Don Lancaster's

Tech Musings

May, 1995

Acoustic cancellations
Chip adaptor resources
PIC micro fundamentals
Fast & easy SMD removal
Essential hacker nutrients

Several helpline callers asked about acoustic cancellation. Schemes to null out existing noises by generating new ones precisely out of phase.

Well, your bottom line is: *It don't work*. Except in very special and very restricted circumstances. There is one incredible amount of misinformation out there on this topic.

If you have a highly *confined* area which is very *much* smaller than your audio wavelengths and *when* all of your acoustics are *fully controllable* and *precisely known* in that area, then *sometimes* a rather *modest* reduction can be achieved. Mostly to *very low* frequencies. Otherwise, forget it.

For instance, it is feasible to use small headphones to pick up limited cancellation *inside* of the headphones themselves. But a total cancellation in an open room is absurd.

It is feasible to somewhat reduce air conditioning noise within a long duct. But completely getting rid of all furnace noise is absurd.

It is feasible to modestly reduce the noise inside a special "active" car muffler. But getting rid of all engine noise is totally absurd.

I'll try to work up some more tech info on this as we go along.

Even cancelling your fundamental, you'd be unlikely to get rid of all the harmonics. Unless the waveforms are *precisely* matched and you have very good transducers. So, your *perceived* noise reduction will usually be even worse than your actual one.

PIC Chips

Since they clearly have become *the* hacker component of the decade, we might take a closer look at a PIC chip and see what all of the excitement is about. PIC refers to a series of quite speedy, low power, and very low cost microcontrollers.

Microchip Technology is the prime source for these devices.

PIC's are *much* easier to use than a 555 timer. So, there is *absolutely no excuse whatsoever* to continue using any outdated bits and pieces design. Unless you purposely want to waste time and money.

PIC's are radically different than older micros. And are much better for most low end uses. So, let us think of these instead as a *universal custom integrated circuit*. One that is cheap and instantly available.

Typical PIC chips contain internal EPROM memory. You program them by connecting a simple cable to any host computer and operating suitable software and interface hardware.

Once it has been programmed, the PIC remembers your instructions and operates just as any other application specific integrated circuit.

Certain PIC's can get erased. Done by shining uv light through a package window. Others can be programmed only once. These are the same chip in a cheaper opaque package.

Normally, you'll develop using the erasable versions. You then ship your smaller volume products with the one time programmable devices. And, for really big time, even cheaper factory preprogrammed chips are sold.

The popular PIC16C54 is shown in figure one. This chip is available in several 18-pin packages of different sizes but similar pinouts. As with any ic, there's a supply pin and a ground pin. Supply voltage can range over a +3 to +6 volt dc range.

The supply current is typically one milliampere per clocking MHz. Four mils at four Megs. Dropping down to microamps in a sleep mode.

A crystal gets hung on two pins to form a system clock. You can run up to 20 MHz at five volts. Dropping to 4 MHz at three volts. The special -LP version is also available, intended for 32 kHz use at a supply voltage as low as 2.5. This micropower gem needs only 15 *microamps* of current.

The obvious choice for most aps is a color tv crystal at 3.58 MHz. You also have options of using an external clock, a ceramic resonator, or even a resistor and capacitor network.

Some ceramic resonators now are sold with built-in capacitors.

