An Example INIT

```
/*
 * An example INIT
 * This is a simple example INIT.  It installs a patch that
 * logs use of _SetTrapAddress to a file on disk.  This INIT must be compiled
 * with the Sysload bit set, so it will be loaded into the System heap.
 * Set Project type to: Code Resource, file type 'INIT', creator '????',
 * name "SetTrapText INIT", resource type 'INIT', ID = 1, Attrs = 50.
 * Caution:  The log file created by this program can grow to great sizes under
 * MultiFinder.
 * Another Caution:  Use care when developing INITs.  A small bug can trash your
 * System or disk with surprising ease.
 */

// Assumes inclusion of <MacHeaders>
#include <Packages.h>
#include <Folders.h>
#include <SetUpA4.h>

#define outfile      ((StringPtr)"\pSetTrapText log")

#define SetTrapAddressTrap 0xA047

void installme(void);
void uninstallme(void);
Boolean OpenFile(unsigned char *);

void __SetTrapAddress(void);
void newSetTrapAddress(void);

void logTrap(void);
void log(short trapNum, long trapAddr, short trapType);
StringPtr GetCodeName(void);
void printHeader(void);
StringPtr NumToHex(unsigned short);
void print(StringPtr message);

void JmpInstr(void);        /* asm label to call original trap's code */
void callMine(void);        /* asm label to 'remove' our patch code */

/*
 * Globals.
 */
short refNum;               /* File Mgr refNum for the open log file. */

void main(void)
{
    Ptr                 myINITPtr;
    register Handle     myINITHandle;
    KeyMap              keys;
    register Boolean installed = FALSE;

    asm {
        move.l a0, myINITPtr
    }
```

```
	RememberA0();
	SetUpA4();
	GetKeys(&keys);

	/* do nothing if the mouse button or shift key are down */

	if (!Button() && !(1 & keys[1])) {
		myINITHandle = RecoverHandle(myINITPtr);
		DetachResource(myINITHandle);
		if (OpenFile(outfile)) {
			installme();
			installed = TRUE;
		}
	}

	if (installed)
		/* Call ShowINIT(), to show "good" icon */ ;
	else {
		if (refNum)
			FSClose(refNum);
		/* Call ShowInit(), to show icon with x thru it */
	}

	RestoreA4();
}

/*
 * Open a text file in the System Folder
 * This routine calls FindFolder().  This is safe under both System 6 &
 * 7 because of glue provided in THINK C and MPW.
 */
Boolean OpenFile(unsigned char * file)
{
	short foundVRefNum;
	short wdRefNum;
	long foundDirID;
	register OSErr err;

	err = FindFolder (kOnSystemDisk, kSystemFolderType, kDontCreateFolder,
			&foundVRefNum, &foundDirID);
	if (!err) {
		err = OpenWD (foundVRefNum, foundDirID, 0, &wdRefNum);
		if (!err) {
			err = Create(file, wdRefNum, 'KAHL', 'TEXT');
			if (err == noErr || err == dupFNErr) {
				if (FSOpen(file, wdRefNum, &refNum) == noErr) {
					SetFPos(refNum, fsFromLEOF, nil);
								/* append to file */
					printHeader();
					return true;
				}
			}
			CloseWD (wdRefNum);
		}
	}
	return false;
```

```
}

/*
 * printHeader -- write a little preamble about when the log begins
 */
void printHeader(void)
{
	long		now;
	Str255		string;

	print((StringPtr)"\p--------SetTrap Log Starting at:");
	GetDateTime(&now);
	IUDateString(now, shortDate, string);
	print(string);
	print((StringPtr)"\p, ");
	IUTimeString(now, false, string);
	print(string);
	print((StringPtr)"\p--------\r");
}

/*
 * Install our head patch by modifing our code.
 */
void installme(void)
{
	register long oldSetTrapAddress;

	oldSetTrapAddress = NGetTrapAddress(SetTrapAddressTrap, OSTrap);
	NSetTrapAddress((long)__SetTrapAddress, SetTrapAddressTrap, OSTrap);
	asm {
		lea		JmpInstr, a1	;	; get address of jmp 0xf0f0f0ff
		add.l	#2, a1			; skip jmp instruction
		move.l oldSetTrapAddress, (a1)	; put address of old code in
							; there
	}
}

#define longNOP	0x4E714E71

/*
 * Deinstall our head patch by writing over the BSR with NOPs.
