X-10 Basics

Learn about how these home-control modules work and build an X-10 Computer Interface that will let you use them with your PC

Since the late 1970s, X-10 has provided a series of controllers and modules for use in home-automation systems. Over the years, the X-10 protocol has become the standard for ac power-line carrier transmission and has resulted in a large base of products that support the X-10 protocol, all very reasonably priced.

Even though application of the various X-10 modules have received fairly good exposure, most of the theory behind X-10 operation has gone undocumented in electronics magazines. Thus, though many readers may be familiar with the products themselves, most aren't familiar with actual X-10 module operation.

While this may be fine if all you want to do is implement a system, I personally would much rather build my own devices than simply go someone else's. Therefore, I dug around to find out exactly how X-10 modules work so that I could build a computer interface and write my own control software that will give me low-level control of my X-10 system. The results of my digging are documented here. Not only do I cover the theory behind the X-10 stable of devices, I show you how to build an interface that lets you use these modules with your PC.

X-10 Protocol

Essentially, X-10 modules place data for transmission on the ac power line, using the 60-Hz waveform as a carrier. The data placed on the 60-Hz carrier is actually packets of bits that are synchronized with the zero crossings of the ac waveform.

Within each packet of data, there are three separate codes: (1) Start Code; (2) House Code; and (3) Key Code.

The start code is a hard-wired code that signals all X-10 modules on the line that an X-10 code is coming. This four-bit code has a pattern of 1110 and is transmitted differently than the House and Key Codes in that one bit is sent for each zero crossing (Fig. 1) versus the complimentary form used in the House and Key Codes transmission.

Since each zero crossing represents one half of the 60-Hz waveform, two bits can be sent per cycle. The first bit is sent on one half of the cycle, the second on the alternate half of the cycle. Therefore, the start code requires two complete cycles (four bits divided by two bits/cycle) for transmission.

Once the start code has been received, the X-10 modules begin "listening" for a valid data packet from the controller. The House Code is the next portion of the packet to be transmitted.

The House Code is a four-bit pattern that's used to prevent controllers in other homes from interfering with local X-10 modules. It's determined by the user and should be set so that it's unique to other X-10 systems operating on the same power lines.

Unlike the Start Code, which requires only two cycles to transmit, the House Code requires four complete cycles for transmitting because the House Code is transmitted in complement form.

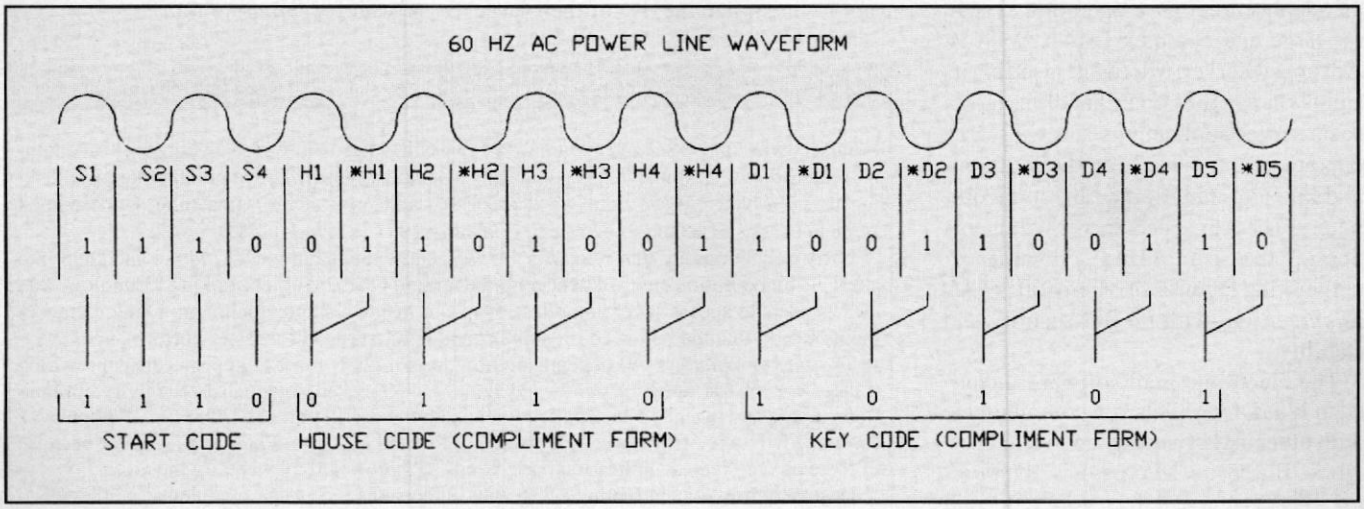
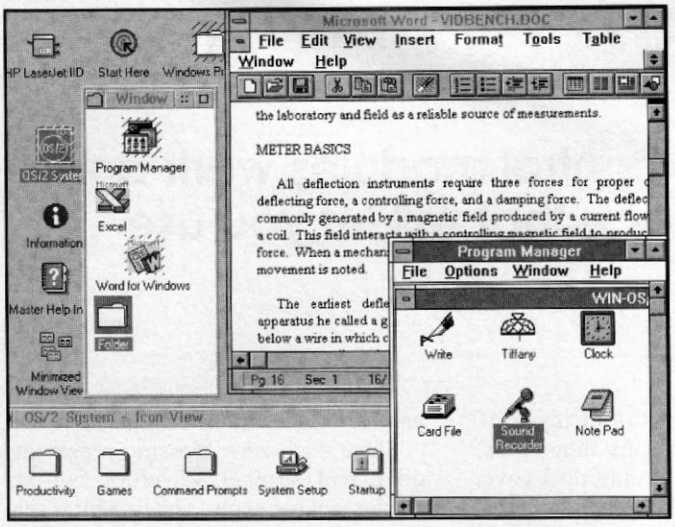
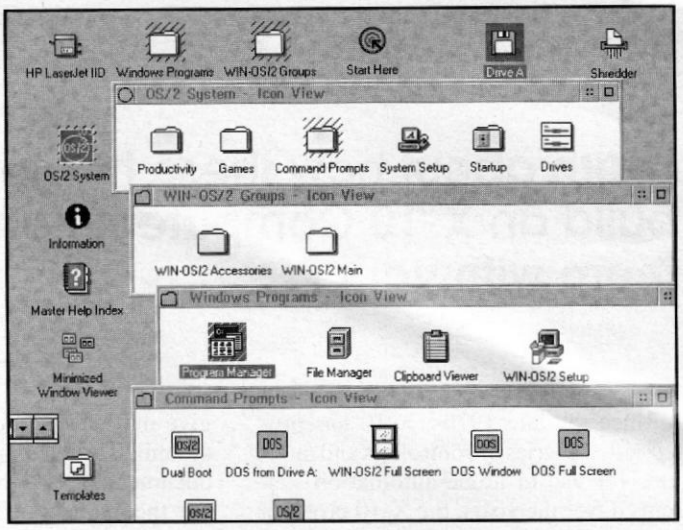


