Number 5 on your Feedback card

Micro IDer

An easy-to-build automatic Morse code station identifier.

by Steve Look KA9SZW and David Pointer

More and more hams these days are attaching miniature radio transmitters to weather balloons, kites, and rockets. This underscores the need for a very small Morse code identifier necessary for legal operation. A very small identifier would also be useful for compact and portable foxhunt transmitters and beacons.

The typical Morse code identifier uses an EPROM and several other logic chips. This confuguration is fine for applications where size, weight, and power consumption are not considerations. The Micro IDer presented here consists of a maximum of 12 components mounted on a single-sided printed circuit board measuring only 1-5/8" x 5/8". The complete unit weighs less than 1/2 gram. Power requirements are 1-2 mA at 3 to 6 volts. One Duracell DL2032 3V lithium cell will power the IDer for hours. Total cost should not exceed $20.

Theory of Operation

The Micro IDer is based on the Xilinx 1736A serial PROM (U2). This eight-pin IC will store 36,288 bits of data. When a clock signal from the 555 timer (U1) is applied to pin 2, each bit in the memory appears at pin 1 in sequence. This pin is connected to the base of transistor Q2 to drive the keying circuit of a transmitter. Q3 and R3 may be needed on the keying circuit to invert the output if you find the code being sent is inverted. Adding the transistor is cheaper than programming a new PROM.

Q1, R4, and R5 form an inverter between pin 6 and pin 3.

When the last bit of the memory has been clocked out, the PROM generates a logic high at pin 6. The inverter applies a momentary logic low to pin 3. This resets the PROM and the entire memory is read out again.

The timing is provided by a CMOS 555 timer in an astable multivibrator configuration. The value of resistor RA sets the clock speed and is determined by the software that generates the actual ID bit pattern. A standard 555 timer may be used instead of the more expensive CMOS part, but the power consumption goes up by a factor of at least 10, greatly affecting the battery life.

Construction

Mount the two ICs first. We recommend only a high quality machine socket for U2 to allow PROM changes. A spring contact socket may cause reliability problems. Mount RA, R1, R2, and R4 on the bottom of the board next. This is done to save space. All the other components can now be mounted to the top of the board.

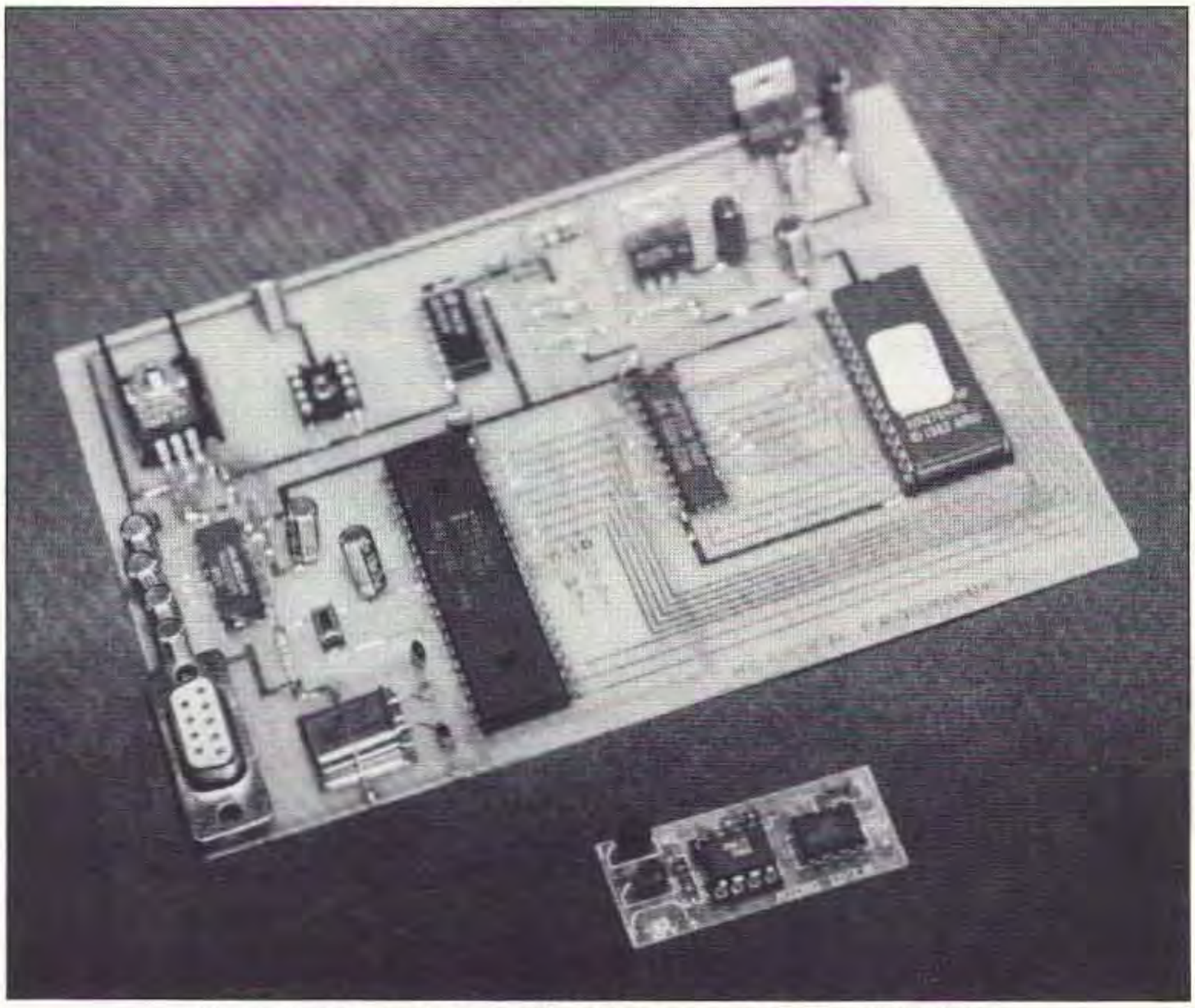


Photo A. Here is the completed Programmer (top) and the IDer.

mkid—A Morse Code Compiler

With this large memory space in the serial PROM available, two programs were written to simplify message generation.

