bandpass filters. Finally, if the filters "recognize" the audio-frequency code tones, an audio driver and power amplifier are energized.

Radio pagers generally are assigned to operate in only one of four frequency bands, depending on the type of service the subscriber requires. The low vhf band from 30 to 50 MHz is usually used for on-site coverage of a facility, such as a factory or construction site. For wider coverage, one might choose a pager designed to operate on the high vhf band between 132 and 174 MHz. For even greater coverage, usually obtained through the use of multiple antennas at the call service's facility, a uhf pager that operates between 406 and 420 MHz would prove to be a better choice. Finally, when signal saturation is desired on floors of buildings and in basements in metropolitan areas where there are many tall buildings, a uhf pager that operates in the 420-to-512-MHz range would be required.

Until last year, these were the only bands available for paging services. In mid-1980, however, Motorola announced the availability of a tone-and-voice pager designed to operate at 800 MHz. The new Dimension IV pager from Motorola operates in the 851-to-866-MHz band and is available to only those subscribers who qualify for their own frequencies. This pager is said to be compatible with existing 800-MHz base-station (call-service) equipment and encoders.

Highly efficient integrated circuitry has reduced power demands to where a single AA-size mercury battery will keep a pager operating for almost seven weeks. This eliminates the bulky, heavy nickel-cadmium batteries used in early pagers and the need for daily recharges as well. Higher-efficiency circuits are only a part of the battery long-life story. New circuits such as those used in the Multitone and Motorola pagers automatically switch off the major portions of the pager's circuit in the absence of calls, switching them on again when there is an incoming call.

What Paging Costs. Most radio-call companies offer a choice of two plans for their paging services. Under one plan, a monthly charge is levied and includes rental of the pager, handling of calls, and maintenance. Prices under this plan vary according to range of coverage and the number and types of pagers rented. The general range is from $13.50 to $18 per month per pager.

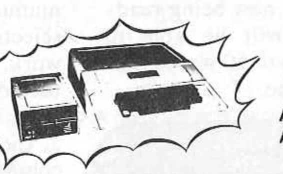
The second plan is designed for the subscriber who wishes to purchase his pagers outright. Cost is about $200 each. In this case, the cost of service including handling of calls and maintenance, ranges from $5 to $8 per month per pager. Again, the exact figure depends on the number and type of pagers being used.

Range of coverage varies from service to service. For example, Beep Communication Systems, Inc. (New York City) offers coverage from Albany and surrounding cities, down through Pennsylvania and parts of Delaware. Radiofone Corp., also in the East, covers about 20,000 square miles and was the first paging-service company to offer coast-to-coast service, connecting New York, Chicago, and Los Angeles via Western Union's Westar satellite. By linking up services in various cities with shared frequencies and exchanging access numbers, a group of paging companies can provide subscribers with a large "umbrella" under which the pager wearer can travel from city to city without loss of service. The most common coverage appears to be 30 to 75 miles.

Considering the modest cost and service, it's small wonder that radio paging has become so popular. The prestigious "beep" has even been known to get one a better table in a restaurant. ◇

EXCITING MAIL ORDER DISCOUNTS

apple® II
16K computer



Disk II with Controller $535 . . . without $429
Nearly Everything for Apple

$949
APPLE II 32K $1049
APPLE II 48K $1149
DOS 3.3 with upgrade kit $54

APPLE II ACCESSORIES

Apple Joystick	47
ABT Numeric Input Keyboard	99
ALF Music Synthesizer	245
Brightpen Lightpen from Softape	32
Clear Cover for Apple Computer	25
CCS Programmable Timer Module	159
Dan Paymer Lower Case Kit (Model I)	59
Dan Paymer Lower Case Kit (Model II)	59
Hayes Micromdem*	335
Extender Board	27
Graphics Input Tablet	649
Introl X-10 Remote Control System	239
Introl X-10 Controller Only	164
M&R Sup-R-Term 80 Column Board	325
Microsoft Z-80 Soft Card	299
Novation Cat Modem	159
Romwriter	149
Speechlink 2000/64 Word Vocabulary	215
SuperTalker Speech Synthesizer	239
Sup-R-Mod RF TV Modulator	25
SVA 8" Disc Controller Card	329
Versa-Writer Digitizer Drawing System	209

WE SHIP FAST!