Fig. 1. Transmission framing diagram.



OS/2 allows you to boot your system from one or more operating systems.



Under OS/2, you can leave often-used icons on the desktop or place them in folders for easy sorting. Duplicate icons are supported for inclusion in more than one folder.

Windows NT includes software support for NetBEUI (NetBIOS extended User Interface); the industry standard TCP/IP (Transmission Control Protocol/Internet Protocol), for access to Internet; and DCE-compatible Remote Procedure Calls (RPC). If you're already on a LAN, *Windows NT* will probably work fine as a node because the drivers needed to run an *NT* machine on a Novell *NetWare* network come in the box. *Windows NT* can also operate with any of these networks: Microsoft *Windows for Workgroups*, Microsoft *LAN Manager*, DEC *Pathworks* and Banyan *VINES*.

Through a special platform called the *Windows NT* Advanced Server, *Windows NT* can compete with *NetWare* as an advanced network operating system for large and complex LANs. With its Advanced Server, Microsoft is setting its sights on a loftier goal: handling large-scale server applications and wide-area operations that have been the traditional domain of mainframes and minicomputers. The Advanced Server will incorporate the centralized client-server networking features of Microsoft's *LAN Manager*, but with greater security and stability.

The electronic-mail software included in *Windows NT* lets you communicate with other users over any connected network. Based on Microsoft's *Windows for Workgroups*, it lets you send and receive electronic messages (e-mail), attach application files (such as spreadsheets and word-processor documents) to your messages, find messages in your mailbox according to a search criteria, organize and store messages in folders and print messages.

The Bottom Line

OS/2 and *Windows NT* probably won't interest the average home or small-office PC user right away. Unless you convert your software to 32-bit versions, your existing 16-bit software will run slower than it does under *Windows* 3.1—which defeats the purpose of upgrading to a more powerful platform.

Windows NT, however, may prove attractive to users in large offices who want to maximize the use of their large networked mainframe computers. OS/2's greatest appeal will, no doubt, be with custom application users, such as large pharmaceutical houses and CAD/CAM designers and manufacturers. But even these businesses are taking a wait-and-see attitude. Most experts don't expect either of the 32-bit platforms to make substantial inroads for at least two years, if then.