You must first create a file with any text editor (or a word processor in ASCII mode) that contains the text of the Morse code you want to be sent. Be careful about your spelling as the 1736A is a one-time programmable part. All characters are supported, but not the prosigns. Two other characters are included to add a solid tone and silent pause function. A pound sign in your text file represents a solid one-second tone. Place as many of these in a row as you want the tone duration to be. An exclamation point in your text file represents a one-second silence. Place as many in a row as you want the silent period to be.

Text may be entered in upper or lower case as the software converts everything to upper case at compile time. Use a carriage return wherever you wish; they are ignored. When your message looks the way you want it to, save it to disk and exit your text editor.

A very simple example text file may look like:

ka9szw balloon #####

This would generate my (Steve's) call, space, "balloon," space, and then a five-second solid tone. Long periods of a solid tone

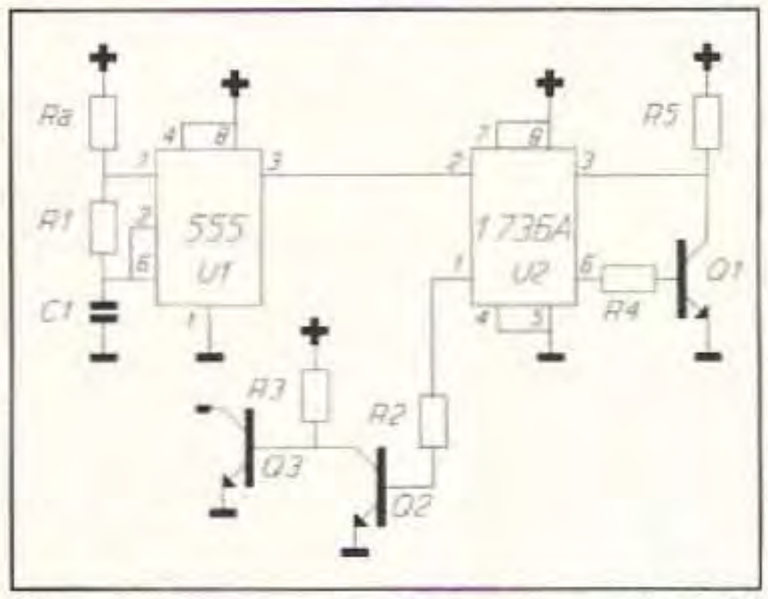


Figure 1. Schematic of the Micro IDer. Q3 and R3 are only used if keying must be inverted.

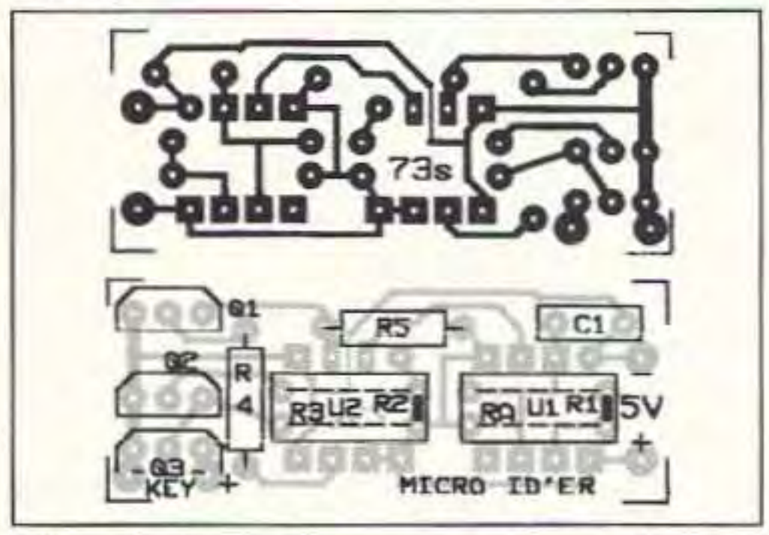


Figure 2. Parts placement outline and circuit board pattern.