SOFTWARE

Adventure	27
Apple Bowl	14
Assembler/Dissassembler	69
Applebug Debugger	27
Apple Graph & Plot System	59
Applepost Mailing List System	44
Applesoft Utility Programs by Hayden	27
Applewriter Word Processor	65
Bill Budges Space Game Album	39
Bill Budges Trilogy of Games	29
The Cashier Retail Management System	189
CCA Data Management by Personal Software	84
The Contributed Volumes 1-5 w/manuals	29
Desktop Plan by Personal Software	79
Dow Jones Portfolio Evaluator	44
Forth II by Programma Software	45
Fortran for use with Language System	159
Pascal Language System	419
Sargon II Chess Game-cassette	27
Sargon II Chess Game-disk	35
Shell Games	27
Space Invader-Cassette	18
Space Invader-disk	25
Sub-Logic FS-1 Flight Simulator Cassette	23
Sub-Logic FS-1 Flight Simulator-disk	29
Sybex Apple 80 8080 Simulator	17
Visicalc by Personal Software	119
Business Package	595

AND MUCH MORE . . . JUST CALL!

APPLE COMPUTER INTERFACE CARDS

Asynchronous Serial Interface #7710A	$145
Applesoft II Firmware Card	149
CCS Arithmetic Processor Card #7811B	329
CCS Parallel Interface Card #7720A	155
Centronics Printer Interface Card	179
Apple Clock/Calendar Card by Mountain Hardware	229
Communications Card & DB25 Cable	179
Integer Basic Firmware Card	149
Parallel Printer Interface Card	139
Prototyping Hobby Card	22
Romplus Card with keyboard filter	169
Romplus Card without keyboard filter	159
SSM AIO Serial/Parallel I/O Assembled & Tested	169
Serial Interface Card	139
Symtec Light Pen Card	214

PRINTERS, MONITORS, DISCS

Daisy Wheel Printer!	1795
Anadex DP8000	850
Anadex DP8000AP	850
Centronics without cable	819
Epson TX80-B W/Graphics	799
Epson MX80	645
MPI 88T	650
NEC Spinwriter #5510	2695
Paper Tiger 460G	1394
Silentype with Interface Card	510
Leedex Video 100-12" B&W	129
Sanyo 9" B&W	169
Dyson disks (pkg 10)	50
Memorex (pkg 10)	40
Verbatim (pkg 10)	30

apple computer
IN STOCK, CALL FOR PRICES!

ATARI 800
PERSONAL COMPUTER SYSTEM
$747

- High resolution COLOR Graphics
- 10K Basic in ROM
- RAM expandable to 48K
- Built-in RF TV modulator
- High speed serial I/O port
- 57 key full stroke keyboard

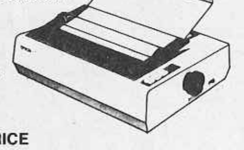
If Atari makes it, we sell it!

Atari 400 Computer	$469
Atari 820 Printer	489
Atari 810 Disk Drive	569
Atari 410 Program Recorder	69
Atari 16K RAM Module	149
Atari 8K RAM Module	99
Atari Basic ROM	45
Atari Visicalc	180
Basketball	30
Video Easel	30
Super Breakout	30
Music Composer	45
Computer Chess	30
Star Raiders	39
3D Tic-Tac-Toe	30
Joystick	19
Assembler/Editor	45

"BEST PRINTER VALUE IN THE WORLD"
Check our prices on THE ULTIMATE IN QUALITY

Epson Printers

APPLE & MOST OTHER COMPUTERS



NEW! MX80, 40, 80, 132 columns
LIST $645
TX 80 col. w/Graphics
LIST $799
Apple Controller & Card & Cable.
LIST $110

CALL FOR OUR PRICE
- Most reliable small printers ever sold!
- Uses standard printed paper.
- Dumps Apple screen graphics right on the paper — with graphics option.