PIC instructions run at *one-fourth* the clock frequency. A 20 MHz clock

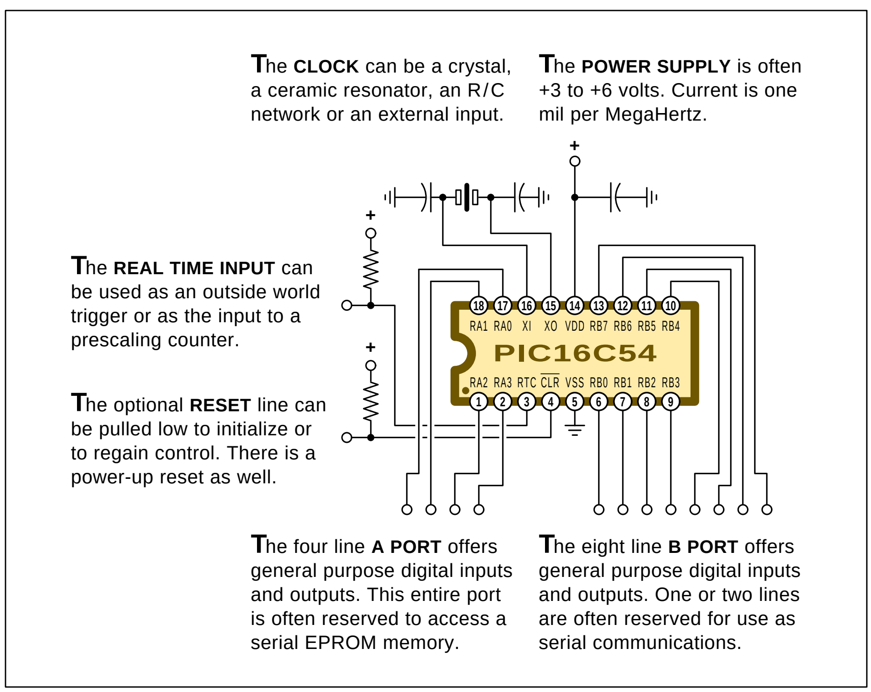
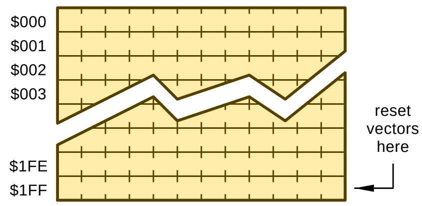


Fig. 1 – THE PIC MICROCONTROLLER is well on its way to becoming the chip of the decade. Here is how to connect one.

The PROGRAM MEMORY consists of one or more 512 byte banks of 12-bit words. Regardless of whether it is actually ROM, EPROM, or OTP EPROM, the program memory is read-only at run time.

Reset jumps you to the highest location in program memory. Which often holds a jump to your actual program start. Unless told otherwise, program steps will be executed in increasing sequential order.

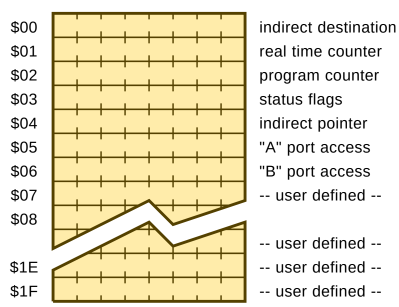
A limited amount of read-only data can be placed in the program memory. Data transfer can be done using the RETLW opcode...



The SCRATCHPAD RAM consists of 32 or more 8-bit bytes of read-write memory.

The first seven locations are dedicated special purpose registers handling the functions shown. The remaining locations are general purpose registers that are yours for any use you want.

If an opcode calls for register $00, an indirect access to the register number stashed in the FSR file select register gets done instead...



There is no direct addressing provision for **EXTERNAL MEMORY**. Instead, one or more port lines are used to access a serial EEPROM storage device.

Here is a typical four-wire external memory lashup. It makes use of a sneaky "logic power" stunt to dramatically cut the long term power consumption...

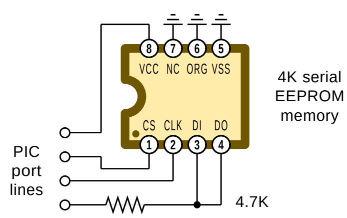


Fig. 2 – PIC MEMORY MANAGEMENT. With the "Harvard" architecture, program and data are pretty much kept apart.

executes 200 nsec instructions.

So far, so good. We have a chip receiving power and clocking. There are twelve I/O input-output pins. You can define any of them as an input or an output. On the fly, even.

Now, all this PIC chip really does is accept the ones and zeros on your input I/O lines. It then generates new ones and zeros for your output lines. But that's all that *any* microprocessor is ever able to do. Your selection of *which* ones and zeros get output will be determined by your *program*.

This leaves us with two pins. An optional active low reset pin. Which gets you started off on the right foot. Or synchronizes to outside events. Or lets you regain lost control.