1. IC POWER CONNECTIONS

	+5U	GND
U1	40	20
U2	20	10
U3	28	14
U4	16	15
U5	14	7

2. All resistors are 5% 0.25W unless otherwise noted.

3. All capacitors are 20% unless otherwise noted.

4. Heat sink attached to Q1.

5. U8 is socketed.

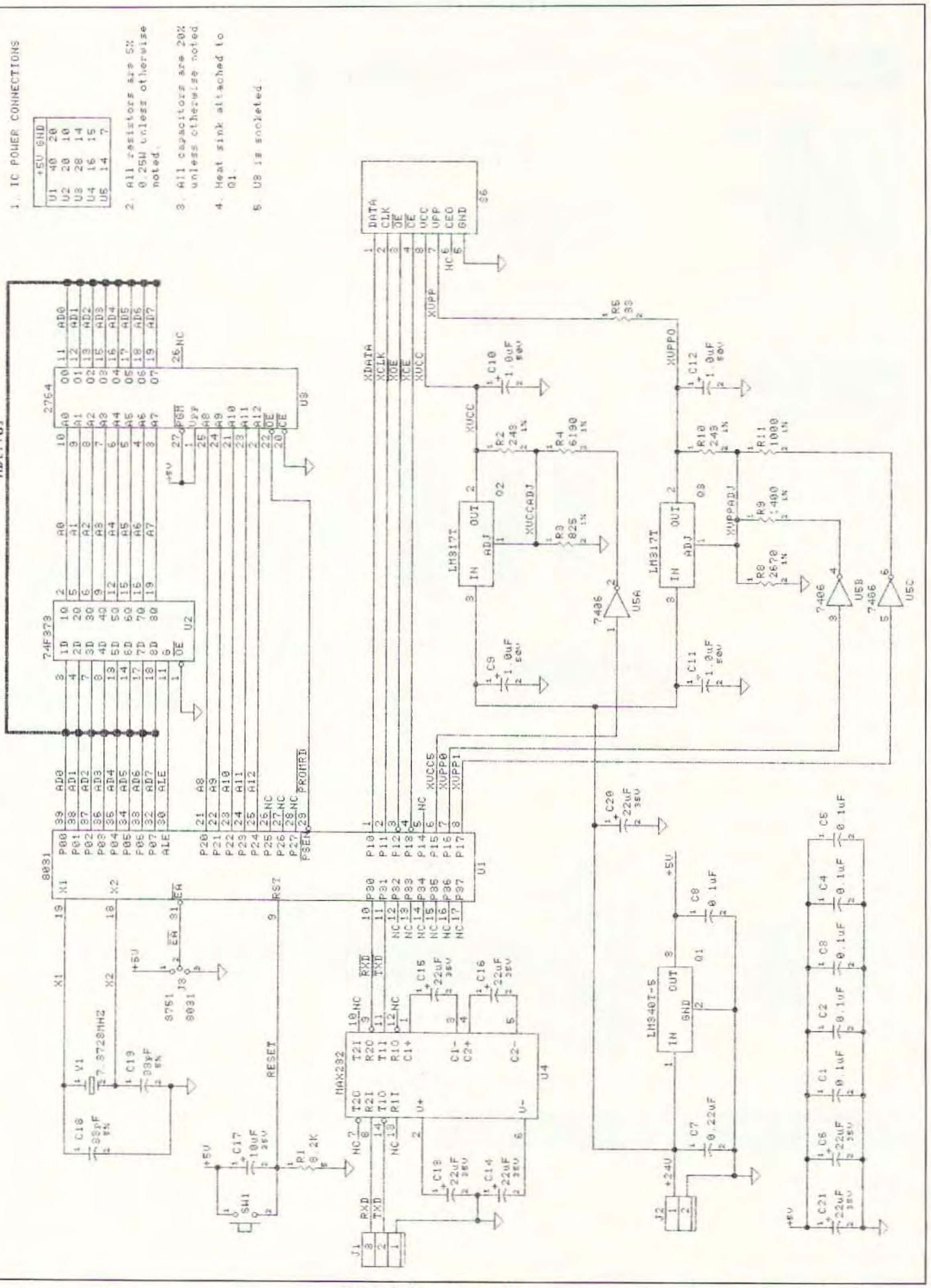
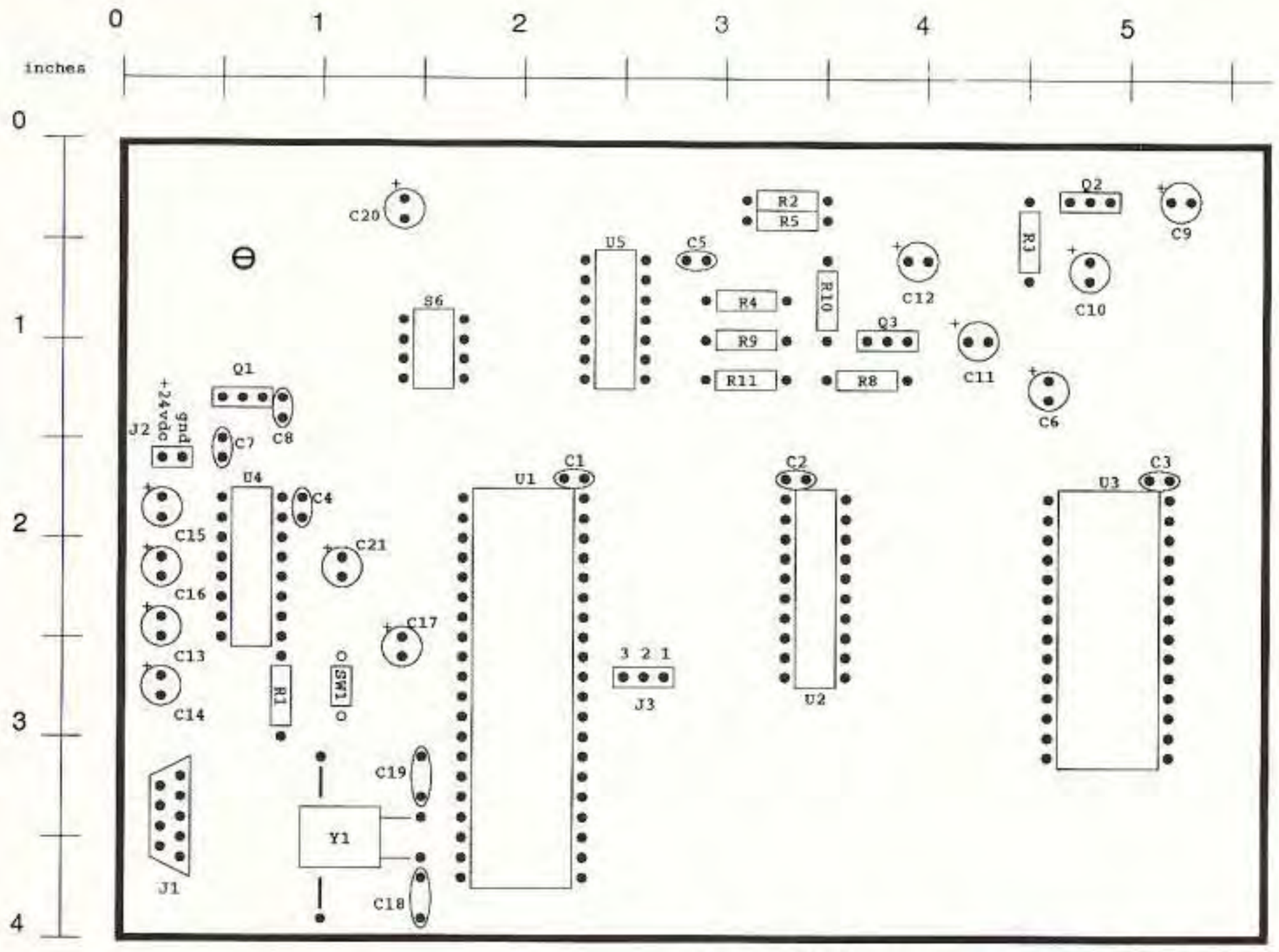


Figure 3. Programmer schematic.



1. Hole sizes (inch dia):
   - ⊖ 0.156 (1 place)
   - ○ 0.052 (2 places)
   - ● 0.040 (213 places)
2. Attach heat sink between board and Q1 using 6-32 x 3/8 machine screw, #6 lockwasher, and 6-32 nut.
3. Lay crystal Y1 flat on copper plane, and solder a bare wire strap over Y1 using the two holes provided.

Figure 4. Programmer parts placement @ 70%.

