code 3 has been "assigned" for security but doesn't actually exist yet so its format has not yet been defined."
Device control is one of the application of X-10 protocol. The above block diagram shows that in a simplified manner. There are two rooms consisting of several electrical devices connected to the power line. The device that requires connection to the power line also has access to a network. Every single room requires one receiver module for controlling devices in that particular room. It receives the data from the unique control transmitter and performs the required function. All these receiving functions are performed at the zero crossings of 50Hz ac.
**MICRCONTROLLER CORE FEATURES**

- High performance RISC cpu
- All single cycle instructions except program branch instructions which are two cycles
- Operating speed: DC-20MHz clock input
- Upto 8Kx14 words of FLASH Program Memory
- Upto 368x8 bytes of Data Memory (RAM)
- Upto 256x8 bytes of EEPROM Data Memory

**CIRCUIT DIAGRAM**

1) **CONTROL TRANSMITTER**
ZERO-CROSSING DETECTOR

In X-10, information is timed with the zero-crossings of the AC power. A zero-crossing detector is easily created by using the external interrupt on the RB0 pin and just one external component, a resistor, to limit the current into the PICmicro MCU (see Figure 3). In India, the peak line voltage is 230V. If we select a resistor of 6 M Ω, \( I_{\text{peak}} = \frac{230\, \text{V}}{6 \, \text{M} \, \Omega} = 38 \, \mu\text{A} \), which is well within the current capacity of a PICmicro MCU I/O pin. Input protection diodes (designed into the PICmicro MCU I/O pins) clamp any voltage higher than VDD or lower than VSS. Therefore, when the AC voltage is in the negative half of its cycle, the RB0 pin will be clamped to VSS - 0.6V. This will be interpreted as a logic zero. When the AC voltage rises above the input threshold, the logical value will become a ‘1’. In this
application, RB0 is configured for external interrupts, and the input buffer is a Schmitt trigger. This makes the input threshold $0.8 \text{ VDD} = 4\text{V}$ on a rising edge and $0.2 \text{ VDD} = 1\text{V}$ on a falling edge.

Upon each interrupt, the Interrupt Edge Select bit within the OPTION_REG register is toggled, so that an interrupt occurs on every zero-crossing.

![Figure E](image)

**120 KHZ CARRIER GENERATOR**

X-10 uses 120 kHz modulation to transmit information over 50 Hz power lines. It is possible to generate the 120 kHz carrier with an external oscillator circuit. A single I/O pin would be used to enable or disable the oscillator circuit output. However, an external oscillator circuit can be avoided by using one of the PICmicro MCU’s CCP modules. The CCP1 module is used in PWM mode to produce a 120 kHz square-wave with a duty cycle of 50%. After initialization, CCP1 is continuously enabled, and the TRISC bit for the pin is used to gate the PWM output. When the TRISC bit is set, the pin is an input and the 120 kHz signal is not presented to the pin. When the TRISC bit is clear, the pin becomes an output and the 120 kHz signal is coupled to the AC power line through a transistor amplifier and capacitor, as depicted in Figure.
Since the impedance of a capacitor is $Z_c = 1/(2\pi fC)$, a 0.1 µF capacitor presents a low impedance to the 120 kHz carrier frequency, but a high impedance to the 50 Hz power line frequency. This high-pass filter allows the 120 kHz signal to be safely coupled to the 50 Hz power line, and it doubles as the first stage of the 120 kHz carrier detector. To be compatible with other X-10 receivers, the maximum delay from the zero crossing to the beginning of the X-10 envelope should be about 300 µs. Since the zero-crossing detector has a maximum delay of approximately 64 µs, the firmware must take less than 236 µs after detection of the zero crossing to begin transmission of the 120 kHz envelope.

SQUARE WAVE GENERATION USING PIC.

PIC operated in PWM mode can be used for 120khz square wave generation. In Pulse Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output. A PWM output has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).
PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

\[
PWM \text{ period} = [(PR2) + 1] \times 4 \times \text{TOSC} \times (\text{TMR2 prescale value})
\]

PR2=41H, TOSC=20MHz, TMR2=1.

PWM frequency is defined as \(1 / [\text{PWM period}]\).

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty Cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H
PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available: the CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>.

The following equation is used to calculate the PWM duty cycle in time:

\[
\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON}<5:4>) \times T_{\text{osc}} \times (\text{TMR2 prescale value})
\]

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register. The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitches PWM operation. When the CCPR1H and 2-bit latch match TMR2 concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

LCD INTERFACING

![Figure H](image)
2) RECEIVER

SIGNAL ISOLATION MODULE & ENVELOPE DETECTOR

To receive X-10 signals, it is necessary to detect the presence of the 120 kHz signal on the AC power line. This is accomplished with a decoupling capacitor, a high-pass filter, a tuned amplifier, and an envelope detector. The components of the carrier detector are illustrated in Figure. Because the impedance of a capacitor is: \( Z_c = \frac{1}{2\pi f C} \), A 0.1 \( \mu \)F capacitor presents a low impedance (13 \( \Omega \)) to the 120 kHz carrier frequency, but a high impedance (26.5 k \( \Omega \)) to the 50 Hz power line frequency. This high-pass filter allows the 120 kHz signal to be safely coupled to the 50 Hz power line, and it doubles as the coupling stage of the 120 kHz carrier generator described in the next section. Since the 120 kHz carrier frequency is much higher than the 50 Hz power line frequency, it is straightforward to design an RC filter that will pass the 120 kHz signal and completely attenuate the 50 Hz. For a simple high-pass filter, the -3-db breakpoint is:

\[
 f_{3\,db} = \frac{1}{2\pi R C} \]