Order desk: 1-800-854-2833 / Information line: 714-579-0330

VISA

TO ORDER: Phone orders invited using Visa, Mastercard, or bank wire transfers. Visa & MC credit card service charge of 2%. Mail orders may send charge card number (include expiration date), cashiers check, money order, or personal check (allow 10 business days for checks to clear). Please include phone number. Include 3% shipping, handling, and insurance ($5.00 minimum) in USA. Shipments within Calif. add 6% sales tax. Foreign orders include 1% handling-shipped freight collect. Foreign orders over $1000 allow 3 weeks extra and include $25 license fee. All equipment is in factory cartons with the manufacturers warranty. Equipment is subject to price change and availability. We ship the same day on most orders. No C.O.D.s accepted. Retail store prices defer from mail order prices.

master charge

MAIL TO: 1251 BROADWAY EL CAJON CA., 92021

AUTHORIZED APPLE SALES & SERVICE

Computer Specialties

DIVISION OF CMI

CIRCLE NO. 11 ON FREE INFORMATION CARD

BINARY OCTAL AND HEXADECIMAL NUMBERS

How to understand and apply the three basic numbering systems in computer programming

BY FRED BLECHMAN

MICROCOMPUTER users often need to understand and apply number systems other than the familiar decimal system. The most popular of these is the hexadecimal (base 16) system, with octal (base 8) and binary (base 2) following close behind.

This article will help clear up some of the mysteries regarding these various systems and show how to convert decimal numbers to any of the bases and vice versa. A BASIC computer program that performs hex/decimal conversions is also given.

The Number Systems. Tables I through IV are designed to help you learn how the familiar decimal system relates to the binary, octal, and hexadecimal systems.

The values associated with the digits of decimal, binary, octal, and hexadecimal systems are given for the first four places in Tables I-IV. In each case, the left-hand column gives the number (or letter) "code" for that system. The other four columns contain the *decimal numbers* that represent each code location. Since decimal numbers are common to all tables, once you learn how to use these tables you can convert from one number system to another using the equivalent decimal number. The hexadecimal system requires more digits than can be expressed by the characters 0,1,2,3,4,5,6,7,8,9, so the first six letters of the alphabet are added. The full set is thus 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. While FFFF may be an odd-looking expression, it is a valid number in the hexadecimal system.

Using these tables takes some explanation, so let's start with Table I, the Decimal Number System. The decimal code column shows the familiar digits 0-9. Now look at the far right (Column I) which also reads vertically from 0 to 9; ten digits in all. Our base, then, is 10. These numbers are actually the numbers 0 to 9 times the zero power of the base 10. Column II shows each of the Column I digits multiplied by the base, 10, raised to the first power. Column III shows each of the Column I digits multiplied by 10^2, or 100; column IV shows each multiplied by 10^3, or 1000.

Suppose you wanted to know the decimal *number* for decimal *code* 6037. This is a 4-digit number, which means we'll use all four columns in the table. The first or most significant digit is 6. Look down the left-hand "Decimal Code" column to 6 and go to the right to Column IV, where you find 6000. Call this the decimal value of Column IV. The next most significant digit is 0. The Column III value for Decimal Code 0 is 0. The next significant digit of 6037 is 3. In Column II, the decimal value for Decimal Code 3 is 30. The last or least significant digit, 7, is seen to equal 7 in Column I for Decimal Code 7. Now simply add the Column I through Column IV values (6000+0+30+7) and you have 6037 (the hard way!).

You can also work backwards—that is, find the decimal *code* for a given decimal *number*. Suppose the decimal number is 2408. Start by finding the *largest* number in the table that is *smaller* than the decimal number 2408. That would be 2000 in Column IV for Decimal Code 2. Therefore, the first decimal digit is 2. Now, subtract 2000 from 2408, leaving a remainder of 408. Again, find the *largest* number in the table *less than* 408. The number is 400, in Column III, for decimal digit 4. Now, the remainder of 8 (after subtracting 400 from 408) *bypasses* Column II and we find the *exact* value, 8, in Column I at Decimal Code 8. This generates *two* decimal code numbers, a 0 for Column II and an 8 for Column I. The result: a decimal code of 2408.

