MIDI LIGHT CONTROLLER

Our MIDI light controller can turn an ordinary musical performance into a concert!

EDWARD J. KEEFE JR.

MOST SMALL BANDS USUALLY HAVE such a hard time paying for travel and instruments that their shows must forgo any type of sophisticated lighting. The lights provided by the clubs and bars where they play do little to showcase a band's talent. With the MIDI (Musical Instrument Digital Interface) light controller presented here, that can all end. A simple microprocessor with a handful of components can transform their act into a full fledged "concert." Everything that is needed to synchronize lights and music already comes out of the MIDI port of MIDI keyboards. This circuit will make use of that information and enhance the show.

MIDI is a communications protocol originally created for interfacing synthesizers and other electronic devices. It has evolved into a communications standard that is used in all phases of audio and video production. MIDI allows devices to talk to each other with different types of control and data values. The values can be either CHANNEL, SYSTEM, REAL-TIME, or SYSTEM EXCLUSIVE messages. MIDI communication is achieved through multi-byte messages, each consisting of one STATUS byte followed by one or two DATA bytes. Real-time and exclusive messages are exceptions to that rule.

Two types of data bytes are sent over the MIDI cable, STATUS and DATA. Status bytes are eight-bit binary numbers in which the most-significant bit is set to "1." A status byte sets the function of the data bytes that follow it, and a new status byte is required for each new action. The MIDI specification also outlines RUNNING STATUS. That defines the action for all data bytes that follow a status byte, until a new status byte is sent. That way more information can flow down the cable. Data bytes are eight-bit binary numbers in which the most-significant bit is set to "0."

The MIDI light controller presented here reacts to NOTE ON, NOTE OFF, START, STOP, and CONTINUE status bytes. The data bytes are used to determine which light to control and how