There is also an automatic internal reset. Provided that you apply your supply voltage *quickly*.

The final pin has several uses. It can wake up a PIC when in its sleep mode. Or this might get used as an external event counter.

PIC Architecture

The internal PIC arrangement is radically different than older micros. The PIC uses a *Harvard* architecture where instructions are typically held separate from data. This lets you use fewer and longer instructions to pick up more speed.

The PIC is a RISC microcontroller. Instead of providing scads of fancy operating modes, there are only a few rather simple commands.

But these are much faster and way more intuitive than the commands on older CISC computers.

Figure two shows us PIC memory management. There is one program memory of 512 bytes. Unlike typical micros, the program instructions are *twelve* bytes wide. Which lets you get by using *single* bytes for each and every instruction.

A program *executes* by starting at some point in program memory and getting an *instruction* or an *opcode*. This opcode then handles some task. Your program then usually moves up to the next location and gets another instruction. The new instruction does something useful. This process goes on forever and ever.

Things do get interesting when you *interfere* with the sequential stepping through your program. An instruction may *conditionally* tell you to *skip* the next instruction. Which gives you a *branch*, letting your program do two different things.

Or an instruction might tell you to *jump* to a different location. This is one method to loop or repeat.

You can also trick a PIC's *program counter* to go somewhere else. This is called a *calculated jump*.

Finally, there are times where you might want to move to some special code area to do something, and then pick up exactly where you've left off. This is called a *subroutine*.

Subroutine access is by way of a *jump to subroutine*. The final opcode in any subroutine usually has to be a *return from subroutine*.

Subroutines can *shorten* code since they can be reused several places in a program. Subroutines 2neaten code if they are logically arranged.

Separating all the big lumps from the little lumps and crumbs.

When you use a subroutine, your PIC has to remember where "back" is. A special *stack* stashes your return address. A *stack pointer* remembers where you are in the stack.

In a PIC, subroutines can be *nested* two deep. Thus a program can call a subroutine which might call another subroutine. But no deeper.

All of the instructions in a PIC are normally locked in at programming time. They are not usually alterable during run time.

This PIC has a small RAM memory of 32 bytes. They can be grouped into *dedicated* and *general purpose* bytes.

These are called *registers*.

The seven dedicated registers will always do one specific task. Such as serving as a "W" accumulator, as a program counter, an I/O data port, a status register, a timer control, and an indirect pointer. Those remaining 26 general purpose registers are yours to do anything you like with. Typically for pointers, counters, addresses, and intermediate results.

There is a provision for both *direct* and *indirect* addressing modes. The direct addressing always goes where you tell it. Indirect addressing goes to a *calculated* location. Addressing RAM location $00 will instead go to the register whose value is stashed in an indirect pointer.

Outside of these few working RAM registers, there is no large read-write memory area in a PIC. Such as you'd want for storage of data or ASCII text or musical notes or whatever. There is also no means to *directly* address *any* external memory.

The way the PIC handles external data storage is to send out or receive *sequential* information. Routed by way of one or more I/O pins.

The usual external memory is a serial EEPROM. Available to 128K bits and beyond. And durable to ten million cycles.

As with any serial system, it takes a while to completely read or write a memory byte. But the PIC clock can be quite fast. Besides, all instructions execute quite quickly. Thus, a serial memory should be fast enough for most real world uses.

Instructions that alter your **PROGRAM FLOW**...

(RESET) – Moves you to the highest location in memory. A **GOTO** usually resides here, jumping you to your real program start.

GOTO – Moves you unconditionally to the location in your program addressed by the instruction's nine lower bits.

CALL – Jumps you to a subroutine with the intent of returning. Subroutine address is set by the instruction's eight lower bits.

RETLW – Returns from subroutine to location in stack. Also loads the accumulator with the instruction's eight lower bits.

BTFSC – Tests a specified bit in a specified register and skips the next instruction if that bit is a zero.

BTFSS – Tests a specified bit in a specified register and skips the next instruction if that bit is a one.

DECFZ – Decrements a specified register and skips the next instruction if the register contents become a zero. "d" bit picks destination.