 */
void uninstallme(void)
{
	asm {
		move.l a0, -(sp)			; probably not necessary
		lea		callMine, a0
		move.l #longNOP, (a0)
		movea.l	(sp)+, a0
	}
	/*
	 * Set refNum to zero, in case we're uninstalling during a series of print()s.
	 */
	refNum = 0;
}
```

```
/*
 * This is the new SetTrapAddress trap entry point
 */
void __SetTrapAddress()
{
	asm {
extern callMine:
		bsr		newSetTrapAddress	; two word instruction
extern JmpInstr:
		jmp		0xF0F0F0FF		; force long jump address
	}
}

static void newSetTrapAddress(void)
{
	asm { movem.l a0/a1/d0-d2, -(sp) }/* save nontrashable regs on stack */
	SetUpA4();
	logTrap();				/* this is where the real stuff is done */
	RestoreA4();
	asm { movem.l (sp)+, a0/a1/d0-d2 }
}

void logTrap(void)
{
	register short trapNum, trapType;
	register long trapAddr;

	asm {
		move.w		d0, trapNum
		move.w		d1, trapType
		move.l a0, trapAddr
	}
	if (refNum)
		log(trapNum, trapAddr, trapType);
	else
		uninstallme();
}

#define TOOLBIT (1L << 10)
#define NSETBIT	(1L << 9)

#define NSETTRAP(trapType)	(NSETBIT & trapType)
#define TOOLTRAP(trapType)	(TOOLBIT & trapType)

void log(short trapNum, register long trapAddr, register short trapType)
{
	print((StringPtr)"\pName: ");
	print(GetCodeName());			/* pascal string */
	print((StringPtr)"\p,\tTrap: 0x");
	print(NumToHex(trapNum));		/* print trap number */

	print((StringPtr)"\p,\tAddr: 0x");
	print(NumToHex((trapAddr >> 16) & 0xffff));	/* top word of address */
	print(NumToHex(trapAddr & 0xffff));		/* bottom word of address */
	print((StringPtr)"\p,\tType: ");
```

```
	if (NSETTRAP(trapType)) {		/* is it an NSetTrap call? */
		if (TOOLTRAP(trapType))			/* is it a ToolBox call? */
			print((StringPtr)"\pTool");
		else					/* must be an OS call... */
			print((StringPtr)"\pOS");
	}
	else					/* old-style SetTrap call */
		print((StringPtr)"\p?");
	print((StringPtr)"\p\r");		/* send <cr> */
}

/*
 * GetCodeName -- try to get the name of the current open code resource.  This is
 * useful for watching INITs, since they don't use CurApName.  It may give misleading
 * results, though.
 */
StringPtr GetCodeName(void)
{
	static unsigned char fName[32];
	FCBPBRec paramBlock;

	paramBlock.ioCompletion = 0;
	paramBlock.ioVRefNum = 0;
	paramBlock.ioFCBIndx = 0;
	paramBlock.ioRefNum = CurResFile();
	paramBlock.ioNamePtr = fName;
	if (PBGetFCBInfo(&paramBlock, false) != noErr)
		return (StringPtr)"\p?";
	return fName;
}

/*
 * Send a pstring to our file.
 */
void print(register StringPtr message)
{
	long count = *message;

	if (refNum)				/* refNum of zero means don't write */
		if (FSWrite(refNum, &count, (message + 1)) != noErr)
			uninstallme();			/* on disk error, de-install self */
}

/*
 * Convert a 16 bit integer to a hex pstring.
 */
StringPtr NumToHex(register unsigned short n)
{
	static unsigned char conv[] = "0123456789ABCDEF";
	static unsigned char s[] = "\pXXXX";
	register short i;

	for (i = 0; i < 4; ++i)
		s[4 - i] = conv[(n >> (i * 4)) & 0xf];
	return s;
}
```