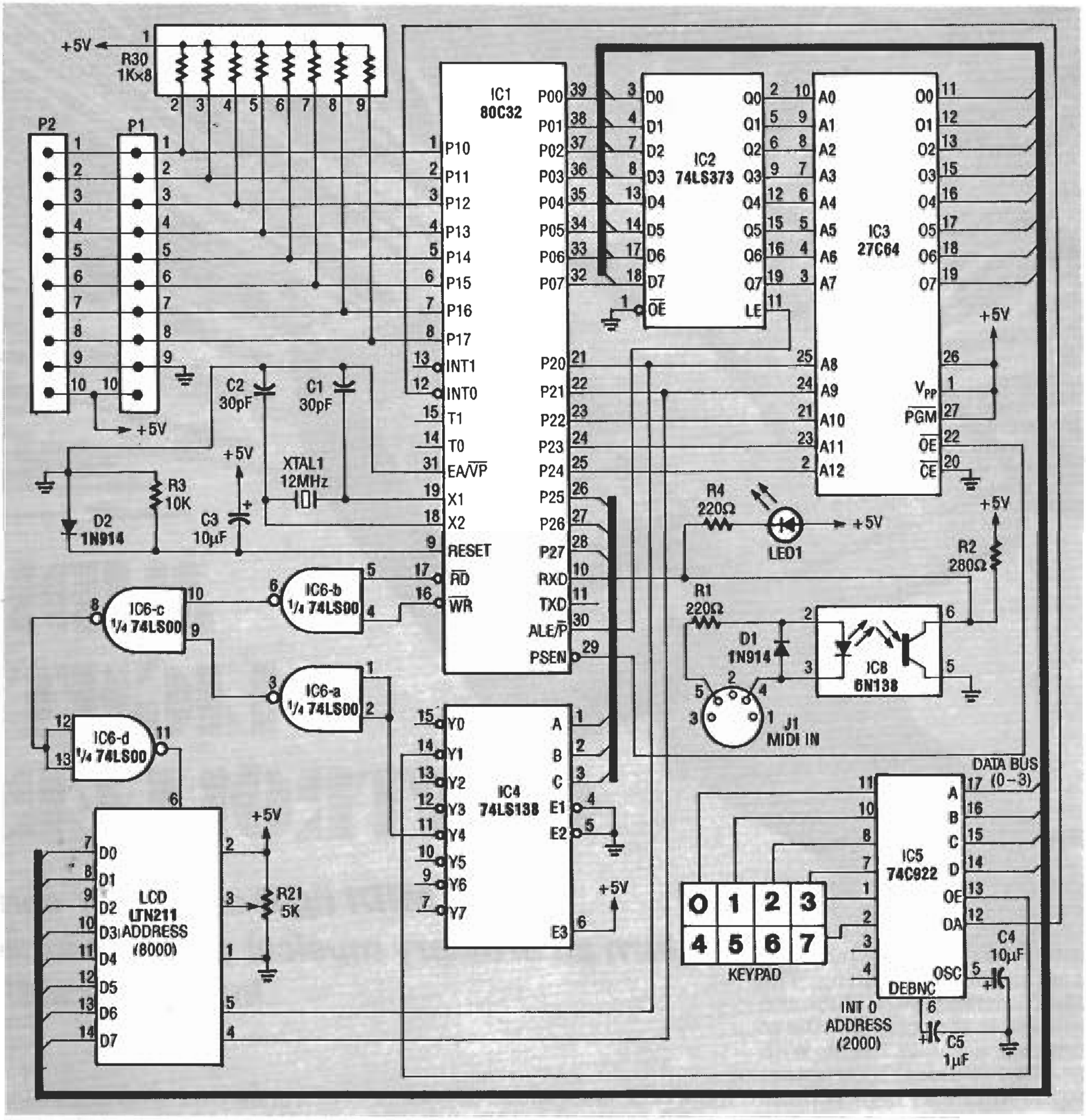


FIG. 1—SCHEMATIC OF THE MAIN PROCESSOR BOARD. The MIDI signal is fed in through an optocoupler (IC8) to the microcontroller (IC1).

to respond to different conditions on the MIDI. It can operate on any of the 16 MIDI channels, and will respond to note information from octaves 1–8. The MIDI light controller is user-configured. You have control over whether the lights will latch, toggle, or stay off in reaction to note information on the MIDI by setting the OUTPUT TYPE keyboard selection.

When a STOP command is received (STOP, START, and CONTINUE commands synchronize all elements of a MIDI system), the lights can either all go off, all go on, or light selectively, depending on the user's STOP DEFINE parameters. That is useful for creating a "scene" between songs and during breaks on stage. The CONTROL OCTAVE determines which range of notes will be used to control the lights. The operating channel is set by selecting CHANNEL DEFINE. The message CHANGE CHANNEL # will be displayed on line 1. Line 2 will toggle between "1" (channels 1–8) and "2" (channels 9–16). You must select "1" or "2" for the desired range of channels (1–8 or 9–16) and select the desired channel.

The lights are controlled by note information on the MIDI. They can respond to actual notes of a song or notes that are placed in the sequence specifically for the light controller. To use existing notes, you would select the channel and octave from the sequence, and enter them on the keyboard. If the lights are to be controlled by a separate track, you would enter the note information for each light, keeping all notes in one octave. Timing and synchronization is provided by a sequencer. (A sequencer is any instrument that can store and read back MIDI data.)

The light controller will also work directly from a keyboard without the aid of a sequencer by connecting a keyboard's MIDI output to the jack on the light controller. As you play the keyboard, the appropriate lights will illuminate.

The circuit

Figure 1 shows a schematic of the main processor board. The MIDI signal is fed in through IC8, an HP 6N138 optocoupler. Any optocoupler can be used as long as it has a rise time of less than 2 microseconds, and can turn on with less than 5 mA. The MIDI signal has the following operating specifications: 31.25-kHz baud rate, asynchronous, 1 start bit, 8 data bits, and 1 stop bit, with a period of 320 microseconds per serial byte. The signal is well suited to the serial portion of the Intel 80C32 microcontroller (IC1) that has built-in transmit and receive serial ports, 128 bytes of internal RAM, two interrupt lines (each with programmable priority), and external memory addressing capability up to 64K. The CMOS version of the 8032 was used because of its higher speed and lower power consumption.

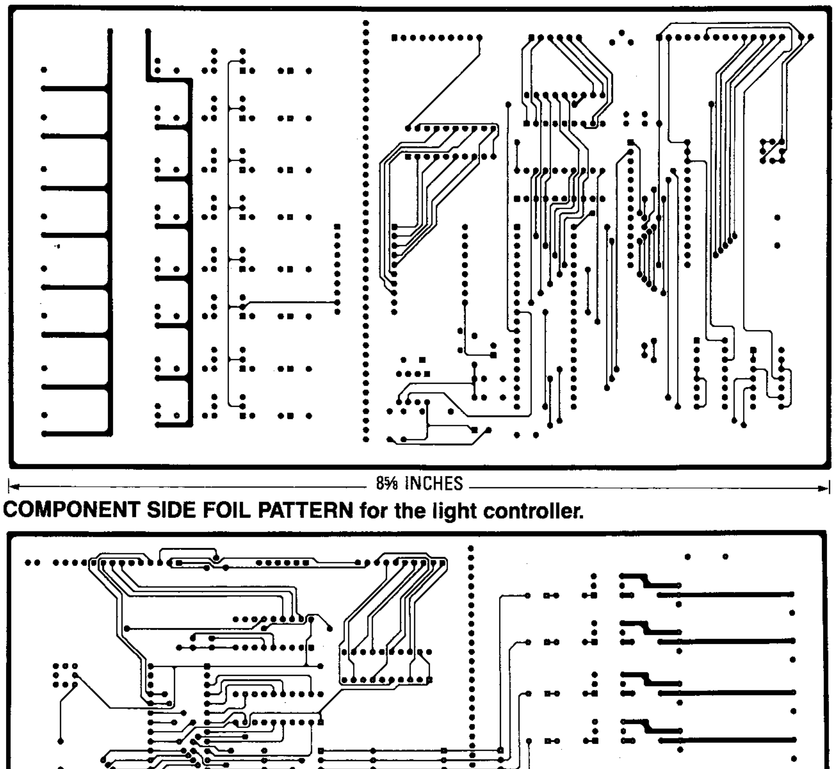
The output of optocoupler IC8 enters the serial receive port (pin 10) of microcontroller IC1. Software in the 27C64 EPROM (IC3) controls port setup and baud-rate selection. At power up, a small reset circuit (R3, D2, and C3) initializes the microcontroller. An oscillator is made up of a 12-MHz crystal (XTAL1) and two 30 pF capacitors (C1 and C2).