INCFZ – Increments a specified register and skips the next instruction if the register contents become a one. "d" bit picks destination.

(ANY) – Any other instruction that alters the program counter register will move you to a calculated location in your program.

Instruction that **DOES NOTHING**...

NOP – Goes on to the next instruction. Used for debugging, time delay, or to reserve room for future options.

Instructions which will **CHANGE REGISTERS**...

COMF – Complements the contents of a selected register, changing all ones into zeros and vice versa. "d" bit sets destination.

DECF – Will subtract one from the contents of a selected register. Decrementing $00 produces an $FF. Also see **DECFZ**.

INCF – Adds one to the contents of a selected register. Incrementing $FF produces an $00. Also see **INCFZ**.

BCF – Clears selected bit in selected register to zero.

BSF – Sets selected bit in selected register to one.

RLF – Rotates the bits in a selected register one to the left, going through carry. "d" bit sets register or accumulator destination.

RRF – Rotates the bits in a selected register one to the right, going through carry. "d" bit sets register or accumulator destination.

SWAPF – Exchanges upper and lower 4-bit nibbles of selected register. The "d" bit sets register or accumulator destination.

Fig. 3A – THE PIC INSTRUCTION SET. All are single byte instructions. Most of them execute in a single clock cycle...

The serial EEPROM memory prices start at a dollar. Some are designed from the ground up for PIC interface. Others are easily adapted.

The usual way of having a PIC talk with another PIC or a host micro or whatever is by means of serial comm. One I/O pin can be set aside for each comm channel.

At first glance, PIC resources may seem appallingly limited. But, with creative programming, they can end up far more than what you'll need for a surprisingly diverse variety of low end apps. Several months ago, we saw how any PIC can generate high quality sinewaves using a mere *six* instructions. A PIC environment does lend itself to creative hacks.

There are also fancier PIC devices available that have more RAM and more program memory.

By far your simplest way around perceived PIC limitations is to *use lots of them*. Since they are so simple and cheap, it often pays to use one PIC for your serial comm, a second for your video interface, a third for data management, or a fourth for a

Instructions that are able to **MOVE DATA**...

CLRF – Clears a specified register to all zeros. If an internal "d" bit in the instruction is set, also clears the accumulator to zero.

CLRW – Clears the accumulator to all zeros.

MOVLW – Fills the accumulator with an immediate constant value set by the instruction's eight lower bits.

MOVF – Moves a copy of selected register contents into the accumulator.

MOVWF – Moves a copy of accumulator contents into a selected register.

Instructions that do **ARITHMETIC**...

ADDWF – Adds the contents of the accumulator to the selected register. The "d" bit selects register or accumulator destination.

SUBWF – Subtracts the contents of the accumulator from the selected register by 2's complement arithmetic. "d" bit destination.

Instructions that perform **LOGIC**...

ANDWF – Bit-by-bit AND's the accumulator against a selected register. The "d" bit selects register or accumulator destination.

IORWF – Bit-by-bit OR's the accumulator against a selected register. The "d" bit selects register or accumulator destination.

XORWF – Bit-by-bit XOR's the accumulator against a selected register. The "d" bit selects register or accumulator destination.

ANDLW – Bit-by-bit AND's the accumulator against an immediate mask set by the lower eight instruction bits. Result to accumulator.

IORLW – Bit-by-bit OR's the accumulator against an immediate mask set by the lower eight instruction bits. Result to accumulator.

XORLW – Bit-by-bit XOR's the accumulator against an immediate mask set by the lower eight instruction bits. Result to accumulator.

Instructions that handle **INTERNAL CONTROL**...

TRIS – Accumulator pattern teaches port lines whether they are inputs or outputs. Port selected by bits internal to the command.

SLEEP – Puts chip in low power mode. Wakeup by way of reset, watchdog timer, or external real time input.

CLRWDT – Resets the watchdog timer to zero. Also resets prescaler and status bits TO and PD.

OPTION – Accumulator pattern teaches option register prescaler ratio, real time clock edge, and counter clock source.