For \( C = 150 \) pF and \( R = 33 \) k\( \Omega \),

\[
 f_{3\,db} = \frac{1}{2\pi * 150 \, \text{pF} * 33 \, \text{k}\Omega} = 32 \, \text{kHz}.
\]
This $f_3$ db point assures that the 50 Hz signal is completely attenuated, while the 120 kHz signal is passed through to the amplifier stages. Next, the 120 kHz signal is amplified using a series of inverters configured as high gain amplifiers. The first two stages are tuned amplifiers with peak response at 120 kHz. The next two stages provide additional amplification. The amplified 120 kHz signal is passed through an envelope detector, formed with a diode, capacitor, and resistor. The envelope detector output is buffered through an inverter and presented to an input pin (RC3) of the PIC16F877A. Upon each zero-crossing interrupt, RC3 is simply checked within the 1 ms transmission envelope to see whether or not the carrier is present. The presence or absence of the carrier represents the stream of ‘1’s and ‘0’s that form the X-10 messages.

![Figure J](image)

![Figure K](image)
SOFTWARE SECTION
ALGORITHM FOR TRANSMITTER

Step1: Do the basic initialization of pwm, external interrupt, LCD interfacing, arrays and basic variables.
Step2: Start code, House code, device code and function code are stored in arrays according to X-10 protocol.
Step3: Configure RB1, RB2, RB3 and RB4 as input ports.
Step4: Display ‘Power line communication using X-10 protocol’
Step5: Give 2 seconds delay.
Step6: Display ‘House:’ and ‘Device:’ on LCD
Step7: PWM generates 120 KHz with 50% duty cycle.
Step8: Enable external interrupt.
Step9: Display house number from 1-5.
Step10: Display device number from 1-5.
Step11: Display device status as ON, Off, ON all, OFF all.
Step12: Interrupt routine.
Step13: Start code, House code, device code, and device status are generated according to the input keying.
Step14: Send address once.
Step15: Send address twice.
Step16: Send function key once.
Step17: Send function key twice.
Step18: 1ms delay.
Step19: Call LCD command.
Step20: Call LCD data.
Step21: Delay.
Step22: Send code.
ALGORITHM FOR RECEIVER

STEP 1 : Do the basic initialization- configure RC2 as input, RB0- RB4 as output, lcd and basic variables.

STEP 2 : Check RC2 at each zero crossing for 1 ms.

STEP 3 : Compare the received bits initially with start code.

STEP 4 : If start code received is correct, store the remaining bits received after start code.

STEP 5 : Check whether the stored bit sequence match with the house code.

STEP 6 : If house code is correct compare the device code to select the device.

STEP 7 : Now compare the function code to select the function (if function is ON then output high to corresponding RB pins).

STEP 9 : Check for sensor data.

STEP 10 : Transmit the sensor data.

STEP 11 : End
Initialize the ports by setting it as inputs and outputs.

Initialize LCD. Display house number, device number and status

Set PWM for 120 KHz with 50% duty cycle

Enable external for checking zero crossings

House key

Y

Increment house number on each key press

House No.>=5

Y

Reset house No. =0

Display house number on LCD

N

F

A

N
Device key
Y
Increment device number on each key press
N
Device No. >= 5
Y
Reset device No. = 0
N
Display device number on LCD

Function key pressed?
N
Y
Increment status number on each key press
N
Status No. >= 4?
Y
Reset status No. = 0
N
Display the specified function according to status number

A

B

XXVIII
Send key pressed?

Set count = 0

Ext Interrupt occurred?

Wait for six external interrupts (six zero crossings)

Send address bits one by one according to X-10 protocol

Send Bit=1?

Out PWM for 4mS

Count = Count++

Count = 2?

Count = Count++

Ext Interrupt occurred?

Out PWM for 4mS

Send command bits one by one according to X-10 protocol

Send Bit=1?

Out PWM for 4mS

Count = 0

Count = Count++

All bits send?

Ende

End
Start

Initialize the ports by setting it as inputs and outputs.

Enable external for checking zero crossings

Ext Interrupt occurred?

Delay 400μ s

Read input port for checking the data received

Bit received=1

Delay 1mS

Input port is low?

A
Store next three bits on each zero crossings

Compare the stored start bit with the received bit

Start bit = received bit?

N

Y

Store the rest of bits received in an array for checking

Compare the house code received with the house address

House code received = house

N

Y

Compare the 2nd house code received with the house address

2nd House code received = house

N

Y
Compare 1st device code received with 2nd device code

If 1st device code = 2nd device code?

If command

Out HIGH to port with the device address

Out LOW to port with the device address

If command

Compare 1st function code received with 2nd function code

If command is = ON

Out HIGH to all ports where the devices connected

Out LOW to all ports where the devices connected

End
FUTURE SCOPE

A model for PLC is shown below.