The address and data buses of the microcontroller are multiplexed. To remove the low-order address information, a 74LS373 8-bit latch is used. The 74LS373 is strobed with the ALE (ADDRESS LATCH ENABLE) signal from the microcontroller, and address data is removed. The 27C64 EPROM (IC3) is used to store program data. The EPROM is enabled by the microcontroller's PSEN (PROGRAM STORE ENABLE) line.

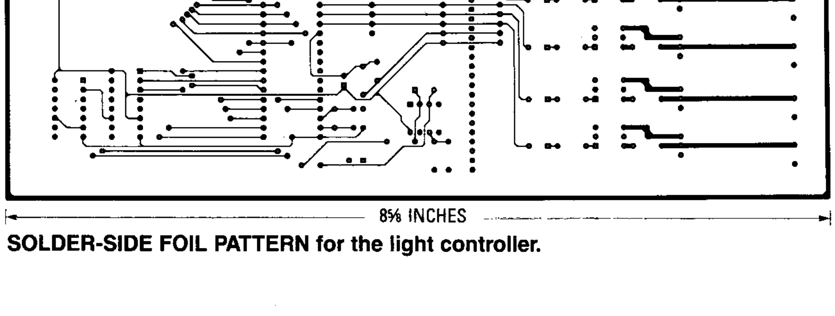
The keyboard and LCD circuits both require an address signal to interact with the data bus. A 74LS138 (IC4) is used to generate a signal when certain addresses are reached (1000h, 2000h, 4000h, 6000h...). The keyboard is mapped at external address 2000h. The keys are scanned by the 74C922 keyboard controller chip, IC5. All keyboard action, including debouncing, is handled by IC5. The scan rate is controlled by C4, and debouncing by C5.

The LCD is mapped at external address 8000h. An Optrex LTN211 two-line LCD module is used to display current channel and mode information. Also, user-definable selections are displayed there. Function data is written to the module at address 8000h, and display data is written at 8200h. Address selection and read/write functions are established by IC6, a 74LS00. Contrast of the LCD is altered by R21. Output from the microcontroller is on port 1, pins 1–8. A resistor network, R30, is used to pull-up output lines. The lines drive the Triac and LED sections.