The bottom line is that, for now, most desktop PC users will find few uses for OS/2 or *Windows NT*. Personal-productivity software, such as database and word-processor applications, will be the last to be ported over to 32 bits. So if you're just running personal-productivity applications, stick with *Windows* 3.1 or wait for *Windows* 4.0. ■

In The Wings: *Windows* 4.0

For users who find OS/2 and *Windows NT* overkill, Microsoft is already working on another, scaled-down, 32-bit *Windows* operating system, which will have preemptive multitasking that lets you type in your word processor and print at the same time. Code-named "Chicago," it's expected to be released in early 1994 as *Windows* 4.0.

Early indications are that *Windows* 4.0 will merge the features of *Windows* and DOS into a full 32-bit operating environment centered around OLE 2.0 object-linking technology that allows users to access several applications from inside a single document. Other enhancements would include better 16-bit support and less overhead for faster performance.

The fate of *Windows* 4.0 depends on how long it takes for it to appear versus how long it takes for a wide range of OS/2- and *Windows NT*-based personal-productivity applications to appear. If real 32-bit applications reach the market quickly, OS/2 or *Windows NT* (or both) will take off for everyday use, with the nod going in *Window NT*'s favor because of the expected larger application base. If not, *Windows* 4.0 may end up the victor.

Despite aggressive marketing campaigns and promises of software aplenty, neither OS/2 or *Windows NT* may be the new DOS that you've have been waiting for. That title may fall to *Windows* 4.0.

Table 1. X-10 House Codes

	H1	H2	H4	H8	PLIX Decimal
A	0	1	1	0	6
B	1	1	1	0	7
C	0	0	1	0	4
D	1	0	1	0	5
E	0	0	0	1	8
F	1	0	0	1	9
G	0	1	0	1	10
H	1	1	0	1	11
I	0	1	1	1	14
J	1	1	1	1	15
K	1	0	1	1	12
L	1	0	1	1	13
M	0	0	0	0	0
N	1	0	0	0	1
O	0	1	0	0	2
P	1	1	0	0	3

Referring to Fig. 1, consider a House Code bit pattern of 0110. The first bit to be sent is a 0. So, on the first zero crossing after the last bit from the Start Code, a 0 is transmitted over the ac power lines. However, when the next power crossing occurs, the complement of that first bit, a 1, is sent instead of the next bit in succession.

In complement form, each bit transmitted requires one complete cycle. The first half of the cycle transmits the data bit, the second the complement of the data bit.

The Key Code is much like the House Code in that it's transmitted in complement form. It's provides the identification and commands to the X-10 modules, is comprised of five data bits and requires five complete cycles to transmit.

To ensure that data is received reliably, the X-10 protocol recommends that data packets always be transmitted in groups of two with three ac power-line cycles between each group of two codes. The exception to this rule applies when Bright and Dim codes are sent. Consecutive Bright and Dim codes should be transmitted continuously (at least twice) with no gaps between codes. Table 1 provides a complete list of all acceptable House and Key Codes and Table 2 the Key Codes currently defined in the X-10 protocol.

Transmission Theory

You already understand the basic theory behind the transmission of X10 codes. In essence, each bit is transmitted synchronous with the zero crossings of the 60-Hz ac power-line waveform. Some specifics remain to be covered to permit

