```
/*
** TINY-C INTERPRETER HEADER
** Copyright (c) 1979 by tiny c associates
**
** #include "tc.h"
*/

/*
** tiny-c error codes
*/
#define STATERR          1
#define CURSERR          2
#define SYMBOLERR        3
#define RPARENERR        5
#define RANGERR          6
#define CLASSERR         7
#define SYNTAXERR        9
#define LVALERR         14
#define POPERR          15
#define PUSHERR         16
#define TMFUNERR        17
#define TMVARERR        18
#define TMVALERR        19
#define LINKERR         20
#define ARGSERR         21
#define LBRACERR        22
#define MCERR           24
#define DECLARERR       26
#define KILL            99
```

```
/*
** tiny-c assembly constants
*/
#define YES              1
#define NO               0
#define NULL             0
#define EOF             -1
#define BYTEBITS         8
#define BYTEMASK      0377

#define FOREVER   for (;;)

#define MAXSTACK        30
#define MAXVAR         200
#define MAXFUN          30
#define MAXPR        20000
#define VLEN             8

/*
** tiny-c table constants
*/
#define ACTUAL  'A'
#define LVALUE  'L'
#define FNREF   'E'
#define BYTE     1
#define WORD     2

/*
** tiny-c tokens
*/
#define INTCON   1
#define STRCON   2
#define CHRCON   3

/*
** tiny-c pseudo storage classes
*/
#define FAST    register
#define GLOBAL  extern
#define LOCAL   static
```

```
/*
** tiny-c pseudo data types
*/
typedef char TEXT;
typedef int BOOL, COUNT, VOID;
typedef unsigned BYTES;

typedef union datum {
        int     ival;
        TEXT    *dptr;
} DATUM;

typedef struct item {
        TINY    class;
        TINY    lvalue;
        TINY    size;
        DATUM   value;
} ITEM;

typedef struct variable {
        TEXT    vname[VLEN];
        TINY    vclass;
        TINY    vsize;
        COUNT   vlen;
        TEXT    *vptr;
} VARIABLE;

typedef struct function {
        VARIABLE        *fvar;
        VARIABLE        *lvar;
        TEXT            *backup;
} FUNCTION;

/*
** tiny-c macros
*/
#define alpha(c) ('a' <= (c) && (c) <= 'z' || 'A' <= (c) && (c) <= 'Z' || (c) == '_')
#define alphanum(c) (alpha(c) || '0' <= (c) && (c) <= '9')
#define num(c) (('0' <= (c) && (c) <= '9') ? (c) - '0' : -1)
```

```
/*
** TINY-C GLOBAL DECLARATIONS
** Copyright (c) 1979 by tiny c associates
**
** #include "gldecl.c"
*/

GLOBAL TEXT pr[], *cursor, *fname, *lname, *errat, *progend, *prused;
GLOBAL BOOL leave, brake;
GLOBAL COUNT err, applvl;
GLOBAL ITEM stackb[], *top;
GLOBAL VARIABLE varb[], *nxtvar;
GLOBAL FUNCTION funb[], *curfun, *curglobal;
```

```
/*
** TINY-C GLOBAL DATA DEFINITIONS
** Copyright (c) 1979 by tiny c associates
**
** Compile: cc gldata.c
*/

#include "dxl:tc.h"

TEXT pr[MAXPR+2] = { 0 };
TEXT *cursor     = { pr };
TEXT *fname      = { 0 };
TEXT *lname      = { 0 };
TEXT *errat      = { 0 };
TEXT *progend    = { pr };
TEXT *prused     = { pr };

BOOL leave = { 0 };
BOOL brake = { 0 };

COUNT err    = { 0 };
COUNT applvl = { 0 };

ITEM stackb[MAXSTACK+1] = { 0 };
ITEM *top               = { stackb - 1 };

VARIABLE varb[MAXVAR+1] = { 0 };
VARIABLE *nxtvar        = { varb };

FUNCTION funb[MAXFUN+1] = { 0 };
FUNCTION *curfun        = { funb - 1};
FUNCTION *curglobal     = { funb + 1};

#include "dxl:gldecl.c"
```

```
/*
** TINY-C INTERPRETER MAIN PROGRAM
** Copyright (c) 1979 by tiny c associates
**
** Compile: cc main
*/

#include "dx1:tc.h"
#include "dx1:gldecl.c"

VOID main(argc, argv)
COUNT argc;
TEXT *argv[];
{

        TEXT *version    = "tiny-c/11 Interpreter  Version 11-01-03";

        TEXT *copyright  = "Copyright [c] 1979 by tiny c associates";

        if(fload() < 0) return;  /* initial program load */
        prused = progend;
        link(cursor = pr);       /* link the ipl program */
        newfun();                /* allocate the lst local symbol table */
        st(cursor = pr);         /* its just a statement ... let 'er rip */
}
```

```
/*
** STATEMENT ANALYZER ROUTINES
** Copyright (c) 1979 by tiny c associates
**
** Compile: cc analyz
*/

#include "dx1:tc.h"
#include "dx1:gldecl.c"

/*
** Match a relational expression optionally followed by an assignment
*/
VOID asgn()
{
        reln();
        if(lit("=")) eq(asgn());  /* if there's an = sign, perform an assignment */
}

/*
** Evaluate an arithmetic series:
**     +- term +- term ... +- term
*/
VOID expr()
{
        int ival;

        /* the leading operator, if any, is unary */
        if (lit("-")) {
                term();
                pushint(-toptoi());  /* negative of lst term to stack */