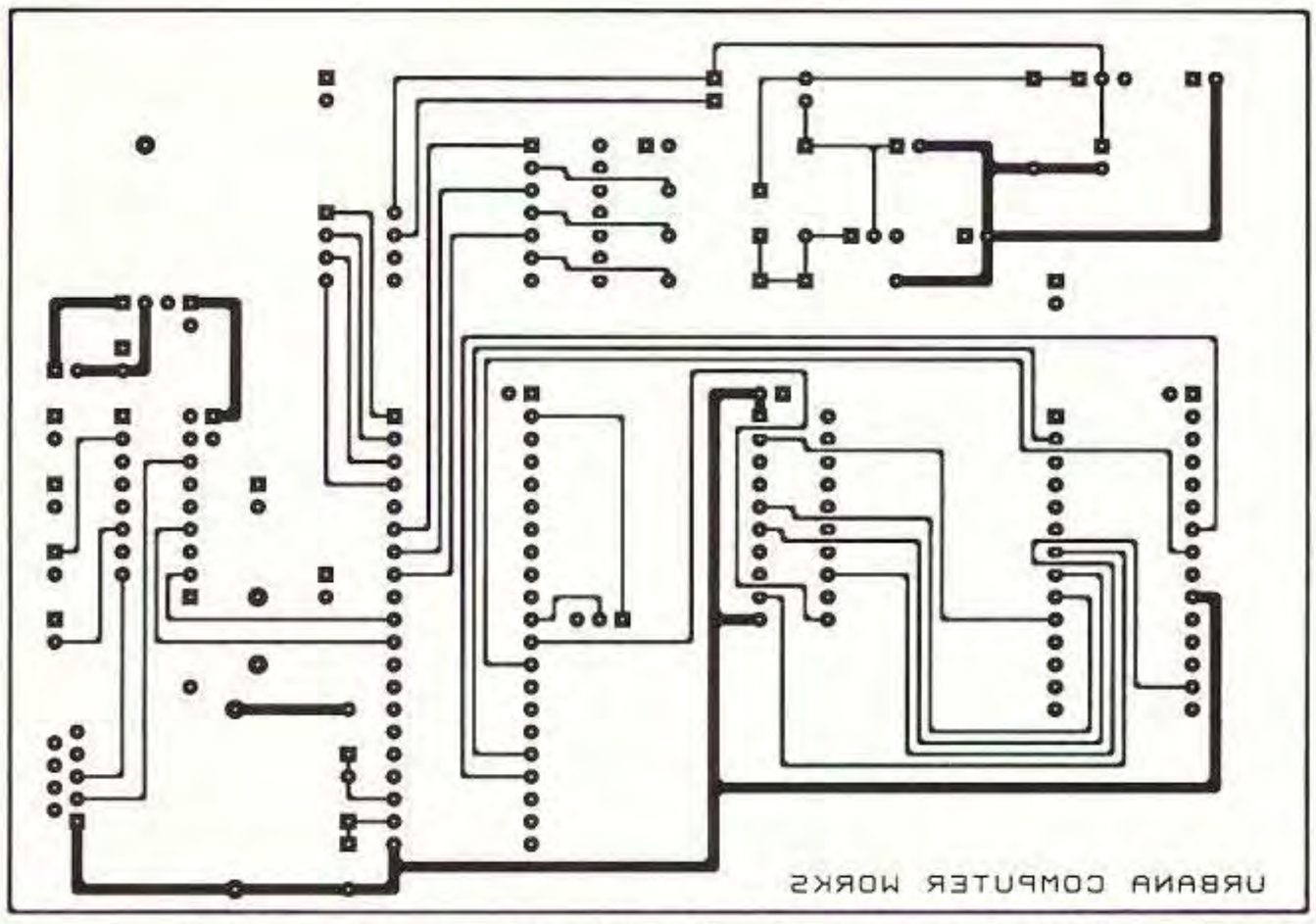


Figure 5. Programmer solder side circuit pattern @ 70%.