Fig. 3B – THE PIC INSTRUCTION SET, continued.

keyboard or LCD display panel.

Distributed processing at its best.

Chances are, after your multi-chip system is up and working that you'll see just how to combine everything anyway. But your multi-chip design time will be far less.

The PIC Instruction Set

Your key secret to understanding a microcomputer is to carefully study the *instruction* set. You then use *each* instruction in as many different ways as you possibly can.

An instruction is just a command that does something. Something can be a *move*, an *add*, a *test* or a *change*. That is all. By creatively combining many instructions, you can upgrade moves, adds, tests, and changes into just about any computer task.

There's only 33 PIC opcodes. They are amazingly powerful and easy to use. Unlike other micros, all opcodes need only one byte. Most of them can execute in a one clock cycle.

Figure three summarizes the PIC instruction set.

Let's start with the commands that can alter your *program flow*. Each successively higher opcode normally should get executed in order. Unless you interfere with the program flow. Such interference, of course, is what microcomputing is all about. And where the fun begins.

A GOTO unconditionally moves you somewhere else.

The CALL command moves you to a subroutine. You'll do this with the intent of returning *just past* where you left off. You usually get out of a subroutine by using RETLW.

RETLW also offers an ultra sneaky use. On return, the eight lower bits of this instruction get loaded into your accumulator. This can let you store read-only *data* in the program side of your PIC memory!

Four testing instructions are able to *conditionally* skip an instruction. These can let you *branch* to different points in your program. BTFSC tests some bit in some register and skips if that bit is a zero. BTFSS tests some bit in some register and skips if that bit is a one. DECFZ decrements any register and skips on any zero result. INCFZ increments some register and skips on a zero result.

Finally, you do have the option to move to a *calculated* location in your program. Done by storing an address in your program counter.

That no operation NOP wastes an instruction. To make room for a later feature, to provide some exact time delay, or as a debugging hook.

So much for commands that alter program flow. Now, let's check into commands that *move* data.

You might clear any register to all zeros by using a CLRF. CLRW clears the accumulator or "W" register. To fill W with a constant, use MOVLW.

To move *from* any register *to* the accumulator, MOVF. To move *from* the accumulator *to* a register, you use MOVWF instead.

There are some commands that can *change* the contents of any register. COMF complements, changing each one to a zero and vice versa.

DECF decrements any register, by subtracting a one. Decrementing any $00 underflows to $FF.

Similarly, INCF can increment any register, adding one to its contents. Incrementing any register at $FF will cause it to overflow to $00.

BCF clears selected register bits to zero. A BSF sets selected bits to one. RLF rotates the bits in any register to the left, going through the carry bit in the status register in the process. RRF rotates bits to the right, also going through the carry bit. Finally, SWAPF *interchanges* upper and lower four bit nibbles in any register.

Next, let's look at commands that take something from somewhere, do something with it, and then move it somewhere else. Your two biggies, of course, are ADDWF which adds the accumulator contents to a register.

And SUBWF for subtraction. Using two's complement arithmetic.

Very handily, these are both *dual mode* instructions. You have a choice of putting your answer back into your accumulator *or* source register.

The *logic* instructions are also dual mode. IORWF will bit-by-bit OR the acumulator against any register. OR is used to force *ones* into a word.

ANDWF will AND the accumulator against a register. AND logic forces *zeros* into a word.

You can use XORWF to bit-by-bit XOR. Another name for exclusive-OR logic is "one but not both". XOR gets used to force *changes*.

You can also perform logic with an immediate value worked against the accumulator. Using ANDLW, IORLW, or XORLW.

Which gets us down to the dregs. SLEEP puts the chip in a power down mode. It stays there until your choice of (A) an external reset event, (B) an internal watchdog timer overflowing, or (C) by using the RTCC pin as an external hardware reset. Power use is far lower in the sleep mode.

The TRIS command can be used to teach the port bit lines whether they are inputs or outputs. Bits internal to this opcode decide whether you are teaching your four port "A" lines, or the eight port "B" lines.