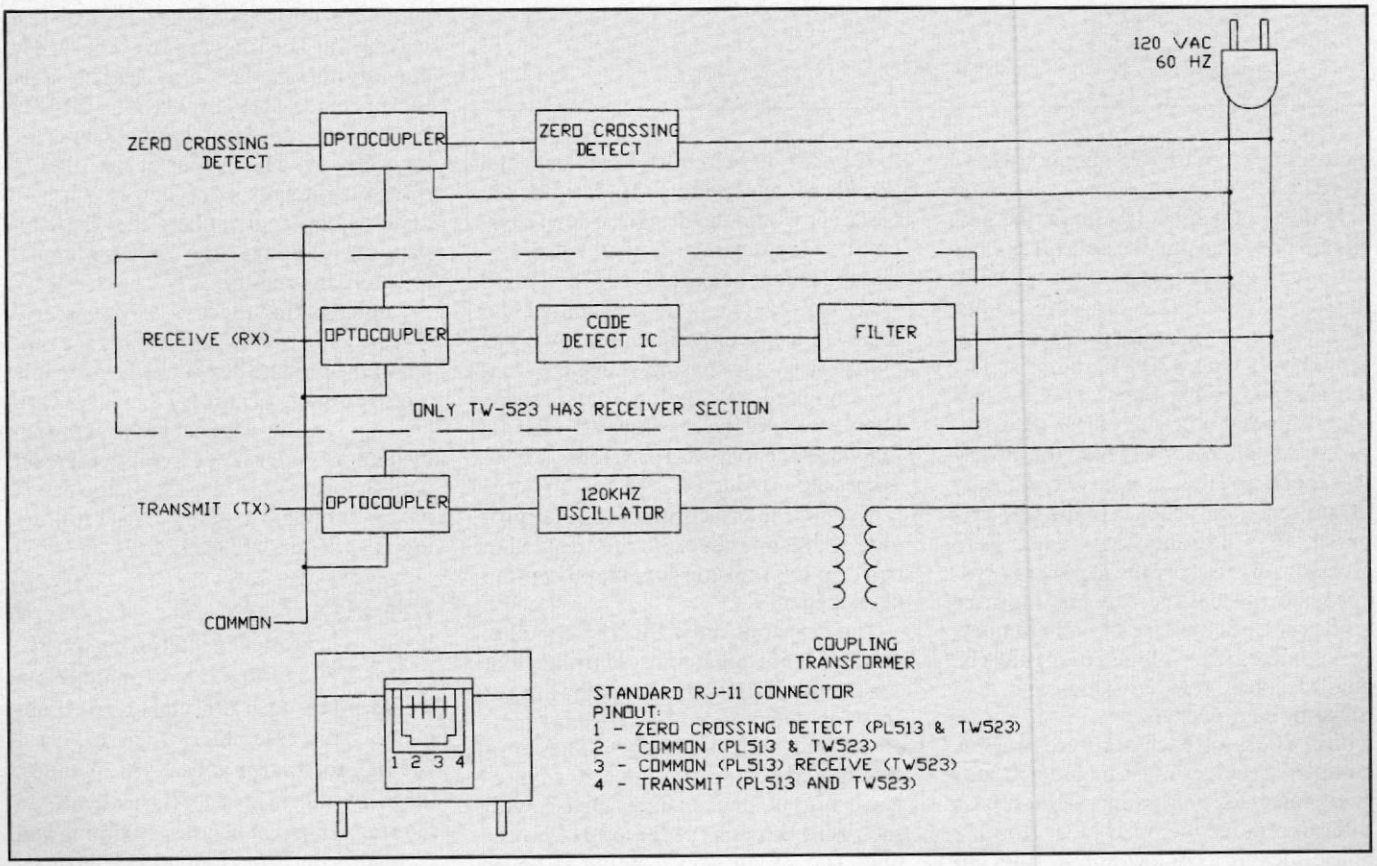


Fig. 2. Block diagram and pinout details for PL-513 and TW-523.

Table 2. X-10 Key Codes

	D1	D2	D4	D8	D16	PLIX Decimal
1	0	1	1	0	0	6
2	1	1	1	0	0	7
3	0	0	1	0	0	4
4	1	0	1	0	0	5
5	0	0	0	1	0	8
6	1	0	0	1	0	9
7	0	1	0	1	0	10
8	1	1	0	1	0	11
9	0	1	1	1	0	14
10	1	1	1	1	0	15
11	0	0	1	1	0	12
12	1	0	0	0	0	13
13	0	1	0	0	0	0
14	1	0	0	0	0	1
15	0	1	0	0	0	2
16	1	1	0	0	0	3
All Units Off	0	0	0	0	1	16
All Lights On	0	0	0	1	1	24
On	0	0	1	0	0	20
Off	0	0	1	1	1	28
Dim	0	1	0	0	1	18
Bright	0	1	1	0	1	26
All Lights Off*	0	1	1	0	1	22
Extended Code*	0	1	1	1	1	30
Hail Request*	1	0	0	0	1	17
Preset Dim*	1	0	1	X	1	21
Extended Data*	1	1	0	0	1	19
Status = On*	1	1	0	1	1	27
Status = Off*	1	1	0	1	1	23
Status Request*	1	1	1	1	1	31

*These functions are not supported by either the TW-523 or PL-513 modules.

a custom design to transmit and receive X-10 codes.

In designing an X-10 circuit, the goal should be to transmit the data bit as close to the zero crossing as possible but within 200 s of the zero-crossing points. When a bit is transmitted, a 1 is represented by a 1-ms 120-kHz burst, a 0 by the absence of the burst. This burst is superimposed onto the 60-Hz ac waveform. At the receiver end, the 60-Hz waveform is filtered out, leaving only the presence and absence of the 120-kHz bursts to represent X-10 data. Zero crossings are used by the receiver to synchronize the data stream, acting as a crude type of clock for digital circuitry.

A complete transmitter circuit must be able to detect zero crossings and then transmit data packets one bit at a time within 200 s of each zero crossing. A complete receiver must include circuitry to detect zero crossings, a high-pass filter to remove the 60-Hz waveform, a phase-locked-loop or notch filter to detect the presence or absence of 120-kHz bursts and circuitry to validate and decode the incoming X-10 data packet. Not only would the design of such a circuit be very detailed, it would also violate the patent held by X-10 USA. X-10 USA's protocol can be transmitted and received using only the company's circuitry or by purchasing a license from the company. Realizing that perhaps people would like to develop circuits that interface to its X-10 systems, the X-10 people provide PL-513 and TW-523 power-line interface modules that provide all the interface circuitry needed to perform the transmission and reception described above.