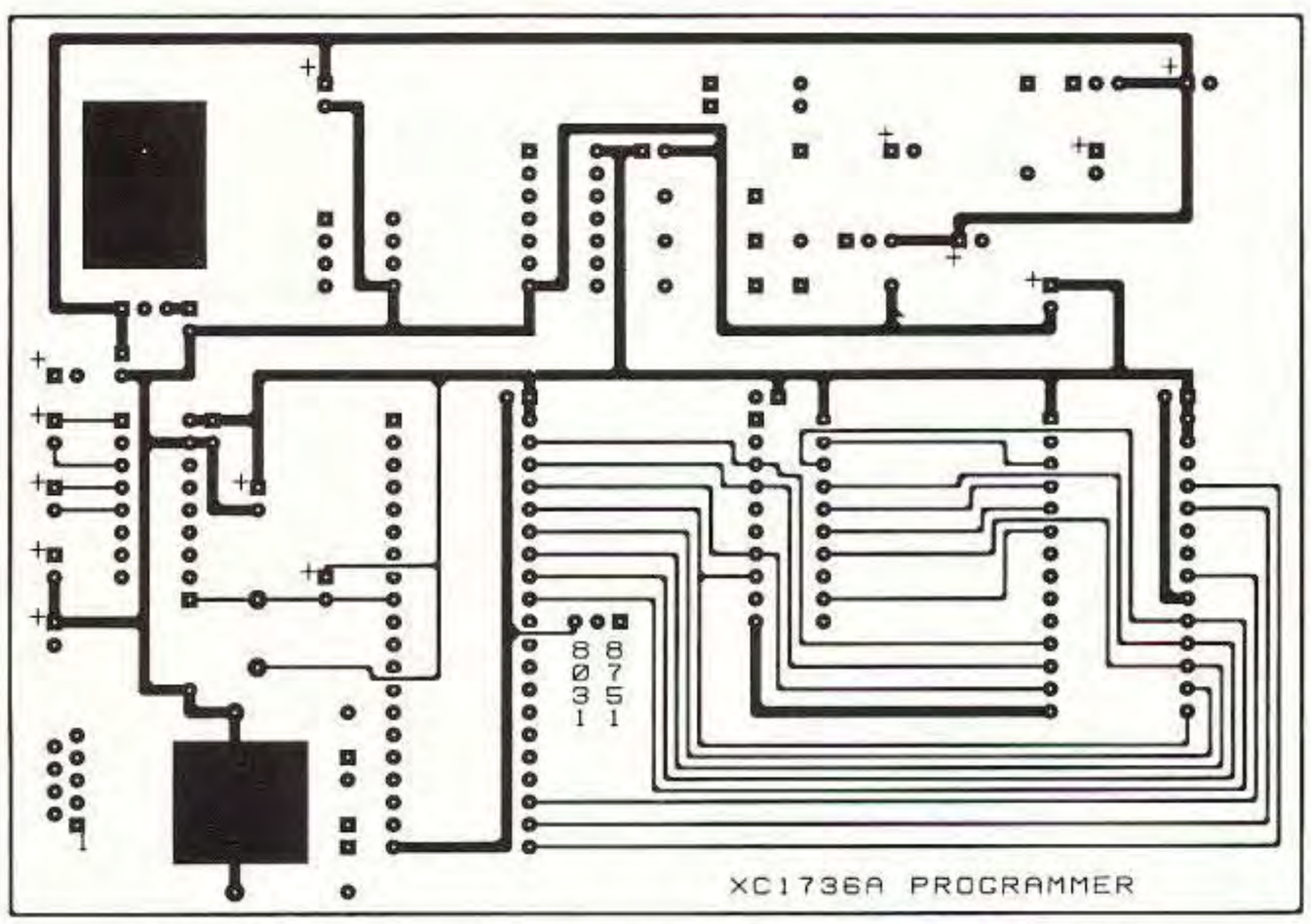


Figure 6. Programmer component side circuit pattern @ 70%.

are helpful to chasers trying to get a fix on a balloon package.

To run the compiler, type:

mkid <text_file>

where <text_file> is the name of the text file you created with the message. The program will display error messages if it cannot find the file. If it has found the file, it will load it while displaying it on the screen so you can check your work one more time. You will be prompted to enter how fast you want the code to be sent. After calculating for a bit you will see a list of available ID delay times. This is how often the message will repeat. Small messages generate large lists of delay times while large messages may offer only a few choices. Enter the number of the delay you wish to use.

After that is done the program opens a file with the same name as the input file, but with a ".jed" extension, and fills it with the keying pattern required for your message.

All the dot and dash timing is done along with key-downs for constant tones. The program will announce when it is done and display some statistics about the ID it just created. The program displays the frequency that the 1736A must be clocked at to get the correct timing, how long the ID will take to send, how long the ID will be silent before restarting, how much of the chip capacity was used, and what the value of the timing resistor (RA) must be.

The file created is in standard JEDEC format and should be accepted by any chip programmer that will handle the Xilinx 1736A, or you can build the companion programmer presented here. Follow the instructions in the manual on how to download a JEDEC file to your particular programmer.

jed2bin—A JEDEC File to Binary File Converter

If you have a programmer that will only accept binary or image files you will also need to use the program "jed2bin." This pro-

gram accepts the ".jed" file and converts it to a binary file. The program prompts you for file names.

All the above programs have been written in generic "C" language to be portable to any computer with a "C" compiler. An executable MSDOS version of each program is available along with the source code.

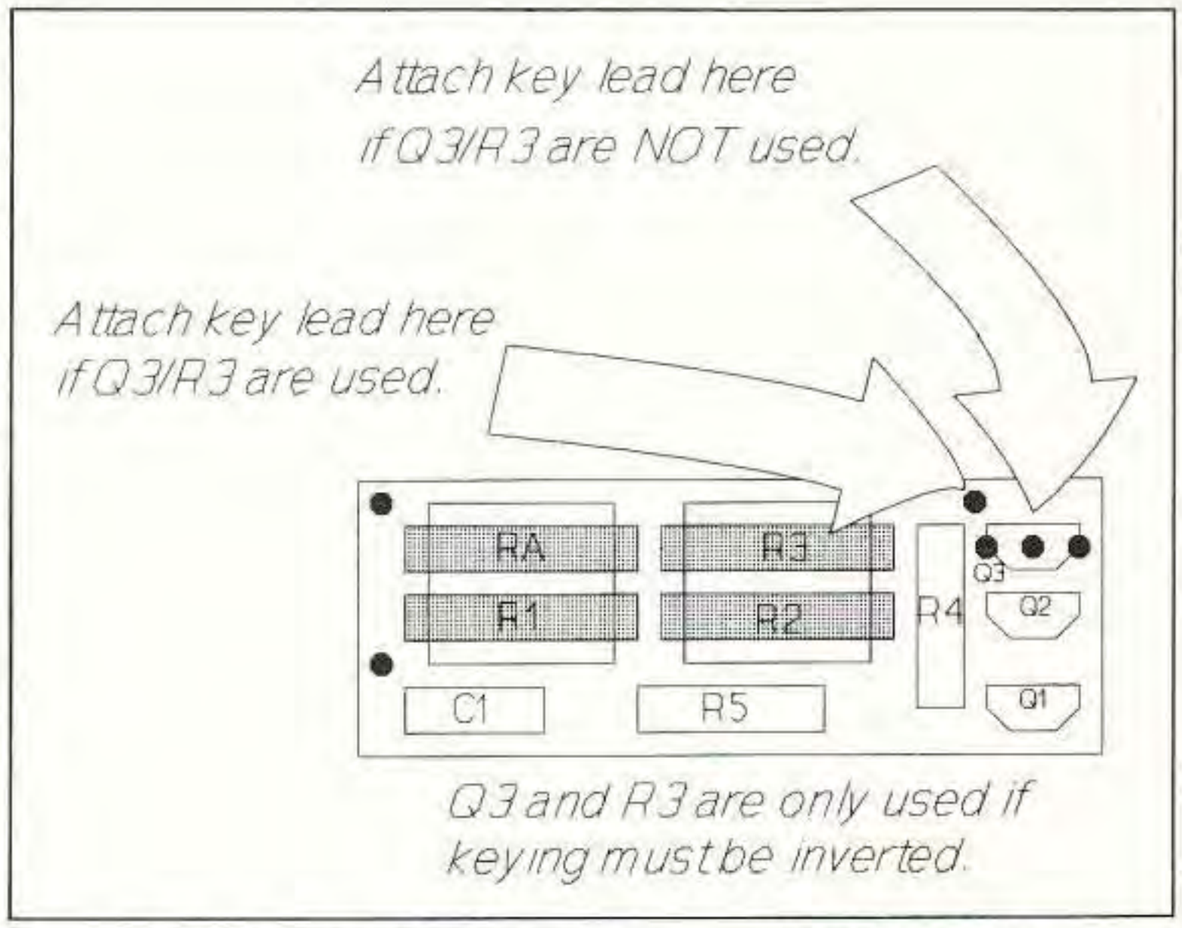


Figure 7. Key wire placement on completed Micro IDer.