Figure E  Power line communications from high voltage network to individual premises.

X10 Technology offers a solution that was initially developed to integrate with low cost lighting and appliance control devices. It is now trying to innovate into higher speeds with regard to establishing the communication between home PCs and controlled home appliances.

In figure 1, a substation may implement a backhaul connection (i.e. T1, E1, etc.), which connects to a PLC unit that connects to the medium voltage grid. Repeater units or similar head end units are placed on the medium voltage grid at intervals to connect substations together and back to the backhaul connection. Signaling could use Digital Spread Spectrum
(DSS) where signals pass through line equipment as this less expensive and easier to deploy or Orthogonal Frequency Multiplexing (OFDM) where signals do not pass through line equipment and must bypass it via a bridge or coupler device (copper or fibre).

Figure F: Power line communications from low voltage network into an individual premises.

Figure 2 demonstrates how end users plug a PLC modem into a power outlet in the home or business premises to connect to a computer. The end user now has an "always-on" Internet connection from a power socket, which is connected to PLC unit at a substation onto the medium voltage network and then to the backhaul connection for Internet connectivity.
CONCLUSION

This seminar as it stands now is a fully functional and useful product. It has a simple protocol, commands set and a low implementation cost. We have one module that when plugged in to a standard AC outlet that controls the devices and broadcast the data over the power line. We have another module that reads the data from the power line and displays the current temperature and luminance besides broadcasting the control data. Compared to other methods, this is a very simple controlling method. By implementing this technology we are overcoming the disadvantages of present data cabling system. We can control the device easily. Cost of implementation is low.
APPENDIX 1
PROGRAM

TRANSMITTER

#include<pic.h>
/*=======================================================
==
functions used to send code
=======================================================*/

void startcode(int x);
void housecode(int x);
void devicecode(int x);
void functioncode(int x);
void sendaddressonce();
void sendaddressstwise();
void sendfunctiononce();
void sendfunctiontwice();
void delay1ms();
void delay1();
void delay();
/*=======================================================
==
functions used to display on LCD
=======================================================*/

void lcd();
void cmd();
void data();
void housenumber();
void devicenumber();
void devicestatus();
void sendcode();

/*=======================================================
==
functions used to generate PWM
=======================================================
=*/

void interrupt pwm();

bank2 char
i, send, set=1, housekey=0, devicekey=0, onkey=0, name[48] = "HOUSE:0DEVICE:0  DEVICE CONTROL USING X10 SELECT";
char scode[5] = "1110";
bank1 char hcode[21] = "01101110001010100001";
bank1 char dcode[26] = "0110011100001001010000010";
bank2 char fcode[26] = "0000100101001110001100001";

int
time=0, count=0, key1=0, key2=0, key3=0, key4=0, on=0, stest=0, htest=0, dtest=0, ftest=0, startbit, devicebit, housebit;
int statuskey=0, functionbit, sendtwice=0, complement=0;

/*=======================================================
=*/

void main()
{
    TRISC2=1;       // pwm output pin disable
    TRISE=0x00;
    TRISB=0x0f;
    TRISD=0x00;
    TRISC=0x0c;
}

XXXVIII
RB5=1;
RB6=1;
RB7=0;
lcd();

/*=====================================================*
|==
|Display "Device control using X10" on LCD
|=====================================================*

send=0x80;
cmd();
for(i=16;i<31;i++)
{
    send=name[i];
data();
}

send=0xC2;
cmd();
for(i=31;i<41;i++)
{
    send=name[i];
data();
}
for(i=0;i<200;i++)
{
delay();
}
send=0x01;
cmd();
Display house and device number on LCD

=*/
    send=0x80;
    cmd();
    for(i=42;i<48;i++)
    {
        send=name[i];
        data();
    }
    send=0xc0;
    cmd();
    for(i=0;i<7;i++)
    {
        send=name[i];
        data();
    }
    send=0xC8;
    cmd();
    for(i=7;i<15;i++)
    {
        send=name[i];
        data();
    }
/*=====================================================*/

==
Set pwm for 120KHz with 50% duty cycle

XL
PR2=41;
TMR2ON=1;
T2CKPS1=0;
T2CKPS0=0;
TMR2=0X00;
CCPR1L=0X13;
//CCP1X=0;
//CCP1Y=0;
CCP1CON=0X0C;
/*=====================================================*/
/*==Enable external interrupt*/
/*=====================================================*/
GIE=1;
RBPU=0;
INTEDG=1;
INTF=0;
INTE=1;
while(1)
{
   housenumber(); /*change house number on each key press*/
   devicenumber(); /*change device number on each key press*/
   devicestatus();
   sendcode();
   /*display on LCD*/
}