The transmit-only PL-513 module, used to build a custom controller that sends only X-10 codes, contains a zero-crossing detector, a 120-kHz burst generator and optical isolators. The zero-crossing detector provides a square-wave output that changes phase with each zero crossing of the 60 Hz power line. The square-wave output is never more than 100 s out-of-phase with its associated power-line waveform, making it a reliable source for synchronizing bit transmission with zero crossings.

The input envelope (120-kHz burst width) is 1 ms in duration. This means that the circuitry external to the PL- 513 must detect the zero-crossing waveform and then transmit a 1-ms logic high that will be encoded into a 120-kHz burst within 50 s of the zero-crossing detect.

Although this timing might seem a bit tight, an eight-bit microprocessor running an assembly program could easily meet the demand. Higher-level language programs, even on faster computers, probably couldn't maintain the microsecond-level timing required for a direct interface to the PL-513 module.

The TW-523 module is exactly like the PL-513, except that it adds built in receiver section to the elements already discussed. This module provides two-way communication over the ac power lines.

As illustrated in Fig. 2, the receiver section of the TW-523 consists of a zero-crossing detector, a filter and a code-detect IC. The zero-crossing detector is the same as the one used for the transmitter in the PL-513 module. It provides a square wave output within 100 s of the actual zero crossing of the ac power-line waveform. The filter removes the 60-Hz waveform from the signal, leaving only the presence and absence of 120-kHz bursts that represent logic 1s and 0s, respectively. The output of the filter is directed into the code-detect IC. The code-detect IC deciphers the X-10 bit pattern and outputs the X-10 command to external circuitry.

Implementing the TW-523 to receive X-10 codes simply requires that the output of the module be checked every time a zero crossing occurs for received X-10 signals. A valid-data packet is signified by the start code. When a consecutive bit pattern is recognized as 1110, the external circuitry knows that the data following is an X-10 data packet.

The PLIX Chip

If you're interested in interfacing the PL-513 or TW-523 to a microcomputer but would prefer to use a higher-level language than assembler, such as C or BASIC, to control your X-10 modules, the PLIX chip is for you. Basically it's an interface chip that handles all timing and validation functions required for an interface to the PL-513 or TW-523 modules.

The PLIX (Power Line Interface for X-10) chip uses a standard bus-style interface. It "talks" to the microprocessor using the D0 through D4 data lines as the data bus, CS as chip select, RDY as the all clear signal and DIR as the direction control for reading and writing data from and to the part. Pinout details for this chip are given in Fig. 3.

The PLIX chip has its own internal clock, derived from an external crystal with a frequency of 3.6864 MHz. This value is important for the proper operation of the chip and should be used consistently.

Data can either be written to the PLIX chip for transmission across the power lines or read from the PLIX chip representing the last valid X-10 code received. Reads and writes are determined by the status of the DIR pin.

Writing data to the PLIX is a straightforward procedure. First, the main processor or computer puts the data to be sent to the PLIX chip on the D0 through D4 lines. DIR is asserted to its proper level for a write operation, and CS is brought high to signal the PLIX chip that data is on the data bus.

The PLIX chip signifies that it has started reading the data on the data bus by asserting the RDY line. When the chip is finished, it drops RDY, and the main processor completes the rest of the cycle. The write operation is terminated when the main processor or computer releases the data, DIR and CS lines.

The first write done to the PLIX chip is a synchronize command that resets that internal counters and pointers of the chip. It consists of three or more 0s and a 31. Once the PLIX chip has been initialized, it's ready to receive commands.

Commands are sent to the PLIX chip in three-byte packets. The first byte has a range of 0 through 15 and signifies the House Code. The second byte has a range of 0 through 31 and represents the Key Code. The final byte is the number of times to repeat the command.

Keep in mind that each 11-bit X-10 data packet should be sent twice in a row, with three power line cycles before the next packet is sent, unless the packet is a bright or dim code, in which case, the packets are sent consecutively without spaces. This means that every command that is sent to the PLIX chip should have a repeat byte of two for normal on/off commands and two to 30 for bright and dim commands.