The Micro IDer XC1736A PROM Programmer

The Micro IDer XC1736A PROM programmer is an inexpensive alternative for those who want to program an ID into the Micro IDer PROM. Since the least expensive commercial programmer that we know of that can program the XC1736A costs $475, we decided to make this special purpose programmer available to users of the Micro IDer.

The programmer that we developed consists of a board and a host computer program. Communication between the host program and the programmer is through an RS-232 serial port. Power to the board is provided by a 24 VDC wall transformer.

Theory of Operation

The host program "xprog" takes the standard JEDEC file that was produced by the mkid program and sends it to the programmer. An Intel 8031 microcomputer on the programmer board (U1) handles this communication and provides control on the board.

The programmer's 8031 code is contained in a 2764 EPROM (U3). This code has all the programmer's communication routines and the routines that implement the algorithm needed to program the XC1736A. This code is available as an INTEL hex file or already programmed into a 2764 EPROM. The 8751, (which is an 8031 with an EPROM on-chip), may also be used instead of the 8031-latch-2764 combination. Some users may find the 8751 a less expensive alternative to the three-chip combination. A strap on the board (J3) allows for the use of either the 8031 or the 8751.

The XC1736A programming algorithm is complicated. It requires that voltages on two pins be varied between 15V, 6V, 5.5V, and 5V. The programmer controls these voltages by switching various feedback resistors on two LM317 variable voltage regulators. In addition, these voltages must be switched in various combinations before and after serial data is clocked into the device in socket S6. The clock, control, and data lines to the programming socket S6 are connected directly to the 8031 microcomputer.

Q2 can be switched between 5V and 5.5V for the VCC pin of the programming socket. This is controlled by the 8031 microcomputer, which switches R4 in and out of the Q2 resistor network via switching the input of the high voltage open collector inverter U5A.

Q3 can be switched among 5V, 6V, and 15V for the VPP pin of the programming socket. This is also controlled by the 8031, which switches R9 and R11 in and out of the Q3 resistor network using two inverters from the U5 package. R5 provides series damping for VPP, as the overshoot on the VPP pin must never exceed 15.5V.

Construction

The parts placement of the programmer board can be seen in Figure 4. Even though the bare board is two-sided without plated-through holes, it is assembled by soldering all connections that have traces on both sides of the board. The board, which was too complicated to be a single-sided board, was considered too expensive as a double-sided plated-through board. So, we struck a compromise by making sure that all traces could be interconnected by soldering component holes with traces on both sides of a non-plated through hole printed circuit board. This is easily accomplished except in the case of the programming socket S6 and the socket for the EPROM, S3. If the machine screw sockets specified in the Parts List are used, approximately 0.05" of bare metal of the barrel of the socket pin is available for soldering on the component side of the board.

Q1 must have a heat sink attached to it as this device converts the 24 VDC input into 5 VDC, producing a lot of heat in the process. In addition, the crystal Y1 needs to be laid flat and have a bare wire strapped across it. Two holes connected to ground are provided for this purpose.

J2 provides the power connection. A wall transformer that provides unregulated 24 VDC with at least 500 mA of current is re-

XC1736A Prom Programmer Parts List

Reference	Part Description	Manufacturer	Part Number	Digi-Key	Quantity
C1-C5,C8	Cap 0.1µF 63V 20%	Panasonic	ECU-S1J104ZU	P4917	6
C7	Cap 0.22 µF 63V 20%	Panasonic	ECU-S1J224ZU	P4918	1
C9-C12	Cap 1 µF 50V 20%	Panasonic	ECE-A1HU010	P6260	4
C17	Cap 10 µF 35V 20%	Panasonic	ECE-A1VU100	P6248	1
C6	Cap 22 µF 35V 20%	Panasonic	ECE-A1VU220	P6249	7
C13-16,C20, C21 C18,C19	Cap 33 pF 100V 5%	Panasonic	ECC-F2A330JCE	P4450	2
Y1	Xtal	7.3728 MHz	CTS	MP074, CTX074	1
R1	Res 8.2k 5% 1/4W			8.2KQ	1
R5	Res 33 5% 1/4W			33Q	1
R2,R10	Res 243 1% 1/4W			243X	2
R3	Res 825 1% 1/4W			825X	1
R11	Res 1000 1% 1/4W			1.00KX	1
R9	Res 1400 1% 1/4W			1.40KX	1
R8	Res 2670 1% 1/4W			2.67KX	1
R4	Res 6190 1% 1/4W			6.19KX	1
U2	IC 74F373 latch	NSC	74F373PC	74F373PC	1
U3	IC 2764 8Kx8 EPROM	Microchip	27C64-15/J-ND	27C64-15/J-ND	1
U5	IC 7406 hex oc inv	NSC	DM7406N	DM7406N	1
U1	IC 8031 micro	Signetics	SCN8031HCCN40		1
U4	IC MAX232 RS232	Maxim	MAX232CPE		1
Q1	LM340 5V 1.5A reg	NSC	LM340T-5	LM340T-5	1
Q2-Q3	LM317 1.5A adj reg	NSC	LM317T	LM317T	2
S3	Skt 28-pin 0.6"	Mill-Max	110-93-628-41-001	ED3628	1
S6	Skt 8-pin 0.3"	Mill-Max	110-93-308-41-001	ED3308	1
J1	Conn 9-pin D-Sub	Norwesco	09S1	509F-ND	1
SW1	Sw mom push-button	Panasonic	EVQ-QEC04K	P8027S	1
	TO-220 heat sink	AAVID	577202B00000	HS107-ND	1
	6-32 x 3/8 mach screw				1
	6-32 hex nut				1
	#6 ext. tooth lock washer				1

Micro IDer Parts List

Reference	Description	Manufacturer	Part #	Digi-Key	Quantity
U1	IC CMOS timer	NSC	LMC555CN	LMC555CN	1
U2	IC serial PROM	Xilinx	XC1736A		1
S2	Skt., 8-pin, 0.3"	Mill-Max	110-93-308-41-001	ED3308	1
Q1-Q3	Transistor, NPN	NSC	2N3904	2N3904	3
RA	*Value determined by software, see text.				1
R1-R5	Res 10k 5% 1/4W			10KQ	5
C1	Cap 0.1 µF 63V 20%	Panasonic	ECU-S1J104ZU	P4917	1

These products available from the authors at Monticello Micro, 727 West Wilson, Monticello IL 61856.