XLI
if(onkey==1)  /*check house and device switch*/
{
  onkey=0;
  key1=1;
  on=0;
  count=0;
  startbit=0;
  housebit=4*housekey;
  devicebit=5*devicekey;
  functionbit=5*statuskey;
  complement=0;
}

if((key1==1)&&(on==1))  /*send corresponding code*/
{
  sendaddressonce();
  if(count==21)
  {
    startbit=0;
    housebit=4*housekey;
    devicebit=5*devicekey;
    complement=0;
  }
  sendaddress twice();
  if(count==43)
  {
    startbit=0;
    housebit=4*housekey;
    devicebit=5*devicekey;
    complement=0;
  }
}
functionbit=5*statuskey;
complement=0;
}
sendfunctiononce();
if(count==71)
{
    startbit=0;
    housebit=4*housekey;
    functionbit=5*statuskey;
    complement=0;
}
sendfunctiontwice();
if(count==94)
{
    key1=0;
}
count=count++;
on=0;
}}}

RECEIVER

#include<pic.h>
#include<string.h>
#include"delay.c"
/*=====================================================*/
/*const unsigned bank1 char
*/

XLIII
const unsigned bank1 char
const unsigned bank1 char
bank2 char bitsrec[91 ];
bank3 char
startbitsrec[5],firsthousecodereceived[9],secondhousecodereceived[9],offallok1=0,onallok2=0,offallok2=0;
bank3 char
firstdevicecodereceived[11],seconddevicecodereceived[11],firstfunctioncode received[11],secondfunctioncodereceived[11];
bank3 char
i,j,key,stest=0,hertest=0,d1test=0,d2test=0,d3test=0,d4test=0,d5test=0,ftest=0,
on=0,time=0,scount=0,count=0,complete=0,firstbit=0;
char
startok=0,houseok1=0,houseok2=0,device1ok1=0,device2ok1=0,device3ok1 =0,device4ok1=0,device5ok1=0,onok1=0,offok1=0;
char
device1ok2=0,device2ok2=0,device3ok2=0,device4ok2=0,device5ok2=0,on
ok2=0,offok2=0;
char ontest=0,offtest=0,onalltest=0,offalltest=0,onallok1=0;
/*=====================================================*/
void interrupt reception();
void uart();
void ext_interrupt();
void port_settings();
void firstbit_chk();
void startcode_chk();
void bits_capture();
void housecode_chk();
void devicecode_chk();
void oncode_chk();
void offcode_chk();
void onallcode_chk();
void offallcode_chk();

/*=====================================================*/

void main()
{
    port_settings();
    ext_interrupt();
    uart();
    firstbit_chk();
    while(1)
    {
        if(on==1)
        {
            on=0;
            firstbit_chk();
            if(firstbit==1)
            {
                startcode_chk();
                bits_capture();
            }
        }
        if(complete==1)
        {
            complete=0;
        }
    }
}
housecode_chk();
if(htest==1)
{
    htest=0;
devicecode_chk();
    if((d1test==1)||(d2test==1)||(d3test==1)||(d4test==1)||(d5test==1))
    {
        oncode_chk();
        offcode_chk();
onalldcode_chk();
        offalldcode_chk();
    }
onok1=0;
onok2=0;
onallok1=0;
onallok2=0;
offok1=0;
offok2=0;
offallok1=0;
offallok2=0;
}
stest=0;
}
Devices Included in this Data Sheet:
- PIC16F873
- PIC16F876
- PIC16F874
- PIC16F877