        }
        else {
                lit("+");
                term();              /* lst term to stack */
        }

        FOREVER {
                if (lit("+")) {
                        term();
                        pushint(toptoi() + toptoi());  /* sum of top values */
                }
                else if (lit("-")) {
                        term();
                        ival = toptoi();
                        pushint(toptoi() - ival);  /* difference of top values */
                }
                else return;
        }
}
```

```
/*
** Evaluate a factor:
**      i) ( asgn )
**     ii) constant
**    iii) variable or function reference
*/
VOID factor ()
{
        int contype, index;
        DATUM value;
        VARIABLE *vbl;

        /* if you see a (, it's a ( asgn ) */
        if (lit("(")) {
                asgn();
                if (!lit(")"))
                        error(RPARENERR);
        }

        /* if you can match a constant, that's what it is */
        else if (contype = const()) {
                switch(contype) {
                case INTCON: /* integer constant */
                        pushint(atoi(fname));
                        break;
                case STRCON: /* string constant */
                        value.dptr = fname;
                        push(1, ACTUAL, BYTE, &value);
                        break;
                case CHRCON: /* character constant */
                        value.ival = *fname;
                        push(0, ACTUAL, BYTE, &value);
                        break;
                }
        }
```

```
        /* otherwise, it better be a known symbol */
        else if (symname()) {

                /* if it's MC, call it */
                if (*fname == 'M' && *lname == 'C')
                        enter(Ø);

                /* otherwise, look it up in the symbol table */
                else if (vbl = addrval()) {

                        /* if it's not a function reference, it's a variable reference */
                        if (vbl->vclass != FNREF) {
                                /* a ( means it's an array reference */
                                if (lit("(")) {
                                        if (vbl->vclass == Ø)
                                                return error(CLASSERR);
                                        /* compute the subscript */
                                        asgn();
                                        if (!lit(")"))
                                                return error(RPARENERR);
                                        index = toptoi();
                                        /* make sure it's in range */
                                        if (vbl->vlen > 1 && (index < Ø || index >= vbl->vlen))
                                                return error(RANGERR);
                                        /* compute the offset */
                                        value.dptr = vbl->vptr + index * vbl->vsize;
                                        /* and push the value */
                                        push(vbl->vclass-1, LVALUE, vbl->vsize, &value);
                                }
                                /* otherwise it's an plain variable */
                                else {
                                        /* get the address */
                                        if (vbl->vclass == Ø) value.dptr = vbl->vptr;
                                        /* force an indirection for unsubscripted array references */
                                        else value.dptr = &vbl->vptr;
                                        /* and push the value */
                                        push(vbl->vclass, LVALUE, vbl->vsize, &value);
                                }
                        }
                        /* if it's a function reference, call it */
                        else enter(vbl->vptr);
                }
                else error(SYMBOLERR);
        }
        else error(SYNTAXERR);
}
```

```
/*
** Evaluate an expression or a comparison of expressions
*/
VOID reln()
{
        int left;

        expr();  /* match the expression */

        /* look for a relational operator & apply it if found */
        if (lit("==")) {
                left = toptoi();
                expr();
                pushint(left == toptoi());
        }
        else if (lit("!=")) {
                left = toptoi();