The Binary Number System. Table II shows the Binary Number System, using the same tabular approach as shown for the decimal system. Since the binary code consists of only two digits, 0 and 1, the columns use a base of 2. That is, Column I (far right) is 2 to the zero power, times the digit. Column II is 2 to the first power, Column III is 2 to the second power, and so forth. Binary code 0 is 0 in all columns (since zero times anything is zero).

Suppose you had a 4-bit binary code 1011. Starting with Column IV and adding the values wherever a binary code of 1 appears for that column, you have 8+0+2+1 equals 11 in decimal. Going the other way—that is, starting with a decimal number and determining the binary code—use the largest-number subtraction technique described previously. Suppose you needed the binary code for decimal 14. You'd have a "1" for Column IV. Then, subtracting 8 from 14, you have 6 as a remainder. Column III has a 4, so you have a "1" for Column III, and a remainder of 2 (6−4), which is an *exact* match with Column II. That gives you a binary code of "1" for Column II, and "0" for Column I. The end result is that decimal 14 equals binary 1110.

The Octal Number System. Table III shows the Octal Number System using our familiar format. Since the base number is 8 this time, all the columns are powers of 8. For example, the 256 in Column III is 4 times 8 to the second power (4×8^2), which equals 4 times 64, or 256. The table is used exactly the same way as the previous tables. Octal code 4273 is equal to decimal 2048+128+56+3, or 2235 decimal. Going the other way, 3273 decimal is equal to 6311 *octal* (3273−3072−192−8−1=0).

The advantage of octal is that a 3-bit binary number (such as 110, 101, etc.), can be represented as a single octal digit, since a 3-bit binary number can only range from 0-7. For example, binary 110 would be octal 6; binary 101 would be octal 5. Thus, a long binary number of 12 bits could be broken up into 4 groups of 3 bits each, and represented by 4 octal digits.

The Hexadecimal Number System. Now we arrive at "Destination Hex," where we computer users wanted to go all along. Actually, there are good reasons why hexadecimal numbers are so popular in computing. Most computers use binary 8- or 16-bit "words," 8-bit "bytes" and 4-bit "nybbles" internally. One hexadecimal character can represent a 4-bit nybble, two hex characters an 8-bit byte, and four hex characters a 16-bit word! Thus, hex code provides a reasonable "middle-ground" between the binary code the computer actually processes and the decimal code we humans like to use.

Table IV shows the tabular approach, and a simple BASIC program is also shown. Either will convert from hex to decimal or decimal to hex for the range of 0 to 65535 or 0 to FFFFh. (An h or H is often added to a hexadecimal number to indicate the type of code.)

Look at Table IV, Hex Code column. Since the base for hexadecimal code is 16, we use single-character letters to represent the 10,11,12,13,14 and 15 codes.

The table is used exactly like the previous ones. Thus, 3CB6h is equal to 12288+3072+176+6 or 15542 decimal. Going the other way, 32650 decimal equals 7F8Ah (32650−28672−3840−128−10=0).

BASIC Hex/Decimal Conversion Program. Although the program is self-prompting and requires no programming knowledge to operate, you may be interested in how it works. A fair understanding of TRS-80 Level II BASIC language statements and functions is assumed.

Lines 5 to 40 introduce the program and print the conversion limits and "menu" on the screen followed by a blank line. Line 50 asks which way you want to go—decimal-to-hex or hex-to-decimal. You enter a 1 or 2 and line 60 sends you to either line 100 or line 500. Assume you typed and entered a "1." Then M equals "1" and you continue at line 100. (Note that line 60 could also be: IF M=2 GO TO 500). For simplicity, error-trapping is not used, and if you enter *any* number but "2" you'll be at line 100 which clears the screen and tells you how to return to the "menu" or exit the program. Line 110 asks for your decimal number for variable N. Line 111 traps numbers over 65535, and line 112 checks if N is zero for return to menu.