Microcontroller Core Features:
- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
- DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory
- Up to 384 x 8 bytes of Data Memory (RAM)
- Up to 256 x 8 bytes of EEPROM data memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Supply/Source Current: 28 mA
- Commercial and Industrial temperature ranges
- Low-power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Peripheral Features:
- Timer0: 8-bit timer-counter with 8-bit prescaler
- Timer1: 16-bit timer-counter with prescaler, can be incremented during sleep via external crystal/lock
- Timer2: 8-bit timer-counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master Mode) and PCF™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCH) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)
### TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>DIP Pin#</th>
<th>PLCC Pin#</th>
<th>GPP Pin#</th>
<th>TOP Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSC1/CLKIN</td>
<td>13</td>
<td>14</td>
<td>30</td>
<td>1</td>
<td>ST/CMOS/4</td>
<td>Oscillator crystal input/external clock source input.</td>
</tr>
<tr>
<td>OSC2/CLKOUT</td>
<td>14</td>
<td>15</td>
<td>31</td>
<td>C</td>
<td>—</td>
<td>Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of CLKIN and doubles the instruction cycle rate.</td>
</tr>
<tr>
<td>PIC18F/TPM1/</td>
<td>1</td>
<td>2</td>
<td>18</td>
<td>IF</td>
<td>ST</td>
<td>Master clear (reset) input or programming voltage input or high voltage test mode control. This pin is an active low reset to the device.</td>
</tr>
<tr>
<td>RDA/ANO</td>
<td>2</td>
<td>3</td>
<td>19</td>
<td>100</td>
<td>TIL</td>
<td>RDA can also be analog input.</td>
</tr>
<tr>
<td>RDA/AN0</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>TIL</td>
<td>RDA can also be analog input.</td>
<td></td>
</tr>
<tr>
<td>RDA/AN1</td>
<td>4</td>
<td>5</td>
<td>11</td>
<td>TIL</td>
<td>RDA2 can also be analog input or negative analog reference voltage.</td>
<td></td>
</tr>
<tr>
<td>RDA/AN2/REF-</td>
<td>5</td>
<td>6</td>
<td>12</td>
<td>TIL</td>
<td>RDA3 can also be analog input or positive analog reference voltage.</td>
<td></td>
</tr>
<tr>
<td>RAMTC0</td>
<td>6</td>
<td>7</td>
<td>13</td>
<td>100</td>
<td>ST</td>
<td>RAM can also be the clock input to the Timer0 timer counter. Outputs open drain type.</td>
</tr>
<tr>
<td>RAMTC1</td>
<td>7</td>
<td>8</td>
<td>14</td>
<td>100</td>
<td>TIL</td>
<td>RAM can also be analog input or the slave select for the synchronous serial port.</td>
</tr>
<tr>
<td>RX0/RINT</td>
<td>33</td>
<td>36</td>
<td>8</td>
<td>100</td>
<td>TIL</td>
<td>RX0 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX1</td>
<td>34</td>
<td>37</td>
<td>9</td>
<td>100</td>
<td>TIL</td>
<td>RX0 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX2</td>
<td>35</td>
<td>38</td>
<td>10</td>
<td>100</td>
<td>TIL</td>
<td>RX0 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX3</td>
<td>36</td>
<td>39</td>
<td>11</td>
<td>100</td>
<td>TIL</td>
<td>RX3 can also be the low voltage programming input.</td>
</tr>
<tr>
<td>RX3</td>
<td>37</td>
<td>41</td>
<td>14</td>
<td>100</td>
<td>TIL</td>
<td>RX3 can also be the low voltage programming input.</td>
</tr>
<tr>
<td>RX5</td>
<td>38</td>
<td>42</td>
<td>15</td>
<td>100</td>
<td>TIL</td>
<td>RX5 can also be the low voltage programming input.</td>
</tr>
<tr>
<td>RX5</td>
<td>39</td>
<td>43</td>
<td>16</td>
<td>100</td>
<td>TIL</td>
<td>RX5 can also be the low voltage programming input.</td>
</tr>
<tr>
<td>RX0/RINT</td>
<td>40</td>
<td>44</td>
<td>17</td>
<td>100</td>
<td>TIL</td>
<td>RX0 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX1</td>
<td>41</td>
<td>45</td>
<td>18</td>
<td>100</td>
<td>TIL</td>
<td>RX1 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX2</td>
<td>42</td>
<td>46</td>
<td>19</td>
<td>100</td>
<td>TIL</td>
<td>RX2 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX3</td>
<td>43</td>
<td>47</td>
<td>20</td>
<td>100</td>
<td>TIL</td>
<td>RX3 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX3</td>
<td>44</td>
<td>48</td>
<td>21</td>
<td>100</td>
<td>TIL</td>
<td>RX3 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX5</td>
<td>45</td>
<td>49</td>
<td>22</td>
<td>100</td>
<td>TIL</td>
<td>RX5 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX5</td>
<td>46</td>
<td>50</td>
<td>23</td>
<td>100</td>
<td>TIL</td>
<td>RX5 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX6/RINT</td>
<td>47</td>
<td>51</td>
<td>24</td>
<td>100</td>
<td>TIL</td>
<td>RX6 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX7/RINT</td>
<td>48</td>
<td>52</td>
<td>25</td>
<td>100</td>
<td>TIL</td>
<td>RX7 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX0/RINT</td>
<td>49</td>
<td>53</td>
<td>26</td>
<td>100</td>
<td>TIL</td>
<td>RX0 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX1</td>
<td>50</td>
<td>54</td>
<td>27</td>
<td>100</td>
<td>TIL</td>
<td>RX1 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX2</td>
<td>51</td>
<td>55</td>
<td>28</td>
<td>100</td>
<td>TIL</td>
<td>RX2 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX3</td>
<td>52</td>
<td>56</td>
<td>29</td>
<td>100</td>
<td>TIL</td>
<td>RX3 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX4</td>
<td>53</td>
<td>57</td>
<td>30</td>
<td>100</td>
<td>TIL</td>
<td>RX4 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX5</td>
<td>54</td>
<td>58</td>
<td>31</td>
<td>100</td>
<td>TIL</td>
<td>RX5 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX6/RINT</td>
<td>55</td>
<td>60</td>
<td>32</td>
<td>100</td>
<td>TIL</td>
<td>RX6 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX7/RINT</td>
<td>56</td>
<td>61</td>
<td>33</td>
<td>100</td>
<td>TIL</td>
<td>RX7 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX0/RINT</td>
<td>57</td>
<td>62</td>
<td>34</td>
<td>100</td>
<td>TIL</td>
<td>RX0 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX1</td>
<td>58</td>
<td>63</td>
<td>35</td>
<td>100</td>
<td>TIL</td>
<td>RX1 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX2</td>
<td>59</td>
<td>64</td>
<td>36</td>
<td>100</td>
<td>TIL</td>
<td>RX2 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX3</td>
<td>60</td>
<td>65</td>
<td>37</td>
<td>100</td>
<td>TIL</td>
<td>RX3 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX4</td>
<td>61</td>
<td>66</td>
<td>38</td>
<td>100</td>
<td>TIL</td>
<td>RX4 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX5</td>
<td>62</td>
<td>67</td>
<td>39</td>
<td>100</td>
<td>TIL</td>
<td>RX5 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX6/RINT</td>
<td>63</td>
<td>68</td>
<td>40</td>
<td>100</td>
<td>TIL</td>
<td>RX6 can also be the external interrupt pin.</td>
</tr>
<tr>
<td>RX7/RINT</td>
<td>64</td>
<td>69</td>
<td>41</td>
<td>100</td>
<td>TIL</td>
<td>RX7 can also be the external interrupt pin.</td>
</tr>
</tbody>
</table>