                expr();
                pushint(left != toptoi());
        }
        else if (lit("<=")) {
                left = toptoi();
                expr();
                pushint(left <= toptoi());
        }
        else if (lit(">=")) {
                left = toptoi();
                expr();
                pushint(left >= toptoi());
        }
        else if (lit(">")) {
                left = toptoi();
                expr();
                pushint(left > toptoi());
        }
        else if (lit("<")) {
                left = toptoi();
                expr();
                pushint(left < toptoi());
        }
}
```

```
/*
** Skip a statement
*/
VOID skipst()
{
        FAST TEXT *c;

        /* bypass a leading remark */
        rem();

        /* skip a compound statement */
        if (lit("["))
                skip('[',']');

        /* skip an `if' statement */
        else if (lit("if")) {
                lit("(");
                skip('(',')');
                skipst();
                if (lit("else"))
                        skipst();
        }

        /* skip a `while' statement */
        else if (lit("while")) {
                lit("(");
                skip('(',')');
                skipst();
        }

        /* skip a one-liner */
        else {
                c = cursor;
                while(*c != ';' && *c != ']' && *c != '\n' && c <= progend) c++;
                if (c > progend)
                        return error(CURSERR);
                cursor = c + 1;
        }

        /* bypass a trailing remark */
        rem();
}
```

```
/*
** Evaluate a statement
*/
VOID st()
{
        TEXT *repeat, *body;

        /* if the error light is lit, don't bother */
        if(err) return;

        /* bypass a leading remark */
        rem();

        /* handle any beginning-of-statement business */
        stbegin();

        /* parse a `char' statement */
        if (lit("char")) {
                valloc('C', 0);
                while(lit(","))
                        valloc('C', 0);
                lit(";");
        }

        /* parse an `int' statement */
        else if (lit("int")) {
                valloc('I', 0);
                while(lit(","))
                        valloc('I', 0);
                lit(";");
        }

        /* parse a compound statement */
        else if (lit("[")) {
                while(!leave && !brake && !lit("]") && !err)
                        st();
        }
```

```
        /* parse an `if' statement */
        else if (lit("if")) {
                lit("(");
                asgn();
                lit(")");
                if (toptoi()) {
                        st();
                        if (lit("else"))
                                skipst();
                }
                else {
                        skipst();
                        if (lit("else"))
                                st();
                }
        }

        /* parse a `while' statement */
        else if (lit("while")) {
                repeat = cursor - 5;
                lit("(");
                asgn();
                lit(")");
                if (toptoi()) {
                        body = cursor;
                        st();
                        if (brake) {
                                cursor = body;
                                skipst();
                                brake = NO;
                        }
                        else cursor = repeat;
                }
                else skipst();
        }
```

```
        /* parse a null statement */
        else if (lit(";")) {
                ;
        }

        /* parse a `return' statement */
        else if (lit("return")) {
                if (lit(";") || lit("\n")) pushint(0);
                else asgn();
                leave = YES;
        }

        /* parse a `break' statement */
        else if (lit("break")) {
                brake = YES;
        }

        /* parse an expression statement */
        else {
                asgn();
                toptoi();
        }

        /* bypass optional semicolon and trailing remarks */
        lit(";");
        rem();
}
```

```
/*
** Evaluate a multiplicative series:
**      factor *%/ factor ... *%/ factor
*/
VOID term()
{
        int ival;

        factor();
        FOREVER {
                if (lit("*")) {
                        ival = toptoi(); factor();
                        pushint(ival * toptoi());  /* product of top values */