The watchdog timer gets cleared with a CLRWDT command. And last but not least, your OPTION command presets how your watchdog timer will behave. It lets you pick an internal clock or an external clock tripping on either selected edge. A divider can be placed *before* or *after* your timer.

With ratios of 1 through 256.

Among other uses, your watchdog lets you "wake up" your PIC every 18 milliseconds up to every 2.5 seconds. This extends battery life for "check it every now and then" uses.

A *status register* works along with your opcodes. Various bits keep track of zero results, byte and word carries or borrows, power, and timeout.

Three remaining bits are yours for any use you like. These make handy program flags.

PIC programming can get done by routing instructions to your I/O lines and suitably controlling the reset and RTCC pins. The use of a commercial programmer is a must.

We saw a list of programmers and PIC support services last month. As before, you start with the *Microchip Data Book* and the *PIC Applications Manual* from *Microchip Technology*. Then a BASIC Stamp from *Parallax* and the *Scott Edwards* tools.

Lots of additional PIC support can be found on the *Pick a peck of PIC's* library shelf of *www.tinaja.com*

CHIP ADAPTOR RESOURCES

Advanced Interconnections
5 Energy Way
West Warwick RI 02893
(401) 823-5000

Aries Electronics
PO Box 130
Frenchtown NJ 08825
(908) 966-6841

EDI Corporation
PO Box 366
Patterson CA 95363
(209) 892-3270

Emulation Technology
2344 Walsh Avenue Bldg F
Santa Clara CA 95051
(408) 982-0660

Ironwood Electronics
PO Box 221151
St Paul MN 55121
(612) 431-7025

ITT Pomona
1500 E Nonth Street
Pomona CA 91769
(909) 469-2900

Keystone Electronics Corp
31-07 20th Rd
Astoria NY 11105
(718) 956-8900

McKenzie Technology
910 Page Avenue
Fremont CA 94538
(510) 651-2700

Meritec
1359 W Jackson Street
Painesville OH 44077
(216) 354-3148

Mill-Max
190 Pine Hollow Road
Oyster Bay NY 11771
(516) 922-6000

Vector Electronic Co
12460 Gladstone Ave
Sylmar CA 91342
(818) 365-9661

Vero
1000 Sherman Ave
Hamden CT 06514
(203) 288-8001

NEED HELP?

Phone or write all your US Tech Musings questions to:

Don Lancaster
Synergetics
Box 809-EN
Thatcher, AZ, 85552
(520) 428-4073

US email: don@tinaja.com
Web page: www.tinaja.com

Chip Adaptors

Those tiny new surface mount ic packages do tend to reduce the need for plate thru holes on circuit boards. But these sure are small and hard to work with. Especially for your initial design and debug.

Chip Adaptors are a workaround. These are small socket-plug setups that step the tiny pins up to older and larger standards where they are more easy to deal with. The good news is that adaptors are readily available.

The bad is that they may cost more than the chip does. A ten dollar horse and a forty dollar saddle.

At any rate, I've gathered together several of the main players offering these test and debugging adaptors as this month's resource sidebar.

SMD Removal

Removing tiny multi-pin beasties from a circuit board can end up a real hassle. But there is a stunning new solution. One I really like because it turns engineering on its ear.

Every once in a while, it pays to go and try to do the *exact opposite* of what everybody else is up to.

Ferinstance, when any metallurgist designs an alloy, they almost always try to *maximize* its strength. After all,

NAMES AND NUMBERS

Armour Products
PO Box 128
Wyckoff NJ 07481
(201) 847-0404

ARRL Handbook
225 Main St
Newington CT 06111
(203) 666-1541

AST Servo Systems
115 Main Rd Box 97
Montville NJ 07045
(201) 335-1007

Books Americana
PO Box 2326
Florence AL 35630
(205) 757-9966

ChipQuik
3 Second Street
Framingham MA 01701
(800) 836-CHIP

Cryptosystems Journal
485 Middle Holland Rd
Holland PA 18966
(215) 579-9888

Scott Edwards Electronics
964 Cactus Wren Lane
Sierra Vista AZ 85635
(520) 459-4802