Legend: 1-2 = input 0-3 = output 4-9 = input/output 10 = power — = Not used TTL = TTL input 11 = Schmitt Trigger input

Notes:
1. The buffers are Schmitt Trigger input when configured as an external interrupt.
2. The buffers are Schmitt Trigger input when configured as an external interrupt.
3. The buffers are Schmitt Trigger input when configured as an external interrupt.
4. The buffers are Schmitt Trigger input when configured as an external interrupt.
### TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>DIP Pin</th>
<th>PLCC Pin</th>
<th>DIP Pin</th>
<th>I/O Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RO1/SP0</td>
<td>10</td>
<td>21</td>
<td>31</td>
<td>PO</td>
<td>ST/TL(9)</td>
<td>FORTE is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.</td>
</tr>
<tr>
<td>RO2/SP1</td>
<td>20</td>
<td>22</td>
<td>39</td>
<td>PO</td>
<td>ST/TL(9)</td>
<td>REG can also be used for the parallel slave port or analog input.</td>
</tr>
<tr>
<td>RO3/SP2</td>
<td>21</td>
<td>23</td>
<td>40</td>
<td>PO</td>
<td>ST/TL(9)</td>
<td>REG can be used for the parallel slave port or analog input.</td>
</tr>
<tr>
<td>RO4/SP3</td>
<td>22</td>
<td>24</td>
<td>41</td>
<td>PO</td>
<td>ST/TL(9)</td>
<td>REG can be used for the parallel slave port, or analog input.</td>
</tr>
<tr>
<td>RO5/SP4</td>
<td>27</td>
<td>30</td>
<td>2</td>
<td>PO</td>
<td>ST/TL(9)</td>
<td>REG can be used for the parallel slave port, or analog input.</td>
</tr>
<tr>
<td>RO6/SP5</td>
<td>28</td>
<td>31</td>
<td>2</td>
<td>PO</td>
<td>ST/TL(9)</td>
<td>REG can be used for the parallel slave port, or analog input.</td>
</tr>
<tr>
<td>RO7/SP6</td>
<td>29</td>
<td>32</td>
<td>4</td>
<td>PO</td>
<td>ST/TL(9)</td>
<td>REG can be used for the parallel slave port, or analog input.</td>
</tr>
<tr>
<td>RO8/SP7</td>
<td>30</td>
<td>33</td>
<td>1</td>
<td>PO</td>
<td>ST/TL(9)</td>
<td>REG can be used for the parallel slave port, or analog input.</td>
</tr>
</tbody>
</table>

**Legend:**
- I = input
- O = output
- D = directional
- N = Not used
- ST = Schottky Trigger input

**Note:**
1. This buffer is a Schottky Trigger input when configured as an external interrupt.
2. This buffer is a Schottky Trigger input when used in serial programming mode.
3. This buffer is a Schottky Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4. This buffer is a Schottky Trigger input when configured in RC oscillator mode and a CMOS input otherwise.
### TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>DIP Pin</th>
<th>PICC Pin</th>
<th>DIP Pin</th>
<th>I/O Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD0/SP0</td>
<td>19</td>
<td>31</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td>FORTE is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.</td>
</tr>
<tr>
<td>RD1/SP1</td>
<td>20</td>
<td>32</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD2/SP2</td>
<td>21</td>
<td>33</td>
<td>40</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
</tr>
<tr>
<td>RD3/SP3</td>
<td>22</td>
<td>34</td>
<td>41</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
</tr>
<tr>
<td>RD4/SP4</td>
<td>27</td>
<td>35</td>
<td>42</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
</tr>
<tr>
<td>RD5/SP5</td>
<td>28</td>
<td>36</td>
<td>43</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
</tr>
<tr>
<td>RD6/SP6</td>
<td>29</td>
<td>37</td>
<td>44</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
</tr>
<tr>
<td>RD7/SP7</td>
<td>30</td>
<td>38</td>
<td>45</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>DIP Pin</th>
<th>PICC Pin</th>
<th>DIP Pin</th>
<th>I/O Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD0/SP0</td>
<td>31</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD1/SP1</td>
<td>32</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD2/SP2</td>
<td>40</td>
<td>41</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD3/SP3</td>
<td>41</td>
<td>42</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD4/SP4</td>
<td>42</td>
<td>43</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD5/SP5</td>
<td>43</td>
<td>44</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD6/SP6</td>
<td>44</td>
<td>45</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD7/SP7</td>
<td>45</td>
<td>46</td>
<td>FO</td>
<td>ST/TTL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Notes:
1. This buffer is a Schmitt Trigger input when configured as an external interrupt.
2. This buffer is a Schmitt Trigger input when used in serial programming mode.
3. This buffer is a Schmitt Trigger input when configured as a general-purpose I/O and a TTL input when used in Parallel Slave Port mode (if interfacing to a microprocessor bus).
4. This buffer is a Schmitt Trigger input when configured in IOC oscillator mode and a CMOS input otherwise.