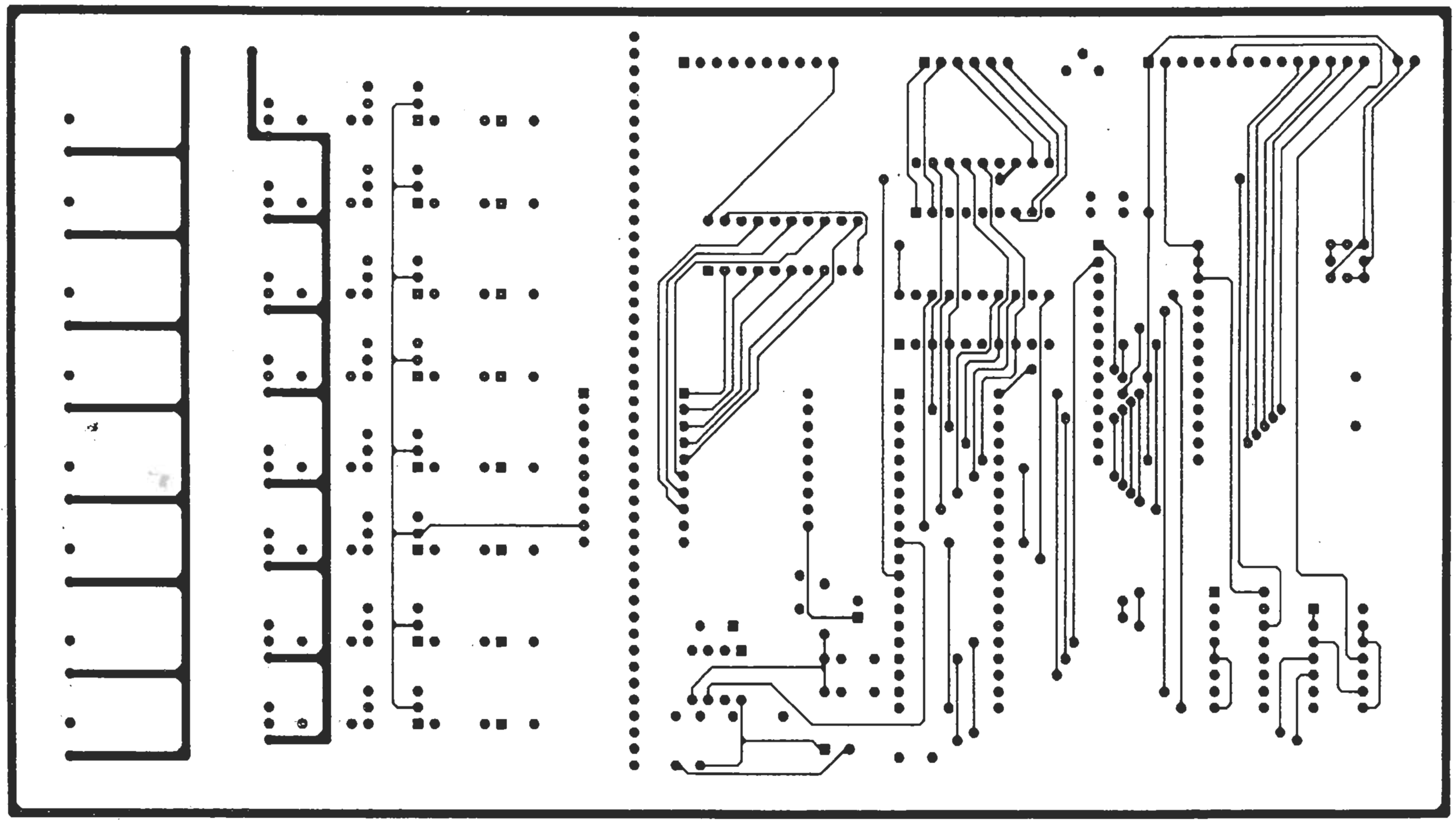
Figure 2 shows the high-power output section of the light controller. Eight identical drivers are used to control the output channels. An output from IC1 is passed through a 1N914 diode and a 120-ohm resistor that drives an MOC3010 Triac-driver optocoupler. Output from the optocoupler is sent to a high-power Triac. The circuit is designed with 6-amp Triacs, which are fused at 5 amps for added protection. The 120-volt AC input to the Triacs must