                }
                else if (lit("/")) {
                        ival = toptoi(); factor();
                        pushint(ival / toptoi());  /* quotient of top values */
                }
                else if (lit("%")) {
                        ival = toptoi(); factor();
                        pushint(ival % toptoi());  /* remainder of top values */
                }
                else return;
        }
}
```

```
/*
** Parse an element of a `char' or `int' declaration list
*/
VOID valloc(type, passed)
TINY type;
TINY *passed;
{
        TEXT *fnsave, *lnsave;
        VARIABLE vbl;

        /* get it's name */
        if (symname()) {

                vbl.vclass = 0;

                /* compute it's length */
                if (lit("(")) {
                        fnsave = fname; lnsave = lname;
                        vbl.vclass = 1;
                        /* compute the dimension */
                        asgn();
                        fname = fnsave; lname = lnsave;
                        lit(")");
                        vbl.vlen = toptoi() + 1;
                }
                else vbl.vlen = 1;

                vbl.vsize = (type == 'C') ? BYTE : WORD;

                vbl.vptr = passed;

                /* add it to the symbol table */
                newvar(&vbl);
        }
        else error(SYMBOLERR);
}
```

```
/*
** CALLING & LINKING ROUTINES
** Copyright (c) 1979 by tiny c associates
**
** Compile: cc enter
*/

#include "dx1:tc.h"
#include "dx1:gldecl.c"

/*
** Enter a called function
*/
VOID enter(entry)
TEXT *entry;
{
        TEXT *retern;
        COUNT argc;
        ITEM *arg;

        argc = 0; arg = top + 1;
        lit("(");

        /*if there are any arguments, evaluate 'em & leave their values on the stack */
        if (!lit(")") && *cursor != ';' && *cursor != '\n' && *cursor !='/') {
                FOREVER {
                        asgn();
                        argc++;
                        if (!lit(",")) break;
                }
                /* if there was trouble, clean up the stack & return */
                if (err) {
                        while (argc--) pop();
                        pushint(0);
                        return;
                }
                lit(")");
        }

        /* if it's a Machine Call, bug out to the MC routine */
        if (!entry) {
                MC(argc);
                return;
        }

        /* otherwise, it's a call on a tiny-c function */
        retern = cursor;  /* mark where you are */
        cursor = entry;   /* set cursor to where you're going */
        newfun();  /* open up a new local symbol block */
```

```
        /* transfer argument values to formal argument variables */
        FOREVER {
                rem();
                if (lit("int")) {
                        setarg('I', arg++);
                        while (lit(","))
                                setarg('I', arg++);
                        lit(";");
                }
                else if (lit("char")) {
                        setarg('C', arg++);
                        while (lit(","))
                                setarg('C', arg++);
                        lit(";");
                }
                else break;
        }

        /* if no. of argument values != no. of argument variables, light the error light */
        if (arg != top + 1) error(ARGSERR);

        /* clear the argument values off the stack */
        while(argc--) pop();

        /* if everything is ok, turn the interpreter loose in the function */
        if(!err) st();

        /* if the function didn't return a value, push a zero */
        if(leave == NO) pushint(0);
        else leave = NO;

        /* pop the cursor & restore caller's local symbols */
        cursor = retern;
        fundone();
}
```

```
/*
** Allocate externals
*/
VOID link()
{
        VARIABLE vbl;

        newfun();  /* establish the system global level */
        while (cursor < progend - 1 && !err) {
                rem();

                /* skip external statements */
                if (lit("[")) {
                        skip('[', ']');
                }

                /* allocate an external `int' list */
                else if (lit("int")) {
                        valloc('I', 0);
                        while (lit(","))
                                valloc('I', 0);
                        lit(";");
                }