Legend: 1 = input, 0 = output, - = Not used, TTL = TTL input, ST = Schmitt Trigger input.
### 2.2.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the resets and the bank select bits for data memory.

The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the units to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the TO and BD bits are not volatile; therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, CDRF STATUS will clear the upper-three bits and set the Z bit. This leaves the STATUS register as 000b or value (where U = unchanged).

It is recommended, therefore, that only BCP, BDF, SWAPF and SWAPF instructions are used to alter the STATUS register because these instructions do not affect the Z, DC or DC bits from the STATUS register. For other instructions not affecting any status bits, see the "Instruction Set Summary".

Note 1: The C and DC bits operate as a borrow and high borrow bit, respectively, in subtraction. See the SWAP and SWAPF instructions for examples.

### REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 183h, 183h)

<table>
<thead>
<tr>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R-J</th>
<th>R-M</th>
<th>RW-x</th>
<th>RW-x</th>
<th>RW-x</th>
<th>RW-x</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP</td>
<td>RP</td>
<td>RP</td>
<td>TO</td>
<td>PS</td>
<td>Z</td>
<td>DC</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>bit7</td>
<td>bit6</td>
<td>bit5</td>
<td>bit4</td>
<td>bit3</td>
<td>bit2</td>
<td>bit1</td>
<td>bit0</td>
<td></td>
</tr>
</tbody>
</table>

- **R = Readable bit**
- **W = Writable bit**
- **U = Unimplemented bit, read as '0'**
- **n = Value at POR reset**

**bit 7:** RP
- R0: Register Bank Select bit (used for indirect addressing)
- 1 = Bank 2, 3 (00h - 1Fh)
- 0 = Bank 0, 1 (00h - FFh)

**bit 6-5:** RP1:RP0: Register Bank Select bits (used for direct addressing)
- 11 = Bank 8 (80h - 1FH)
- 10 = Bank 2 (40h - 5Fh)
- 01 = Bank 1 (20h - 3Fh)
- 00 = Bank 0 (00h - FFh)

Each bank is 128 bytes.

**bit 4:** TO: Time-out bit
- := After power-up, CLEAR instruction, or SLEEP instruction
- := A WDT time-out occurred

**bit 3:** PS: Power-down bit
- := After power-up or by the CLEAR instruction
- := By execution of the SLEEP instruction

**bit 2:** Z: Zero bit
- := The result of an arithmetic or logic operation is zero
- := The result of an arithmetic or logic operation is not zero

**bit 1:** DC: Digit carry/Borrow bit (ADD, ADDL, ADDH, SUB, SUBL instructions)
- (for borrow the polarity is reversed)
- := A carry-out from the 4th low order bit of the result occurred
- := No carry-out from the 4th low order bit of the result

**bit 0:** C: Carry/Sorrow bit (ADD, ADDL, ADDH, SUB, SUBL instructions)
- := A carry-out from the most significant bit of the result occurred
- := No carry-out from the most significant bit of the result occurred

Note: For borrow the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For return (RET, RIP) instructions, the bit is loaded with either the high or low order bit of the source register.
2.2.2 OPTION_REG REGISTER

The OPTION_REG Register is a readable and writable register which contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the External INT, Interrupt TMR0 and the weak pull-ups on PORTB.

Note: To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS $11h, $18h)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pull-up Enable (PORTB)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Interrupt on rising edge of RB0/INT pin</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Interrupt on falling edge of RB0/INT pin</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TMR0 Clock Source Select bit</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Transistor on RA4/TOCKI pin</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Internal instruction cycle clock (CLKOUT)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Prescaler Assignment bit</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Prescaler is assigned to the WDT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>TMR0 Ratio</th>
<th>WDT Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1:4</td>
<td>1:10</td>
</tr>
<tr>
<td>001</td>
<td>1:2</td>
<td>1:2</td>
</tr>
<tr>
<td>010</td>
<td>1:1</td>
<td>1:1</td>
</tr>
<tr>
<td>011</td>
<td>1:16</td>
<td>1:16</td>
</tr>
<tr>
<td>100</td>
<td>1:32</td>
<td>1:32</td>
</tr>
<tr>
<td>101</td>
<td>1:128</td>
<td>1:64</td>
</tr>
<tr>
<td>110</td>
<td>1:256</td>
<td>1:128</td>
</tr>
</tbody>
</table>

Note: When using low voltage ICSP programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-ups on RB3 and ensure the proper operation of the device.
## PIC16F87X

### 22.2.3 INTCON REGISTER

The INTCON Register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB Port change and External RB0 INT pin interrupts.