Once this packet is received, the PLIX

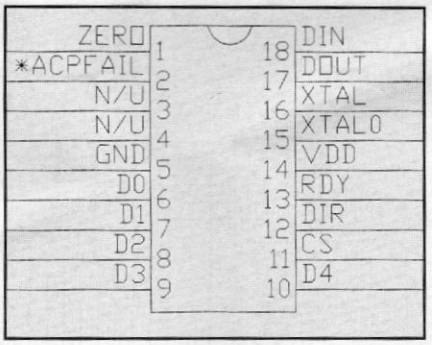


Fig. 3. Pinout details for PLIX chip.

chip sends out a formatted data packet in the X-10 protocol over the power lines. The power line interface portion of the PLIX chip has all of the signals necessary to communicate with either the TW-523 or PL-513 modules. D_{Out} is the data output that represents the data packet to be transmitted. It serially transmits the individual bits of the data packet, using the Zero signal (the zero-crossing detection square wave) as the clock. D_{In} receives data serially from either of the interface modules, using Zero as the clock.

When a valid X-10 data byte is received, it is stored in the PLIX chip until it has been read by the controlling microprocessor or another X-10 packet has been received. To ensure no receptions are lost, the PLIX chip should be polled at least every 300 ms.

To read the received code from PLIX chip, the main processor or computer must assert CS high and leave DIR low. The PLIX chip will then place the first data byte on the data bus (D0 through D4) and bring RDY high. When the main processor or computer senses that RDY is high, it reads the data and asserts CS low. Then the chip releases the data bus and brings RDY low. The next read cycle can then begin.

When the received code is read from the PLIX chip, the most significant bit, bit 4, represents whether it is a newly received code or simply the same code as the one that was read last time. Once the received code has been read, bit 4 is reset to signify that the code in memory now is an old code.

The last PLIX chip feature to be discussed the is /ACPFail signal, which is asserted when no ac power is connected to the interface module. It can be used to interrupt the microprocessor in the event of power loss. This is particularly useful

PARTS LIST

Semiconductors
D0 thru D4—1N4148 diode
LED1—TIL220 or similar light-emitting diode
Q1—LB7805 fixed 5-volt regulator
Q2—PN2222 silicon npn transistor
Q3—PN2907 silicon pnp transistor
U1—PLIX chip (see text)
U2—74HC241
Capacitors
C1,C2—22-pF ceramic disc
C3—10µF, 15-volt electrolytic
V4—47µF, 15 volt electrolytic
Resistors (1/4-watt, 5% tolerance)
R1 thru R4—10,000 ohms
R5—100,000 ohms
SIP1—4,700-ohm SIP resistor pack
Miscellaneous
J1—25-pin male sub-D connector
J2—Four-pin RJ-11 jack
J3—20-pin header
J4—Not used
J5—Two-position screw block
X1—3.6864-MHz crystal
Printed-circuit board; sockets for ICs; suitable enclosure; machine hardware; hookup wire; solder; etc.

Note: The following items are available from Montgomery Engineering, 3845 S.W. 25, Oklahoma City, OK 73108 (tel.: 405-681-9979): TW-523 module, $34.99; PL-513 module, $25.99; PLIX chip, $25.99; demo board kit with software, $36.99; example software on floppy disk, $5; technical information packet on X-10 theory and operation, free.

For more information on the X-10 system, contact: X-10 USA Inc., 91 Ruckman Rd., Box 420, Closter, NJ 07647-0420 (tel. 201-784-9700).

For quantity pricing on the PLIX chip and demo board, contact: MicroMint, Inc., 4 Park St., Vernon, CT (tel.: 203-871-6170).

Table 3. Ports Defined in Generic IBM Parallel Printer Port

Port	D7	D6	D5	D4	D3	D2	D1	D0
Data Out	NU	NU	NU	D4	D3	D2	D1	D0
Pins	9	8	7	6	5	4	3	2
Control	NU	NU	NU	NU	POW	DIR	U2	CS
Pins	—	—	—	—	17	16	14	1
Status	U2IN	ACPF	U2IN	U2IN	RDY	NU	NU	NU
Pins	11	10	12	13	15	—	—	—