                /* allocate an external `char' list */
                else if (lit("char")) {
                        valloc('C', 0);
                        while (lit(","))
                                valloc('C', 0);
                        lit(";");
                }

                /* shift from system globals to regular globals */
                else if (lit("endlibrary"))
                        newfun();

                /* allocate a function name */
                else if (symname()) {
                        vbl.vclass = FNREF; vbl.vsize = WORD;
                        vbl.vlen = 1; vbl.vptr = cursor;
                        newvar(&vbl);
                        /* skip over the function's body */
                        while (*cursor != '[' && cursor < progend) cursor++;
                        if (*cursor != '[' ) return error(LBRACERR);
                        else {
                                cursor++;
                                skip('[', ']');
                        }
                }
                else error(LINKERR);
        }
}
```

```
/*
** Set formal function argument variables to their called values
*/
VOID setarg(type, arg)
ITEM *arg;
TINY type;
{
        FAST unsigned word;

        /* if it's an actual, just set it */
        if (arg->lvalue == ACTUAL)
                valloc(type, arg->value.ival);

        /* if it's a byte indirect, the call will make it an `int' */
        else if (arg->size == BYTE && arg->class == 0)
                valloc(type, *arg->value.dptr);

        /* otherwise, we have to assemble it ourselves */
        else {
                word = *(arg->value.dptr + 1);
                word = (word << BYTEBITS) | (*arg->value.dptr & BYTEMASK);
                valloc(type, word);
        }
}
```

```
/*
** SCANNING TOOLS
** Copyright (c) 1979 by tiny c assocaites
**
** Compile: cc scan
*/

#include "dxl:tc.h"
#include "dxl:gldecl.c"

/*
** Skip over blanks & tabs
*/
TEXT *blanks()
{
        FAST TEXT *c = cursor;

        while (*c == ' ' || *c == '\t') c++;
        return cursor = c;
}

/*
** Match integer, string, and character constants
*/
COUNT const()
{
        FAST TEXT *c;

        c = blanks();

        /* try for an integer constant */
        if (*c == '+' || *c == '-' || num(*c) != -1) {
                fname = cursor;
                while (num(*c) != -1) c++ ;
                lname = c-1; cursor = c;
                return INTCON;
        }

        /* try for a string */
        else if (*c == '\"') {
                fname = ++c;
                while (*c && *c != '\"' && c++ < progend);
                if (c <= progend) {
                        *c++ = 0; cursor = c; lname = c-2;
                        return STRCON;
                }
                else return error(CURSERR);
        }
```

```
        /* try for a character constant */
        else if (*c == '\'') {
                fname = ++c;
                while (*c != '\'' && c++ < progend);
                if (c <= progend) {
                        cursor = ++c; lname = c-2;
                        return CHRCON;
                }
                else return error(CURSERR);
        }

        /* none of the above */
        else return 0;
}

/*
** Match a literal
*/
BOOL lit(s)
FAST TEXT *s;
{
        FAST TEXT *c = cursor;

        if(*c == ' ' || *c == '\t') c = blanks();
        if(*c++ != *s++) return NO;
        while (*s)
                if (*c++ != *s++) return NO;
        cursor = c;
        return YES;
}
```

```
/*
** Skip over remarks and adjacent white space
*/
TEXT *rem()
{
        FAST TEXT *c;

        while((c = blanks()) <= progend) {
                if (*c == '\n') c++;
                else if (*c != '/' || *(c + 1) != '*')
                        return cursor = c;
                else while (*c++ != '\n' && c <= progend);
                cursor = c;
        }
        return progend;
}

/*
** Skip balanced delimiters and everyting inbetween
*/
TEXT *skip(left, right)
char left, right;
{
        FAST TEXT *c;
        FAST COUNT parity;

        parity = 1;
        for (c = cursor; c <= progend; c++) {
                if (*c == left) parity++;
                else if (*c == right) parity--;
                if (parity == 0)
                        return cursor = ++c;
        }
        return error(CURSERR);
}

/*
** Match a symbol name and set fname & lname
*/
BOOL symname()
{
        FAST TEXT *c;

        c = blanks();
        if (!alpha(*c)) return NO;