IDER-AT: Assembled and tested, Micro IDer complete with programmed XC1736A PROM, $20 (please include message to be programmed, the time delay between IDs, and code speed).

IDER-CD: Programmed XC1736A PROM alone, $8 (please include message to be programmed, the time delay between IDs and code speed).

XPROG-AT: Assembled and tested XC1736A programmer, $79.

XPROG-CD: Programmed 2764 EPROM with 8031 code, $10. IDER-DISK: 5.25" 360K diskette containing mkid 'C' program source code, mkid program .EXE executable, jed2bin 'C' program source code, jed2bin program .EXE executable, xprog 'C' program source code, xprog program .EXE executable, IBM PC compatible serial port driver source code, XC1736A programmer 8031 object code (Intel HEX format), $5.

Shipping and handling is included in the above.

We assume that you can acquire the 24 VDC 500 mA wall transformer at a hamfest far cheaper than we could provide it to you. For that matter, you probably already have one stashed in your junk box.

The IDer circuit board alone is available from FAR Circuits, 18N640 Field Court, Dundee IL 60196 for $3 plus $ 1.50 S&H. The XC1736A programmer circuit board alone is also available from FAR Circuits for $11 plus $1.50 S&H.

For all orders, please include your name, address, and phone number. Illinois residents please include 6.25% sales tax.

quired to power the programmer board. The wall transformer leads may be soldered directly into the board. You must take care to make sure that the +24 VDC wire of the transformer connects to pin 1 (marked with a "+") on J2.

xprog—a JEDEC to IDer Programmer Board Communication Program

The host program xprog, which is written in "C", is completely portable and can be compiled on any computer that has a serial port and a "C" compiler. The hardware dependent code for the serial port is written as a separate piece of code so that a person may add a serial port driver appropriate for their particular computer. A driver for IBM PC and compatible COM1 and COM2 ports is provided along with the host program source code. Instructions detailing what is required for other serial port drivers is also provided. The program is invoked by:

xprog <filename>.jed

where <filename> is the name of the JEDEC file produced by mkid. 73

References:

1) *ABEL Design Software User Manual*, September 1990, DATA I/O Corp., Appendix B: JEDEC Standard Number 3A.

2) *Programmable Gate Array Design Handbook*, 1986, Xilinx Inc., pp. 1-50 to 1-60.

3) *Radio Amateurs Handbook*, 1985, Amateur Radio Relay League, pp. 9-8.

Program 1.

```
/* mkid ver 1.0 - Morse code to JEDEC file convertor.
Copyright (c) 1991 Stephen R. Look
This program is available for unlimited non-commercial
distribution. Modifications, bug reports and questions
(SASE please) may be sent to:
Stephen R. Look
727 W Wilson
Monticello, IL 61856
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
struct id
{
int     key[1];
char    morse[25];
```

Program 1 continues

```
};
char bits[20000];
char dits[36288];
/*—— function to exit upon invalid character detection ——*/
void char_error()
{
printf("\nerror - the last character displayed is invalid!");
printf("\n        please correct the message text and retry\n");
exit(-1);
}
/*—— function to return correct unit of time ——*/
void time_frame(t,p)
float t;
char *p;
{
if(t>=3600) sprintf(p,"%.1f hrs",t/3600);
else if(t<3600&&t>=60) sprintf(p,"%.1f mins",t/60);
else sprintf(p,"%.1f secs",t);
}
/*———————— main function ————————*/
main(argc,argv)
int argc;
char *argv[];
{
static  struct id reference[44] = {
{ 32 , "00"},            /* SP */
{ 33 , "0"},             /* !  1 second silence marker */
{ 35 , "1"},             /* #  1 second tone marker */
{ 44 , "11111101010111101110"},/* , */
{ 45 , "11101010101101110"},/* - */
{ 46 , "10111010111010101110"},/* . */
{ 47 , "111010101110101"},       /* / */
{ 48 , "1111011101110111101110"},/* 0 */
{ 49 , "1011101110111101110"},/* 1 */
{ 50 , "10101110111101110"},/* 2 */
{ 51 , "101010111101110"},       /* 3 */
{ 52 , "1010101011110"},         /* 4 */
{ 53 , "10101010101"},           /* 5 */
{ 54 , "111010101010"},          /* 6 */
{ 55 , "111011101010101"},       /* 7 */
{ 56 , "11101110111101010"},/* 8 */
{ 57 , "111011101110111010"},/* 9 */
{ 63 , "10101110111101010"},/* ? */
{ 65 , "101110"},                /* A */
{ 66 , "111010101010"},          /* B */
{ 67 , "11101011101010"},        /* C */
{ 68 , "1110101010"},            /* D */
{ 69 , "10"},                    /* E */
{ 70 , "1010111010"},            /* F */
{ 71 , "111011101010"},          /* G */
{ 72 , "10101010"},              /* H */
{ 73 , "1010"},                  /* I */
{ 74 , "101110111101110"},       /* J */
{ 75 , "111010111010"},          /* K */
{ 76 , "101110101010"},          /* L */
{ 77 , "1110111010"},            /* M */
{ 78 , "111010"},                /* N */
{ 79 , "11101110111010"},        /* O */
{ 80 , "10111011101010"},        /* P */
{ 81 , "1110111010111010"},      /* Q */
{ 82 , "1011101010"},            /* R */
{ 83 , "101010"},                /* S */
{ 84 , "1110"},                  /* T */
{ 85 , "1010111010"},            /* U */
{ 86 , "101010111010"},          /* V */
{ 87 , "101110111010"},          /* W */
{ 88 , "11101010111010"},        /* X */
{ 89 , "1110101110111010"},      /* Y */
{ 90 , "11101110101010"} };      /* Z */
long cntr;
int tone,ch,bitcnt,num,refcntr,mcntr,message[1000];
float wpm,time,clk,expandobits,expandoclk;
FILE *in,*out;
char c[20],*ptr;
ptr=c;
if(argc!=2)
{
printf("Usage: mkid [source_file_name]\n");
exit(-1);
}
in=fopen(argv[1],"r");
if(in==NULL)
{
printf("mkid: %s doesn't exist!\n",argv[1]);
exit(-1);
}
/*——————————clear the screen——————————————*/
for(cntr=0;cntr<=25;cntr++) printf("\n");
/*——————————roll the credits——————————————*/
printf("\nMake ID - Version 1.0");
printf("\nMorse Code to JEDEC compiler");
printf("\ncopywrite (c) 1991 by Stephen R. Look  ka9szw\n\n");
/*—read in the text file to memory and echo it to the screen—*/
printf("\nFile read in:\n");
cntr=0;
while(ch!=EOF)
{
ch=getc(in);
if(ch=='\n') ch=32;
message[cntr]=toupper(ch);
printf("%c",message[cntr]);
if(message[cntr]>=0&message[cntr]<10) char_error();
if(message[cntr]==11) char_error();
if(message[cntr]>12&message[cntr]<26) char_error();
if(message[cntr]>26&message[cntr]<32) char_error();
if(message[cntr]==34) char_error();
if(message[cntr]>35&message[cntr]<44) char_error();
if(message[cntr]==64) char_error();
if(message[cntr]>90) char_error();
cntr++;
}
fclose(in);
/*—get code speed and calculate clock speed per ARRL Handbook-*/
printf("\n\ncode speed in wpm: ");
wpm=atoi(gets(c));
clk=wpm/1.2;
/*—use the lookup table to generate bit patterns————————-*/
for(mcntr=0;message[mcntr]!=EOF;mcntr++)
{
for(refcntr=0;refcntr<=43;refcntr++)
{
if(reference[refcntr].key[0]==message[mcntr]) break;
}
if(message[mcntr]==35)
{
for(tone=0;tone!=(int)clk;tone++) strcat(bits,"1");
if(message[mcntr+1]!=35) strcat(bits,"000");
}
else if(message[mcntr]==33)
{
for(tone=0;tone!=(int)clk;tone++) strcat(bits,"0");
if(message[mcntr+1]!=33) strcat(bits,"000");
}
else
{
strcat(bits,reference[refcntr].morse);
strcat(bits,"00");
}
}
/*—get timing desired and calculate the number of bits needed—*/
bitcnt=0;
bitcnt=strlen(bits);
cntr=1;
ch=1;
while(1)
```