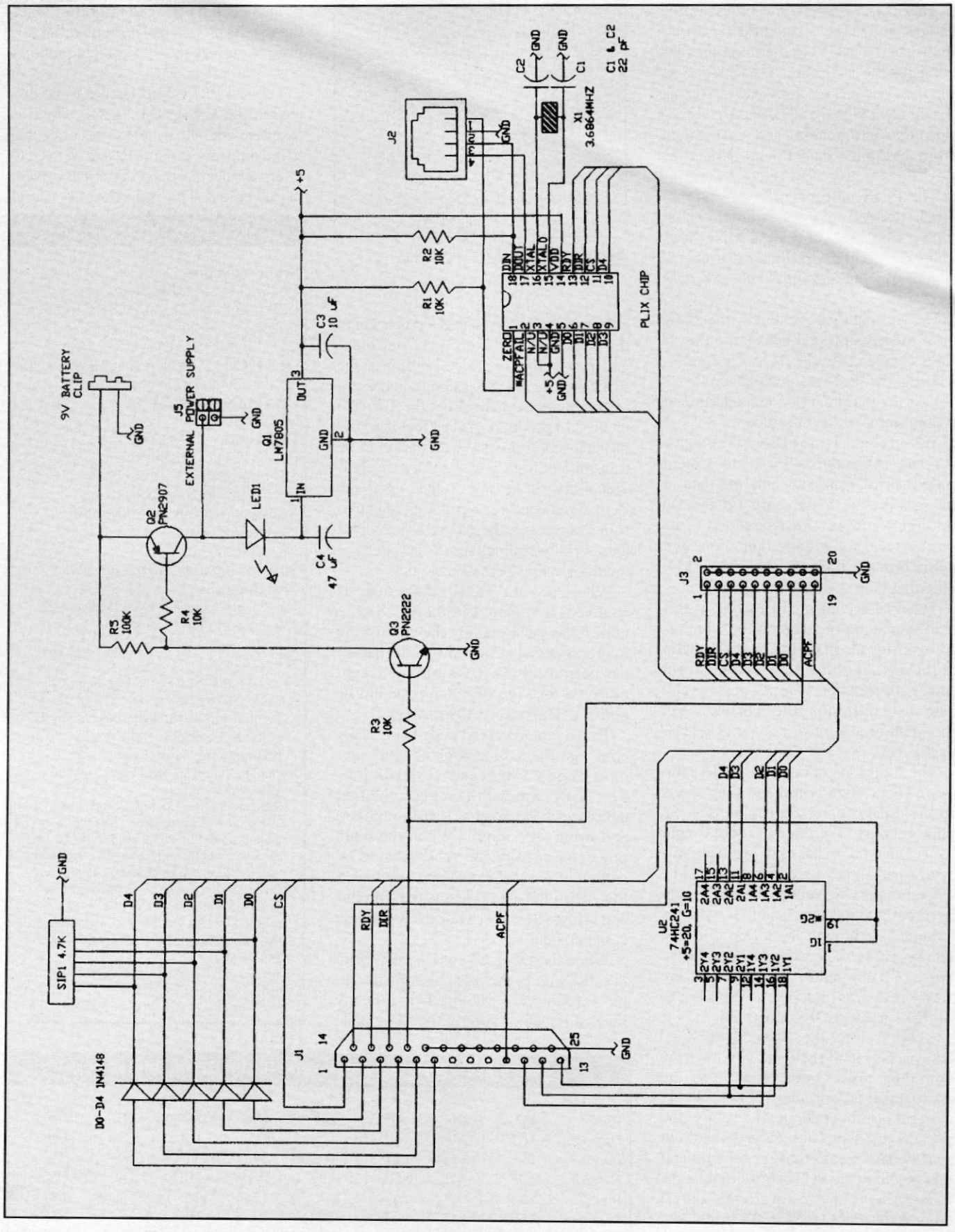


Fig. 4. Complete schematic diagram of X-10 Computer Interface circuitry. (Copyright Micromint Inc., reprinted with permission of same)

in battery backed-up systems because it can signal the computer to transfer to the battery supply and begin to backup data in its memory.

Computer Interface

Shown in Fig. 4, is the complete schematic diagram for an X-10 interface circuit that plugs into the parallel port of an IBM PC/compatible computer. In this circuit, *J1* provides the signals from the computer's parallel port to the interface board. The X-10 computer interface makes use of all three ports defined in a generic IBM parallel printer port, as detailed in Table 3.

In Table 3, if a port bit isn't used in this design, it is marked "NU" for not used. Below each port description is a row of pin definitions. This translates each bit in the port to an actual pin of *J1*. For example, the bit used to turn on transistor *Q3* is Bit 3 of the control port. Looking directly below this bit in the Pins row, you'll find that is pin 17. You can confirm this on the schematic diagram.

The topics below break down the operation of the X-10 computer interface to allow you to understand how the circuit works and how to program the circuit.

- **Data Out.** The data-output port is used to transmit data to the PLIX chip. Because the PLIX chip uses only data bits D0 through D4, bits D5 through D7 are left not used (NU in Table 2). The data port is located at I/O address 278H, 378H or 3BCH.

- **Control.** The control port is used to interface to the control lines of the PLIX chip and to select the data bits to be shifted into the input lines by *U2*. Bit 3 is raised to active-high to turn on power to the circuit when operating under battery power. Bit 2 is set to control the direction of data transfer, either to or from the PLIX chip. Bit 1 is used to control which bits of data are present on the inputs for the status port. Finally, Bit 0 is brought active high to enable the PLIX chip. The control port is located at I/O address 27AH, 37AH or 3BEH.