Line 115 sets the value of A equal to N, and also sets counter variable X to zero. Line 120 starts a "countdown" by subtracting 4096 (which is 16 to the third power) from A. Line 130 looks at the result to see if it is less than zero; if it is, then 4096 is added, the variable Z is set to the value of X and the program branches to a subroutine, lines 1000-1060. We'll get to that later.

If, however, line 130 is bypassed because A is zero or larger, then line 140 adds 1 to the value of X and sends the program back to line 120 for another subtraction of 4096.

What's happening here is that the computer is counting down from the top of Column IV of Table IV. Each time it repeats line 120, it is adding to the hex code by 1 (X increases by 1). Finally, when line 130 finds A is less than zero, the value of X is the hex code for the most significant digit. The subroutine starting at line 1000 then defines the value of this digit.

Suppose you enter the decimal number 32650 for N in line 110. Lines 120 and 140 will subtract 4096 seven times until, on the eighth time, the result is −118. The variable X has been advanced by line 140 seven times so it has a value of 7. Line 130 now sees that A is less than zero, so 4096 is added, making A equal to 3978. Variable Z is set equal to 7 (the present value of X) and we move to line 1000.

Line 1000 sees that the value of Z is less than 10, so it sets string-variable X$ equal to STR$(X), the numerical value of X in string form. This allows X$ to be a number or a letter. Here, X$ is equal to a string value of 7.

Lines 1005-1050 are ignored, since Z is equal to 7, and line 1060 returns the program to the latter part of line 130, where the branching took place. The next statement sets E$ to the value of X$, for later use. Then the program is directed to line 145.

Line 145 sets variable B equal to the last value of A, which is 3978 in our example. Then X is reset to zero. Now lines 150-170 subtract 256, inspect for a value less than zero, and count, in the same manner as lines 120-140. After 16 subtractions, B has a value of −118. Line 160 catches this, adds 256 to bring the value of B up to 138, sets Z equal to 15, and branches to line 1000.

Lines 1000-1040 are ignored, since Z equals 15. However, line 1050 is satisfied and X$ is made equal to the letter F. Line 160 then makes F$ equal to F.

Similary, lines 175-200 determine the third hex code, which in the example turns out to be 8, with the final value of C equal to 10. The subroutine at line 1000 sets X$ equal to a string value of 8, and line 190 makes G$ equal to 8.

Line 210 takes the remainder (C in line 190, with a value of 10 in our example) and lets variable D equal this value. It now sets Z equal to D and again goes to line 1000. Since our example has Z equal to 10, line 1000 is ignored, but line 1005 sets X$ equal to the letter A. Lines 1010-1050 are ignored, line 1060 returns the program to the end of line 210 and H$ is set to a value of A.

In memory we have E$ equal to 7, F$ equal to F, G$ equal to 8 and H$ equal to A. Line 220 skips a line on the screen and prints out 7F8A and line 225 prints a message reminding you to ignore leading zeros. Then line 230 sends the program back to line 110 for another decimal number after printing a blank screen line.

Hex-To-Decimal Conversion. If you had indicated, as a response to line 50, that you wanted hex-to-decimal conversion by typing and entering a "2," the program would branch to line 500. The screen is cleared, two blank lines are printed, followed by the menu/exit message and a blank line. Line 510 prints a prompt asking for the hexadecimal value, V$. If you enter zero, line 511 returns the program to the menu. Suppose you type in 45FD. Line 512 makes sure this is a 4-character string by adding leading zeros if necessary. Line 520 extracts the rightmost character of V$ and makes X$ equal to it. In our example, this would make X$ equal to D. Now the program branches to a subroutine starting at line 2000.

Line 2000 looks at X$ to see if it has a value less than A. Since numbers come before letters in the computer's pre-programmed ROM value sequence, the letter D allows the program to "fall thru" to line 2030. Since X$ *is* equal to D, this program line now makes the variable Z equal to 13. Lines 2040 to 2060 are ignored, and line 2070 returns the program to the branching point, the end of line 520. Variable D is made equal to Z, which is currently 13.

Line 2060 is a simple error-trap to keep you from entering any letters beyond F, since they would be invalid in hex code.