Program 1 continues

```
{
expandobits=(float)cntr*(float)bitcnt;
expandoclk=(float)cntr*clk;
time=(36288.-expandobits)/expandoclk;
if(expandobits>36288)
{
printf("\nChoose ID delay time - ");
printf("<number> or [Enter] to quit->");
gets(c);
num=atoi(c);
if(num<=0) exit(0);
else break;
}
time_frame(time,ptr);
printf("<%ld> %s\t",cntr,c);
if(ch%4==0) printf("\n");
ch++;
if(cntr%88==0)
{
printf("\nChoose ID delay time - ");
printf("<number> or [Enter] for more->");
gets(c);
num=atoi(c);
if(c[0]!=NULL) break;
}
cntr++;
}
/*----------generate JEDEC file----------*/
strcat(argv[1],".jed");
printf("\nCreating file: %s\n",argv[1]);
out=fopen(argv[1],"w");
ch=0;
for(cntr=0;bits[cntr]!=NULL;cntr++)
{
for(mcntr=0;mcntr<num;mcntr++)
{
dits[ch]=bits[cntr];
ch++;
}
}
fprintf(out,"ider JEDEC file produced by Mkid\n");
fprintf(out,"The 555 timer frequency needs to be %.1f Hz\n",(float)num*clk);
fprintf(out,"Resistor RA = %.1f Ohms",(14400000/((float)num*clk))-2000);
fprintf(out,"%c%c\n"\nL0\n",0x0a,0x02);
mcntr=0;
for(cntr=0;dits[cntr]!=NULL;cntr++)
{
fprintf(out,"%c",dits[cntr]);
if(mcntr==63)
{
fprintf(out,"\n");
mcntr=0;
}
else mcntr++;
}
fprintf(out,"*\n%c0000\n",0x03);
fclose(out);
printf("\nFile complete\n");
/*---------calculate timer frequency and device stats--------*/
printf("\nThe 555 timer frequency needs to be %.1f Hz",(float)num*clk);
printf("\nResistor RA = %.1f Ohms",(14400000/((float)num*clk))-2000);
time_frame((float)cntr/((float)num*clk),ptr);
printf("\nID will take %s to send",c);
time_frame((36288-cntr)/(num*clk),ptr);
printf(" with a delay of %s before it repeats\n",c);
printf("and uses %.1f%% of the chip capacity.\n",(float)cntr/36288*100);
exit(0);
}
```

Program 1 ends

Program 2.

```
/* jed2bin v1.0 - JEDEC to binary file convertor for stupid prom programmers.
Copyright (c) 1992 Stephen R. Look
This program is available for unlimited non-commercial
distribution. Modifications, bug reports and questions
(SASE please) may be sent to:
Stephen R. Look
727 W Wilson
Monticello, IL 61856
*/
#include <stdio.h>
/*---------- main function ----------*/
main()
{
FILE *in,*out;
int num,ch,cntr;
unsigned x;
char bucket[80];
char one[20],two[20];
printf("\nEnter input file name: ");
scanf("%s",one);
printf("\nEnter output file name: ");
scanf("%s",two);
in=fopen(one,"r");
out=fopen(two,"wb");
/*----------roll the credits----------*/
printf("\nJEDEC to Binary file converter - Version 1.0");
printf("\n     -a program of limited usefulness-");
printf("\n   copywrite (c) 1992 by Stephen R. Look\n\n");
/*-----read in the text file to memory and echo it to the screen-----*/
printf("\nStripping header text:\n");
for(cntr=0;cntr<=5;cntr++)
{
fscanf(in,"%[^\n]\n",bucket);
printf("%s",bucket);
}
printf("\nStarting conversion:\n");
x=0;
while(1)
{
num=0;
for(cntr=0;cntr<=7;cntr++)
{
ch=getc(in);
if(ch==0x0a) ch=getc(in);
if(ch==0x0d) ch=getc(in);
if(ch==0x2a) break;
num>>=1;
x++;
if(ch=='1') num |= 128;
}
if(ch!=0x2a)
{
putc(num,out);
printf("\r%u ",x);
}
else
{
printf("\nFilling out file space with 0's:\n");
num=0;
while(x<36288)
{
putc(num,out);
printf("\r%u ",x);
x+=8;
}
exit(0);
}
}
```

Program 2 ends