Feed Point
NTMS % Wes Atchison
Rt 4 Box 565
Sanger TX 76266

Filtercrest
100 Carol Place
Moonachie NJ 07074
(201) 807-0809

GEnie
401 N Washington St
Rockville MD 20850
(800) 638-9636

Hand Papermaking
PO Box 77027
Washington DC 20013
(301) 587-3635

Microchip Technology
2355 W Chandler Blvd
Chandler AZ 85224
(602) 963-7373

Mo Hotta Mo Betta
PO Box 4136
San Luis Obispo CA 93403
(800) 462-3220

NEXT Generation
1350 Old Bayshore Hwy #210
Burlingame CA 94010
(415) 696-1661

Parallax
3805 Atherton Rd, #102
Rocklin CA 95765
(916) 624-8333

Radar Sales
5485 Pineview Lane
Plymouth MN 55442
(612) 557-6654

Small Parts
PO Box 4650
Miami Lakes FL 33014
(800) 220-4242

Synergetics
Box 809
Thatcher AZ 85552
(520) 428-4073

Telecom Library
12 West 21st St
New York, NY 10010
(800) LIBRARY

Vaughn Duplication Services
4141 E Raymond Ste G
Phoenix AZ 85040
(602) 437-5344

what possible use would a *minimum* strength alloy be?

Just this: Replace all the solder on your chips with a *minimum strength* alloy. One which is so *utterly wimpy* you can pop chips off a board!

The system is called the *ChipQuik SMD* Removal Kit. Each kit is good for eight to ten multi-pin chips. The product costs $13 in quantity. A free video is offered.

Use is amazingly simple: You first apply liquid flux. Then you'll melt a special Chip Quick alloy *into* all your existing solder joints.

The alloy interacts with the solder and produces a *zero strength* joint. You then pop the chip off the board using a dental pick.

Finally, you use desoldering braid to clean up your board.

I'm not sure of the alloy formula, but I suspect indium may be involved somewhere along the way.

New Tech Lit

Everybody knows about the *ARRL Handbook for the Radio Amateur*. Just reissued in its new and expanded *seventy-second* edition! Needless to say, this book is an essential text for *all* of you technical types. Especially for beginner fundamentals.

But lesser known are the dozens of wireless, high frequency, television, microwave, and antenna publications from the same source. They do have a free catalog available.

Books on computer telephony are covered in depth by *Telecom Books*.

Toni Patti has just issued Volume III of his *CryptoSystems Journal* on amateur cryptography and on related items such as fractals and chaos. $45 including PC software.

Next Generation is a unique video gaming mag. Well done reviews on CD ROM, Sega, Nintendo, Jaguar, Arcade, and on-line systems. $29 per year. *Feed Point* is a ham microwave newsletter. And *Hand Papermaking* is an interesting craft pub.

A free slide chart on international television standards is provided by *Vaughn Duplication Services*.

Free industrial foam samples are offered by *Filtercrest*. Glass etching supplies are available from *Armour Products*. Sports radars (both new and recycled police units) are sold by *Radar Sales*. Heavy iron is offered by *AST Servo Systems* in their new and free catalog. Pricey, though.

Small Parts has just released their brand new free catalog #16. Which is *the* place to go for robotics or nearly anything else your hardware store has never even heard of.

Essential hacker nutrients are now stocked in depth by the *Mo Hotta Mo Betta* people. Their free catalog is a must. Uh, better use *extreme* caution when trying *Scotch Bonnets*. These are *strictly* for professional use only. Make sure you have your necessary state Haz-mat permits.

The *Collector's Guide to Personal Computers* is a Tom Haddock book on collecting personal computers. It gives product histories, tech details, and current market values. $15 from *Books Americana*.

Speaking of which, I do still have a few classic Apples left at prices far below the values listed in Haddock's book. Along with bunches of obscure cards. Even impossible-to-get Integer BASIC ones. Call for a list.

I've also once again expanded my *Book-on-demand publishing kit* with lots of new and updated info. These days, you can easily produce superb quality books or technical info. And do so cheaply and quickly. This kit has all of the startup info needed.

Additional support on BOD can be found on *www.tinaja.com*Let's hear from you. ❖