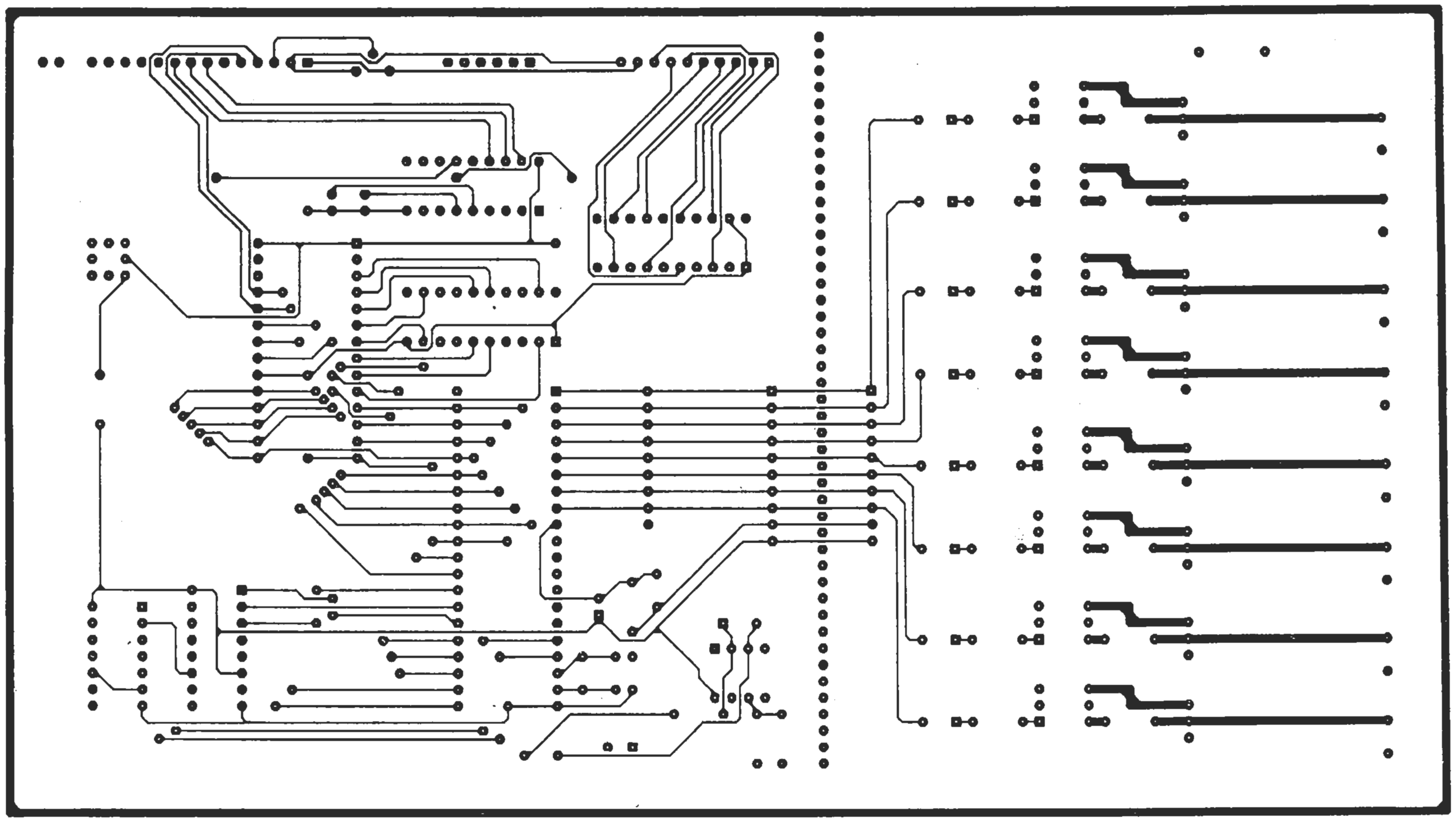
COMPONENT SIDE FOIL PATTERN for the light controller.



SOLDER-SIDE FOIL PATTERN for the light controller.



COMPONENT SIDE FOIL PATTERN for the light controller.



SOLDER-SIDE FOIL PATTERN for the light controller.