- **Status.** The data-out port isn't bidirectional and, therefore the status port must be used to input the information from the PLIX chip to the computer. However, only a limited number of input bits are usable on the status port, five to be exact. Of these, two must be used to detect the presence of ac power via ACPF (Bit 6) and the ready condition of the PLIX chip via RDY (Bit 3). This leaves three bits (Bits 7, 5 and 4) for input. U2 controls the input of data from the PLIX chip into these three bits. The status port is located at I/O address 279H, 379H or 3BDH.

- **Power Supply.** There are two ways to power the X-10 computer interface. You can power it externally from a 9-volt dc power adapter via *J5,* or you can power it from a 9-volt battery. When operating the circuit on battery power, the program must set Bit 3 of the control port to active high to provide power to the circuit. This applies positive bias to the base of *Q3*, turning on this transistor, which, in turn, biases the base of *Q2*. When Q2 is conducting, it provides power to voltage regulator *Q1* through *LED1*.

To conserve battery life, Bit 3 of the control port should be reset to inactive low when the controlling program is terminated.

- **Programming.** The programming of the PLIX chip was covered in detail above. I'll describe here the specifics for controlling the X-10 computer interface via the generic IBM parallel port.

Initially, you must determine the base address for the board on your system. For instance, my parallel printer port is located at 378H, meaning that my status and control ports are located at 379H and 37AH, respectively.

Once the ports are defined, the first step in programming the X-10 computer interface is to provide power to the board by enabling Bit 3 of the control port. With power applied, the PLIX chip must be initialized with the sequence 0-0-0-31. To do this, a generic transmission procedure must be developed. Data is first placed on the data-output port (I/O address 378H on my machine). Then DIR must be asserted high by setting Bit 2 of the control port. Finally, CS is brought to active-high by setting Bit 0 of the control port. When the PLIX chip starts reading the data, it brings the RDY line high and then low again when it's done. The condition of the RDY line is sensed at Bit 3 of the status port.

Using the write procedure described above, it's possible to send the initialization sequence. Further write operations can be performed for module selection and control, as described above. Remember that for the PLIX chip to stay synchronized, you must send three bytes on every write sequence: a House Code, a Key Code and the number of times to repeat.

To read data from the PLIX chip, the program brings CS high and leaves the DIR pin low. The PLIX chip then places the data on D0 through D4 and brings the RDY line high (which is read at the status port on Bit 3). The program should then read the data and bring CS low.

U2 is used to shift the five data bits into the three input bits available on the status input port. When Bit 1 of the control port is held high, U2 is selected such that D0 through D2 from the PLIX chip are placed on Bits 4, 5 and 7 of the status port, respectively. When Bit 1 of the control port is held low, U2 is selected such that D3 and D4 from the PLIX chip are connected to Bits 4 and 5 of the status port, respectively.

During a read cycle, the data on D0 through D4 from PLIX is valid when the RDY line is brought high by the PLIX chip. While this data is valid, U2 should be read through the status port to get D0, D1 and D2. Then, U2 is re-selected by setting Bit 1 of the control port to 0 to read in D3 and D4. The two inputs are then put together to form the input byte.

Each read cycle should input two bytes from the PLIX chip to keep it synchronized with the program. The first data byte is the last received House Code, and the second data byte is the Key Code.

Because the control program is too long to publish here, I'm making it available on floppy disk (see the Note at the end of the Parts List for details).

Construction

There's nothing critical about building the X-10 Computer Interface. Therefore, you can use any wiring method you have at your disposal. If you prefer to build the project on a printed-circuit board, you can purchase a complete kit of parts with the controlling software and a scheduling program from the source given in the Note at the end of the Parts List.

If you choose another method of construction, just make sure you follow common prototyping practices. Keep all control lines as short as possible. Solder carefully to avoid creating cold-solder joints and solder bridges between closely spaced components. You'll save a lot of time later if you verify the electrical and mechanical integrity of each connection for as you solder it.

You can use a standard phone cable for connection between the X-10 computer interface and the TW-523/PL-513 module, but beware that it's a straight-through cable. Because this isn't a requirement for most telephones, most phone cable manufacturers don't ensure that all wires go straight through. If you don't use a straight-through cable, your circuit won't work.

If you choose to use the PL-513 module instead of the TW-523, you'll also have to disconnect pin 3 of *J2* from the PLIX chip and ground it. The reason becomes obvious if you examine Fig. 3.

Conclusion

In this article, I've introduced you to the concepts of X-10 and the theory behind X-10 operation and shown you how to build an complete interface circuit to computerize your X-10 system. If you have any other comments or questions, drop me a line or call me at the address or telephone number given in the Note at the end of the Parts List. I'd like to hear about any home-control projects any of you out there are brewing. ■