        fname = c++;
        while (alphanum(*c)) c++;
        lname = c-1; cursor = c;
        return YES;
}
```

```
/*
** STACK TOOLS
** Copyright (c) 1979 by tiny c associates
**
** Compile: cc stack
*/

#include "dxl:tc.h"
#include "dxl:gldecl.c"

/*
** Perform an assignment
*/
VOID eq()
{

        int ival;

        ival = toptoi();
        /* make sure you've got an lvalue */
        if (top->lvalue == LVALUE) {
                if (top->class == Ø)
                        movn(top->value.dptr, &ival, top->size);
                else
                        movn(top->value.dptr, &ival, WORD);
                pop();
                /* assignments have values too */
                pushint(ival);
        }
        else error(LVALERR);
}

/*
** Pop the tiny-c stack & return the ival
*/
int pop()
{
        if (top >= stackb) return (top--)->value.ival;
        else return error(POPERR);
}
```

```
/*
** Push item parts onto the tiny-c stack
*/
VOID push(pclass, plvalue, psize, pdatum)
TINY pclass, plvalue, psize;
DATUM *pdatum;
{
        if (++top <= stackb + MAXSTACK) {
                top->class = pclass;
                top->lvalue = plvalue;
                top->size = psize;
                if (pclass == ACTUAL && plvalue == Ø) top->value.ival = pdatum->ival;
                else top->value.dptr = pdatum->dptr;
        }
        else error(PUSHERR);
}

/*
** Push an integer onto the tiny-c stack
*/
pushint(ival)
int ival;
{
        if (++top <= stackb + MAXSTACK) {
                top->class = Ø;
                top->lvalue = ACTUAL;
                top->size = WORD;
                top->value.ival = ival;
        }
        else error(PUSHERR);
}
```

```
/*
** Resolve the item on the top of the stack & return it
*/
int toptoi()
{

        FAST int topval;

        /* if it's an actual, just pop it */
        if (top->lvalue == ACTUAL)
                return pop();

        /* if it's a byte indirect, sign extend it */
        else if (top->size == BYTE && top->class == 0)
                topval = *top->value.dptr;

        /* otherwise, assemble it */
        else {
                topval = *(top->value.dptr + 1);
                topval = (topval << BYTEBITS) | (*top->value.dptr & BYTEMASK);
        }
        pop();
        return topval;
}
```

```
/*
** SYMBOL TABLE TOOLS
** Copyright (c) 1979 by tiny c associates
**
** Compile: cc symbol
*/

#include "dx1:tc.h"
#include "dx1:gldecl.c"

/*
** Search 1) current locals
**        2) application globals
** and    3) system globals
** for a symbol name.
*/
VARIABLE *addrval()
{
        TEXT name[VLEN];
        FAST FUNCTION *sfun;
        FAST VARIABLE *pvar;

        canon(name);
        for (sfun = curfun; ; sfun = (sfun == curfun) ? curglobal : funb) {
                for (pvar = sfun->fvar; pvar <= sfun->lvar; pvar++)
                        if (ceqn(name, pvar->vname, VLEN))
                                return pvar;
                if (sfun == funb) break;
        }
        error(DECLARERR);
        return 0;
}
```

```
/*
** Canonicalize the symbol between fname & lname
*/
VOID canon(name)
FAST TEXT *name;
{
        FAST COUNT i;

        for (i = VLEN; i > 0 && fname <= lname; i--)
                *name++ = *fname++;
        if(i == 0) *(name-1) = *lname;
        else *name = '\0';
}

/*
** Pop function stack ... return to previous local symbol table
*/
VOID fundon()
{
        nxtvar = curfun->fvar;
        prused = (curfun--)->backup;
}

/*
** Open up a new local symbol table
*/
VOID newfun()
{
        if(++curfun <= funb + MAXFUN) {
                curfun->fvar = nxtvar;
                curfun->lvar = nxtvar - 1;
                curfun->backup = prused;
        }
        else error(TMFUNERR);
}
```

```
/*
** Push a new variable on the variable stack
*/
VOID newvar(vbl)
FAST VARIABLE *vbl;
{
        FAST TEXT *c;

        if (nxtvar <= varb + MAXVAR) {
                canon(nxtvar->vname);
                nxtvar->vclass = vbl->vclass;
                nxtvar->vsize = vbl->vsize;
                nxtvar->vlen = vbl->vlen;

                /* if it's indirect, just pass the pointer */
                if (vbl->vptr != 0 && vbl->vclass != 0)
                        nxtvar->vptr = vbl->vptr;

                /* otherwise, allocate some new room */
                else {
                        nxtvar->vptr = prused + 1;
                        if (((long) prused + vbl->vsize * vbl->vlen) <= (long) pr + MAXPR) {
                                prused += vbl->vsize * vbl->vlen;