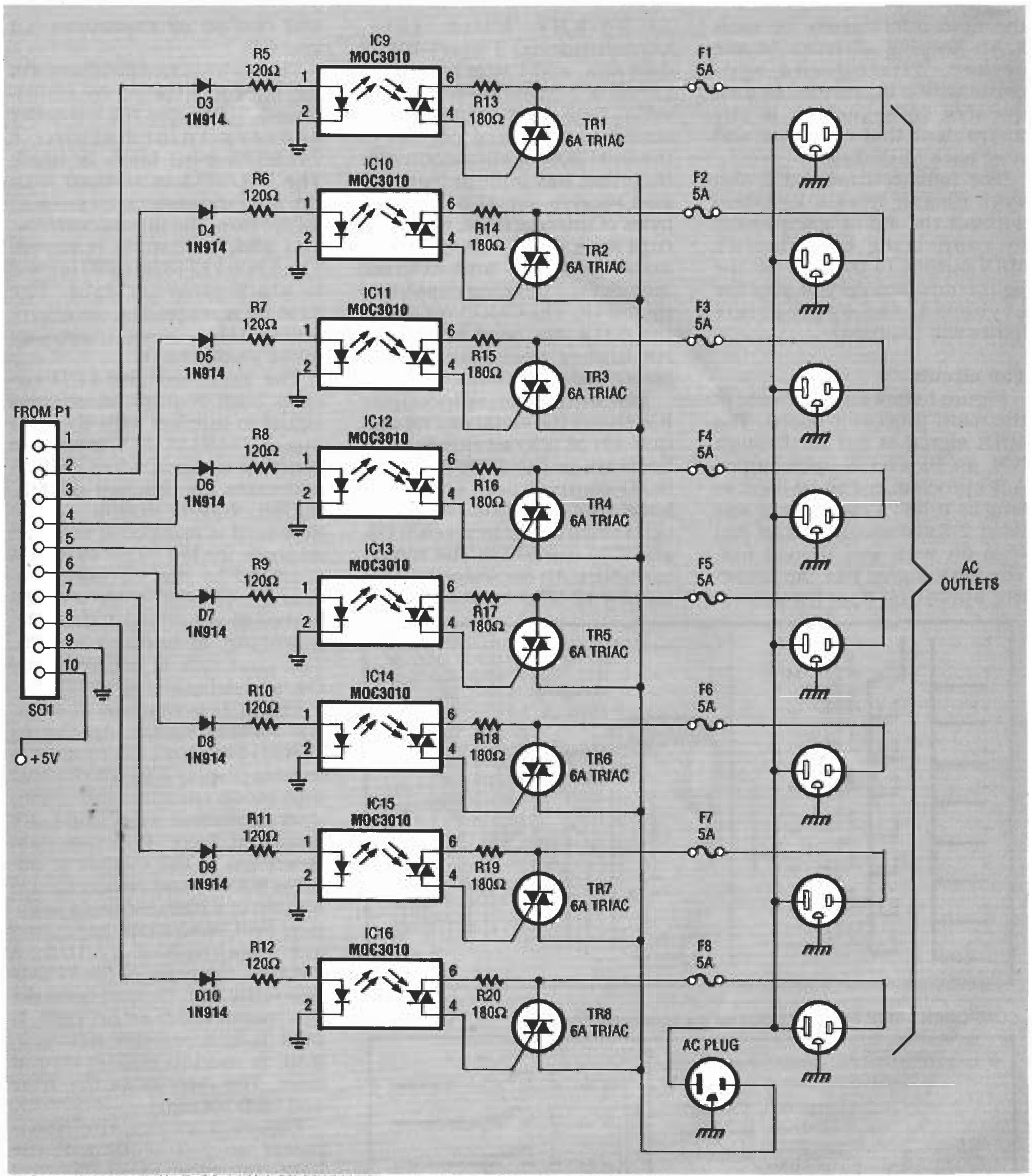


FIG. 2—HIGH-POWER OUTPUT SECTION. An output from the microcontroller is passed through a 1N914 diode, a 120-ohm resistor, an MOC3010 Triac-driver optocoupler, and finally to a high-power Triac.

be connected to a circuit that can handle the current. Power consumed by the light controller at 120 volts AC with all eight channels operating at full power is 4400 watts—that's almost 40 amps! Most household wiring is 15 or 20 amps per circuit breaker, so the AC input should be separated into multiple circuits to control that much power.

Figure 3 is the LED display section. It's used to indicate which output channel is active. It also helps you set up MIDI sequences without having to hook up any external lights.

A clean 5-volt DC power source is required for the light controller. The author used a self-contained 5-volt supply that includes a built-in transformer, rectifier, and regulator. The supply accepts a 120-volt AC input and outputs 5-volts DC. You can use a similar supply if you like, although they are more expensive. Otherwise any 5-volt supply will do.

Software

Software for the light controller is interrupt-driven. Upon reception of MIDI data, an interrupt is generated. The software jumps to the interrupt routine for the serial port. First the interrupt is cleared, then the byte is placed in the receive buffer. As more data comes in, it is buffered. The keyboard also generates an interrupt any time a key is pressed. As in the serial section, first the interrupt is cleared.

The main monitor loop of the program simply tests to see if data is in the receive buffer or the keyboard flag is high. If data is in the buffer, the byte is checked for a high MSB indicating a status byte. If it is a status byte, the status register is updated. The status will then remain until a new status byte is received. The program then continues the main monitor loop. If the byte is data, it is examined to determine the action to be taken. Depending on the current user-set mode of the controller, a light will be turned on, a light will pulse, or nothing will happen. If the data turns out to be bad, it is flushed.

After examining the data and acting upon it, the main monitor loop is resumed. When the key flag is high, the data in the temporary buffer is examined. The action desired is stored in the appropriate register. The main monitor loop is then resumed. All user-defined actions are entered on the keyboard. As options are entered, the LCD displays the choices available. The option being defined is displayed on the top line, and the selections are cycled through. Press the key for the desired action. If more information is needed for the chosen item, it will be presented on the LCD. The source code for the EPROM and some sample files to run on a sequencer are available on the RE-BBS as a self-unarchiving zip file called MLC1.EXE. A programmed EPROM is available from the source mentioned in the Parts List.

Construction

The PC board for the light controller can be made using the foil patterns we've provided, or it can be purchased from the source mentioned in the Parts List. A parts-placement diagram is shown in Fig. 4. Notice the row of pads to the left of the P1 header that divides the board down the center; the board can be cut there if you want to remotely locate the power section of the board and then run low-voltage wiring between the two sections.

If you are going to cut the board in two, do it before stuffing it, as it will be easier. Install the components according to Fig. 4, and check your work as you go. The pads marked "INV" on the board are for a voltage inverter (see parts list), if used. It that generates the proper voltages for the backlight on the LCD if the LCD you use has one. If it does have a backlight, and you wish to use it, install the inverter and connect the LCD's backlight terminals to the pads marked "BL" on the board.