**Note:** Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt.

### REGISTER 2-3: INTCON REGISTER (ADDRESS 08h, 88h, 108h, 188h)

<table>
<thead>
<tr>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIE</td>
<td>PEIE</td>
<td>TOIE</td>
<td>INTE</td>
<td>RBIE</td>
<td>TOIF</td>
<td>INTF</td>
<td>RBIF</td>
<td></td>
</tr>
</tbody>
</table>

- **R =** Readable bit
- **W =** Writable bit
- **U =** Unimplemented bit, read as '0'
- **n =** Value at POR reset

### Bit 7: GIE: Global Interrupt Enable bit
- 1 = Enables all unmasked interrupts
- 0 = Disables all interrupts

### Bit 6: PEIE: Peripheral Interrupt Enable bit
- 1 = Enables all unmasked peripheral interrupts
- 0 = Disables all peripheral interrupts

### Bit 5: TOIE: TMR0 Overflow Interrupt Enable bit
- 1 = Enables the TMR0 interrupt
- 0 = Disables the TMR0 interrupt

### Bit 4: INTE: RB0/INT External Interrupt Enable bit
- 1 = Enables the RB0/INT external interrupt
- 0 = Disables the RB0/INT external interrupt

### Bit 3: RBIE: RB Port Change Interrupt Enable bit
- 1 = Enables the RB port change interrupt
- 0 = Disables the RB port change interrupt

### Bit 2: TOIF: TMR0 Overflow Interrupt Flag bit
- 1 = TMR0 register has overflowed (must be cleared in software)
- 0 = TMR0 register did not overflow

### Bit 1: INTF: RB0/INT External Interrupt Flag bit
- 1 = The RB0/INT external interrupt occurred (must be cleared in software)
- 0 = The RB0/INT external interrupt did not occur

### Bit 0: RBIF: RB Port Change Interrupt Flag bit
- 1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
- 0 = None of the RB7:RB4 pins changed state
# APPENDIX-3

## CD4069UBC Inverter Circuits

### General Description

The CD4069UB consists of six inverter circuits and is manufactured using complementary MOS (CMOS) to achieve wide power supply operating range, low power consumption, high noise immunity, and symmetric controlled rise and fall times.

This device is intended for all general purpose inverter applications where the special characteristics of the MM74C001, MM74C007, and CD4049A Hex inverter/Buffers are not required. In those applications requiring larger noise immunity the MM74C14 or MM74C141 Hex Schmitt Trigger is suggested.

All inputs are protected from damage due to static discharge by diode clamps to $V_{CC}$ and $V_{SS}$.

### Features

- Wide supply voltage range: 3.0V to 18V
- High noise immunity: 0.45 $V_{CC}$ typ.
- Low power TTL compatibility: Fan out of 2 driving 74L or 1 driving 74LS
- Equivalent to MM74C04

### Ordering Code:

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Package Number</th>
<th>Package Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD4069UBCM</td>
<td>M14A</td>
<td>14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150&quot; Narrow</td>
</tr>
<tr>
<td>CD4069UBSJ</td>
<td>M14U</td>
<td>14-Lead Small Outline Package (SOP), EIAJ TYPE II, 0.300&quot; Wide</td>
</tr>
<tr>
<td>CD4069UBCN</td>
<td>N14A</td>
<td>14-Lead Plastic Dual-Inline Package (PDIP), JEDEC MS-001, 0.300&quot; Wide</td>
</tr>
</tbody>
</table>

Device also available in Tape and Reel. Specify by appending suffix "R" to the ordering code.

### Connection Diagram

[Connection Diagram Image]

### Schematic Diagram

[Schematic Diagram Image]
APPENDIX-4

HD44780U (LCD-II)

(Dot Matrix Liquid Crystal Display Controller/Driver)

HITACHI

Description

The HD44780U dot-matrix liquid crystal display controller and driver LSI displays alphanumericics, Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver.

A single HD44780U can display up to one 8-character line or two 8-character lines.

The HD44780U has pin function compatibility with the HD44780S which allows the user to easily replace an LCD-II with an HD44780U. The HD44780U character generator ROM is extended to generate 208 5 × 8 dot character fonts and 32 5 × 10 dot character fonts for a total of 240 different character fonts.

The low power supply (2.7V to 5.5V) of the HD44780U is suitable for any portable battery-driven product requiring low power dissipation.

Features

- 5 × 8 and 5 × 10 dot matrix possible
- Low power operation support
  — 2.7 to 5.5V
- Wide range of liquid crystal display driver power
  — 3.0 to 11V
- Liquid crystal drive waveform
  — A (One line frequency AC waveform)
- Correspond to high speed MPU bus interface
  — 2 MHz (when VCC = 5V)
- 4-bit or 8-bit MPU interface enabled
- 80 × 8-bit display RAM (80 characters max.)
- 9,020-bit character generator ROM for a total of 240 character fonts
  — 208 character fonts (5 × 8 dot)
  — 32 character fonts (5 × 10 dot)
REFERENCE

1) www.How X10 Works - SmartHomeUSA_com.
2) www.Home Toys Article.htm
3) Data Transmission through Power Lines
   C. A. Duque (M-IEEE), P. G. Barbosa (M-IEEE) and D. P. Baptista,
   Student Member, IEEE
4) An Assessment of Power Line Communications
   for the delivery of broadband network access, James Sankar, July 2004
5) www.Welcome to IEEE Xplore 2_0 Design of wireless remote module in
   X-10 intelligent home.htm