Line 530 looks at the third character in X$, which is an F in our 45FD example. The subroutine sets the value of Z to 15 in line 2050 and the end of line 530 makes C equal to 15 times 16 or 240.

Line 550 finds the second character in X$ to be 5. The subroutine catches this value in line 2000 as being less than A, and sets Z to the *decimal* value of 5. (VAL(X$) converts a string value to a numerical value.)

The end of line 550 makes B equal to 5 times 256, or 1280. Similarly, lines 560 and 1000 take the first character (4) of X$ and makes A equal to 4 times 4096, or 16384. Notice that A,B,C and D are all now numerical values, and can be added together. Line 600 does this and prints the result. In our example, 45FD equals 16384 plus 1280 plus 240 plus 13, for a result of 17917 decimal. Line 620 goes to line 510 for another hex number. Line 999 is just good form to identify the beginning of a subroutine, but could be omitted.

Other Number Conversion Programs. While the program given here is simple and can be entered from your

TABLE I—THE DECIMAL NUMBER SYSTEM

Decimal Code	IV	III	II	I
	$10^3=1000$	$10^2=100$	$10^1=10$	$10^0=1$
0	0	0	0	0
1	1000	100	10	1
2	2000	200	20	2
3	3000	300	30	3
4	4000	400	40	4
5	5000	500	50	5
6	6000	600	60	6
7	7000	700	70	7
8	8000	800	80	8
9	9000	900	90	9

TABLE II—THE BINARY NUMBER SYSTEM

Binary Code	IV	III	II	I
	$2^3=8$	$2^2=4$	$2^1=2$	$2^0=1$
0	0	0	0	0
1	8	4	2	1

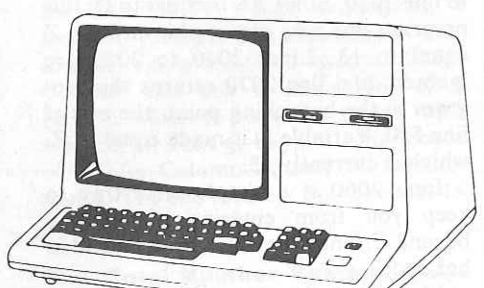
TABLE III—THE OCTAL NUMBER SYSTEM

Octal Code	IV	III	II	I
	$8^3=512$	$8^2=64$	$8^1=8$	$8^0=1$
0	0	0	0	0
1	512	64	8	1
2	1024	128	16	2
3	1536	192	24	3
4	2048	256	32	4
5	2560	320	40	5
6	3072	384	48	6
7	3584	448	56	7

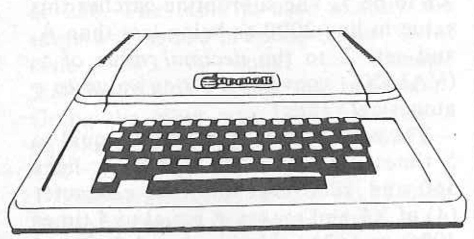
TABLE IV—THE HEXADECIMAL NUMBER SYSTEM

Hexadecimal Code	IV	III	II	I
	$16^3=4096$	$16^2=256$	$16^1=16$	$16^0=1$
0	0	0	0	0
1	4096	256	16	1
2	8192	512	32	2
3	12288	768	48	3
4	16384	1024	64	4
5	20480	1280	80	5
6	24576	1536	96	6
7	28672	1792	112	7
8	32768	2048	128	8
9	36864	2304	144	9
A	40960	2560	160	10
B	45056	2816	176	11
C	49152	3072	192	12
D	53248	3328	208	13
E	57344	3584	224	14
F	61440	3840	240	15

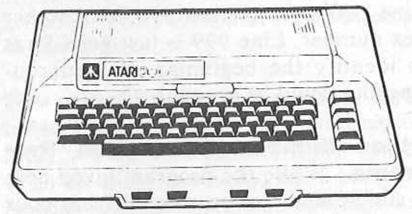
TRS-80™ MODEL III
48K $1039



APPLE - II +
48K $1119



ATARI 800
48K $1069



PRINTERS

MICROLINE-80	$499.00
MICROLINE-82	$669.00
MICROLINE-83	$999.00
EPSON MX-70	$449.00
EPSON MX-80	$519.00
CENTRONICS 737	$769.00
VISTA V300	$1795.00