Any kind of case will do for this project, as long as everything fits inside—keep in mind the power supply you will use and be sure to leave room for it. If you use a metal enclosure be

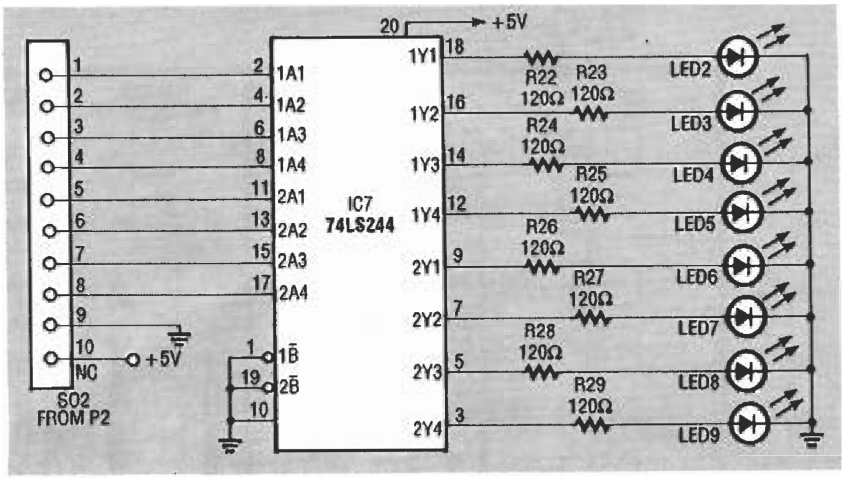


FIG. 3—LED DISPLAY SECTION. These LED's will indicate which output channel is active without having to set up the full-power lights.

All resistors are ¼-watt, 5%, unless otherwise noted.
R1, R4—220 ohms
R2—280 ohms
R3—10,000 ohms
R5, R6, R8, R10, R12, R15, R17, R19, R22–R29—120 ohms
R7, R9, R11, R13, R14, R16, R18, R20—180 ohms
R21—5000 ohms, potentiometer
R30—1K×8 SIP resistor
Capacitors
C1, C2—30 pF, mica
C3—10 µF, 16 volts, electrolytic
C4—10 µF, tantalum
C5—1 µF, tantalum
Semiconductors
IC1—80C32 microcontroller
IC2—74LS373 8-bit latch
IC3—27C64 EPROM
IC4—74LS138 3-to-8 demultiplexer
IC5—74C922 16-key keypad encoder
IC6—74LS00 quad NAND gate
IC7—74LS244 octal buffer/line driver
IC8—6N138 optocoupler
IC9–IC16—MOC3010 optoisolator
D1–D10—1N914 diode
TR1–TR8—any 6-amp Triac in a TO-220 case
LED1–LED9—Chassis-mount light-emitting diode, any color
Other components
XTAL1—12-MHz crystal
F1–F8—5-amp fast-blo fuse
P1, P2—10-pin header strip
SO1, SO2—10-pin header socket
J1—5-pin DIN socket
Miscellaneous: inverter (for use with LCD backplane, see text), 5-volt power supply (see text), project case, grounded AC linecord, 8 grounded AC outlets, wire, solder, etc.
Note: The following items are available from Audio Visual Imagery, P.O. BOX 332, Randolph, MA 02368:
• **PC board—$35.00**
• **Programmed 27C64 EPROM—$10.00**
A diskette with the sample files will be included with any order. Please add $1.50 S&H to any order. MA residents must add 5% sales tax.