TSE-HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

TERMS: Prices and specifications are subject to change. HARDSIDE accepts VISA & MASTERCARD. Certified checks and Money Orders; Personal checks accepted (takes 3 weeks to clear). HARDSIDE pays all shipping charges (within 48 states) on all PREPAID orders OVER $100.00. On all orders over $100 a $2.50 handling charge must be added. COD orders accepted (orders over $250 require 25% deposit) there is a $5.00 handling charge. UPS, Blue Label, Air Freight available at extra cost.

CIRCLE NO. 27 ON FREE INFORMATION CARD

NUMBER CONVERTING PROGRAM

```
5 REM * COPYRIGHT FRED BLECHMAN 1980 *
10 CLS
20  PRINT:PRINT:PRINT:PRINT"                NUMBER CONVERTER PROGRAM"
25 PRINT"          (MAXIMUMS: 65535 DECIMAL, FFFF HEX)"
30 PRINT:PRINT"(1) DECIMAL TO HEX CONVERSION
40 PRINT"(2) HEX TO DECIMAL CONVERSION
50 PRINT:INPUT"WHICH DO YOU WANT, 1 OR 2?";M
60 ON M GOTO100,500
100 CLS:PRINT:PRINT:PRINT"ENTER 0 FOR MENU, BREAK TO EXIT....":PRINT
110 INPUT"WHAT IS THE DECIMAL NUMBER (65535 MAX.)";N
111 IF N>65535 PRINT"ABOVE 65535 LIMIT! TRY AGAIN...":GOTO110
112 IF N=0 GOTO10
115 A=N:X=0
120 A=A-4096
130 IF A<0 THEN A=A+4096:Z=X:GOSUB1000:E$=X$:GOTO145
140 X=X+1:GOTO120
145 B=A:X=0
150 B=B-256
160 IF B<0 THEN B=B+256:Z=X:GOSUB1000:F$=X$:GOTO175
170 X=X+1:GOTO150
175 C=B:X=0
180 C=C-16
190 IF C<0 THEN C=C+16:Z=X:GOSUB1000:G$=X$:GOTO210
200 X=X+1:GOTO180
210 D=C:Z=D:GOSUB1000:H$=X$
220 PRINT:PRINT"THE HEXADECIMAL VALUE IS: ";E$;F$;G$;H$
225 PRINT"(IGNORE LEADING ZEROES....)"
230 PRINT:GOTO110
500 CLS:PRINT:PRINT:PRINT"ENTER 0 FOR MENU, BREAK TO EXIT...":PRINT
510 INPUT"WHAT IS THE HEXADECIMAL VALUE (FFFF MAX.)";V$
511 IF V$="0" GOTO10
512 V$=RIGHT$("0000"+V$,4)
520 X$=RIGHT$(V$,1):GOSUB2000:D=Z
530 X$=MID$(V$,3,1):GOSUB2000:C=Z*16
550 X$=MID$(V$,2,1):GOSUB2000:B=Z*256
560 X$=LEFT$(V$,1):GOSUB2000:A=Z*4096
600 PRINT:PRINT"THE DECIMAL NUMBER IS";A+B+C+D
620 PRINT:GOTO510
999 END
1000 IF Z<10 THEN X$=STR$(Z)
1005 IF Z=10 THEN X$=" A"
1010 IF Z=11 THEN X$=" B"
1020 IF Z=12 THEN X$=" C"
1030 IF Z=13 THEN X$=" D"
1040 IF Z=14 THEN X$=" E"
1050 IF Z=15 THEN X$=" F"
1060 RETURN
2000 IF X$<"A" THEN Z=VAL(X$)
2005 IF X$="A" THEN Z=10
2010 IF X$="B" THEN Z=11
2020 IF X$="C" THEN Z=12
2030 IF X$="D" THEN Z=13
2040 IF X$="E" THEN Z=14
2050 IF X$="F" THEN Z=15
2060 IF X$>"F" PRINT"ERROR!! NO LETTER GREATER THAN F! TRY AGAIN...":GOTO 510
2070 RETURN
```

keyboard in 15 to 20 minutes and stored on cassette or disk, it is limited to hex/decimal conversions and to only four hex characters. Several other cassette programs with more flexibility can be obtained from the following sources:

(1) "BASECONV/BAS" converts any number with a base between 2 and 16. No known limit to number size. Excellent, fast, easy to use. $3.95, postpaid, if payment with order. International Data Services, 340 West 55th St., New York, NY 10019. Phone: 212-757-8046.

(2) "Number/Base Converter" will add, subtract or convert numbers of any base between 2 and 16. No known limit to number size. Very flexible. $6.95, postpaid. Demi-Software, P.O. Box 570, Lynbrook, NY 11563.

(3) "Programmer's Converter" has three programs: Base Calculator converts numbers with bases 2 through 16 and performs calculations, including fractions; Hexadecimal/Decimal Training not only performs conversions but includes a teaching/testing program; Number Base Conversions converts decimal, binary, octal and hexadecimal numbers from zero to FFFF, and displays all four values simultaneously. $9.95 plus $1 shipping. Instant Software, Peterborough, NH 03458.

(4) "HEX-DEC Converter" provides hexadecimal-to-decimal or decimal-to-hexadecimal conversions "in memory" up to 65535 decimal or FFFF hex. It can be used during the writing of a BASIC program by using SHIFT-D to convert from decimal or SHIFT-H to convert from hex. The answer appears on the screen and the program returns to BASIC. Very handy. Uses less than 256 bytes of reserved upper memory. $6.95 postpaid. J. Lindsly, 8106 Quailwood Ct., West Chester, OH 45069.

For $2 postpaid, you can receive a BASIC *listing only* (no cassette) of a "number Conversion Program" that converts decimal, hex, octal and binary numbers in the range of zero to FFFF. Dean R. Zimmerman, 444 North Grove Drive, Alpine, UT 84003. ◇

BY JAMES BARBARELLO AND EDWIN IRIYE

Inexpensive, accurate instrument measures inductance from 1 microhenry to 1 henry as well as capacitance

BUILD THE

Reactance Measuring Set

NOW you can measure the inductance of coils and loudspeaker windings without resorting to expensive laboratory instruments. The Reactance Measuring Set (RMS) presented here will measure inductance from 1 microhenry to 1 henry, using any multimeter as a readout device. Furthermore, capacitance from 1 picofarad to 1 microfarad can be determined.

Accurate, stable, and easy to build, the RMS project uses a measurement technique based on the relationships between currents flowing through and voltages appearing across reactive components. The resulting measurements are not influenced by any effective or internal resistances of the components under test. Moreover, the RMS can be aligned without using a precision reference standard. As a bonus, it can function as a crystal-controlled frequency standard. No batteries are needed, thanks to the presence of an internal, line-powered supply.

Measuring Reactors. The voltage drop across a pure inductance is directly proportional to the rate at which the magnitude of the current flowing through it changes with time. Mathematically, this is expressed by the differential equation: $v = L\, di/dt$. If a current that has a constant rate of change flows through the inductor, the voltage drop across it will be constant. Similarly, if the waveform of the current that flows through the inductor is a triangle, the resulting voltage is a square wave.

The current that flows through a pure capacitance is directly proportional to the time rate of change of the voltage across it ($i = C\, dv/dt$). If a voltage with a constant rate of change is applied across the capacitance, a current of constant magnitude flows through it. Similarly, if the waveform of the voltage that is applied across the capacitor is a triangle, a square–wave current flows through it.

Now let's look at a block diagram of the RMS (Fig. 1). A triangle-wave voltage source is the heart of the measurement circuit. It drives both a voltage-dependent current source and a buffer/voltage-to-current converter stage. The former generates a triangle-wave current which is applied to an inductance whose value is to be determined (L_X). If a value of capacitance is to be measured, a triangle-wave voltage is applied across