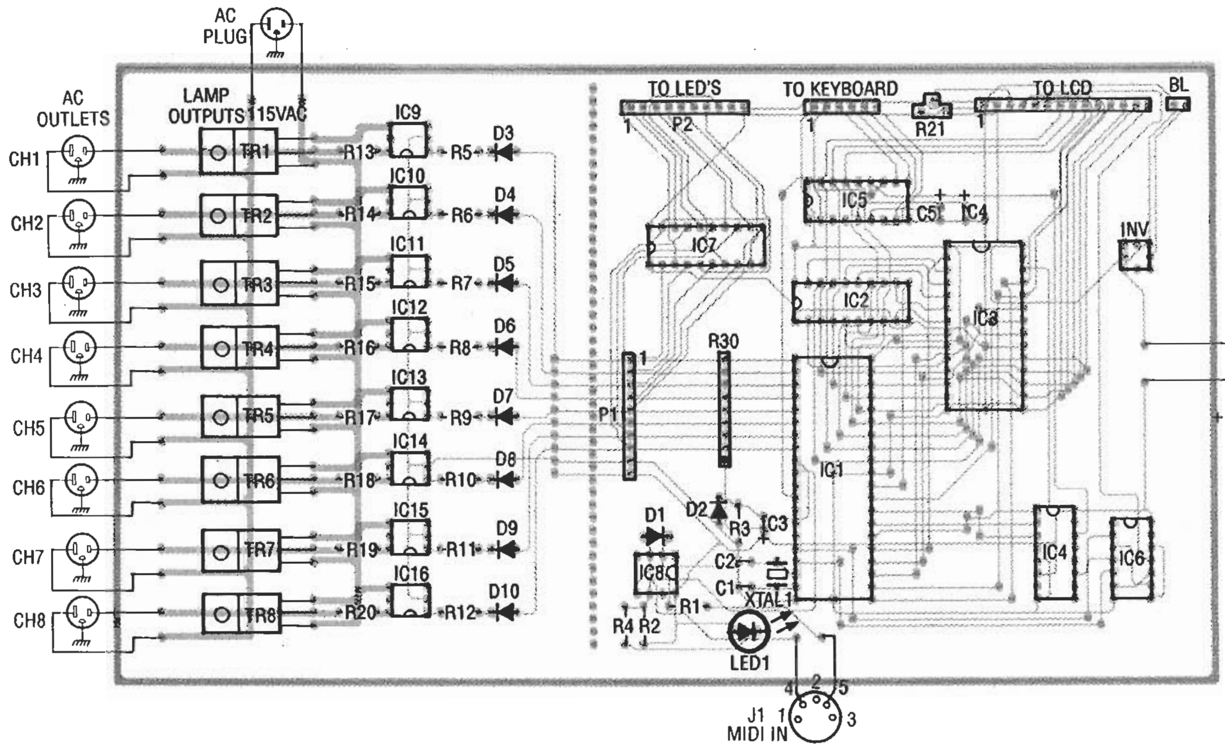


FIG. 4—PARTS-PLACEMENT DIAGRAM. The PC board can be cut down the row of pads to the left of the P1 header if you want to run low-voltage wiring between the two sections.

sure to ground it to the AC safety ground as shown in Fig. 2, and also properly ground all eight AC outlets.

As a reminder, the light controller can output about 5 amps per channel, and any wire common to all eight outputs must carry about 40 amps. The controller's internal AC wiring must therefore be chosen accordingly. Also, if your house wiring is rated 15 or 20 amps per circuit breaker, the AC input to the controller should be separated into at least two circuits. Figure 5 shows the current version of the PC board.

Testing

To test the light controller, simply adjust R21 to the middle of its range, and apply a clean 5-volt DC power source to the circuit. A message will appear on the LCD. Adjust R21 for the best-looking display. Plug a MIDI cable into the MIDI IN jack (J1) on the light controller. The other end should be placed in the OUTPUT jack of a sequencer or the THRU jack of a synthesizer. (If you're not using a sequencer, connect it to the OUTPUT jack of the keyboard.) At power up, the light controller is on MIDI channel 1. Make sure your MIDI equipment is sending notes out on channel 1. As notes are sent, LED1 will light, indicating that data is being sent over the MIDI. As notes are received and examined, the appropriate LED(s) will light (as long as the notes are in the CONTROL OCTAVE of the light controller).

If the circuit does not work, check the supply voltage. Make sure XTAL1 is oscillating. Check for a 1-microsecond reset pulse on pin 9 of IC1 at power up. If the reset circuit does not produce a pulse, the circuit will not start up. At power up, the eight LED's should light and then extinguish. If that does not happen and the clock and reset circuits are working, check the wiring of the LED's and the Triac section. If nothing happens still, the problem is with the microcontroller. Check the wiring of the ADDRESS and DATA bus. The circuit will function without the keyboard and LCD sections. If the circuit seems to be working (responding to MIDI notes, turning on LED's) but the keyboard and/or LCD do not respond, check the wiring of IC4 and IC6. Check for READ and WRITE (RD and WR) signals from IC1, pins 17 and 18 respectively.

Along with source code and HEX data for the EPROM, sample songs are provided in the zip file on the RE-BBS. (The sample files are also included with any order from the source men-

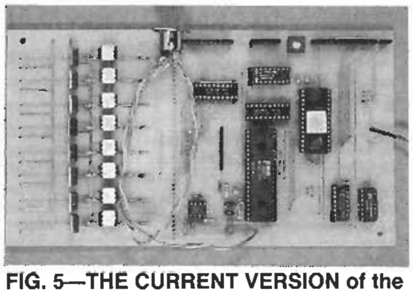


FIG. 5—THE CURRENT VERSION of the PC board, shown here with all parts installed.

tioned in the Parts List.) Three samples are provided in both *.WRK and *.MID formats that will help you get started. The songs were set up using a Roland MT-32, Cakewalk 3.0, and our MIDI Light Controller. The MIDI Light Controller and knowledge of your sequencer's operation will provide you—and your audience—with a great looking show. R-E