                                /* if it's being passed, move the value */
                                if (vbl->vptr)
                                        movn(nxtvar->vptr, &vbl->vptr, vbl->vsize);
                                /* otherwise, zero out the new space */
                                else
                                        for (c = nxtvar->vptr; c <= prused; *c++ = '\0');
                        }
                        else error(TMVALERR);
                }

                /* expand the current local symbol table */
                curfun->lvar = nxtvar++;
        }
        else error(TMVARERR);
}
```

```
/*
** UTILITY FUNCTIONS
** Copyright (c) 1979 by tiny c associates
**
** Compile: cc util
*/

#include "dxl:tc.h"
#include "dxl:gldecl.c"

/*
** Convert an ASCII string to an integer
*/
int atoi(c)
FAST TEXT *c;
{
        BOOL sign;
        FAST int digit, value;

        sign = NO;
        if (*c == '+' || *c == '-')
                sign = (*c++ == '-') ? YES : NO;
        for(value = 0; (digit = num(*c)) >= 0; c++)
                value = 10 * value + digit;
        return sign ? -value : value;
}

/*
** Test for equality of two strings
*/
BOOL ceq(a, b)
FAST TEXT *a, *b;
{
        while(*a == *b++)
                if (*a++ == '\0') return YES;
        return NO;
}
```

```
/*
** Test if first n characters of two strings are equal
*/
BOOL ceqn(a, b, n)
FAST TEXT *a, *b;
FAST COUNT n;
{
        while(n-- > 0) {
                if (*a != *b++) return NO;
                if (*a++ == 0) return YES;
        }
        return YES;
}

/*
** Set the error flags
*/
VOID error(number)
int number;
{
        if (!err) {
                err = number;
                errat = cursor;
        }
}

/*
** Move n bytes from b to a without regard for overlap
*/
VOID movn(a, b, n)
FAST TEXT *a, *b;
FAST COUNT n;
{
        while(n-- > 0)
                *a++ = *b++;
}
```

```
                                .TITLE TINY-C/11-01-03/GLDATA
                                .DSABL REG
                                .ENABL GBL
000000                          .PSECT DATA
000000                          PR:
000000      000                 .BYTE 0
                                .BLKB 47041
                                .EVEN
047042                          CURSOR:
047042  000000'                 .WORD PR
                                .EVEN
047044                          FNAME:
047044  000000                  .WORD 0
                                .EVEN
047046                          LNAME:
047046  000000                  .WORD 0
                                .EVEN
047050                          ERRAT:
047050  000000                  .WORD 0
                                .EVEN
047052                          PROGEND:
047052  000000'                 .WORD PR
                                .EVEN
047054                          PRUSED:
047054  000000'                 .WORD PR
                                .EVEN
047056                          LEAVE:
047056  000000                  .WORD 0
                                .EVEN
047060                          BRAKE:
047060  000000                  .WORD 0
                                .EVEN
047062                          ERR:
047062  000000                  .WORD 0
                                .EVEN
047064                          APPLVL:
047064  000000                  .WORD 0
                                .EVEN
```

```
 39 047066                      STACKB:
 40 047066     000              .BYTE 0
 41                             .BLKB 5
 42                             .BLKB 264
 43                             .EVEN
 44 047360                      TOP:
 45 047360  047060'             .WORD STACKB-6
 46                             .EVEN
 47 047362                      VARB:
 48 047362     000              .BYTE 0
 49                             .BLKB 7
 50                             .BLKB 6
 51                             .BLKB 5360
 52                             .EVEN
 53 054760                      NXTVAR:
 54 054760  047362'             .WORD VARB
 55                             .EVEN
 56 054762                      FUNB:
 57 054762  000000              .WORD 0
 58                             .BLKB 4
 59                             .BLKB 264
 60                             .EVEN
 61 055254                      CURFUN:
 62 055254  054754'             .WORD FUNB-6
 63                             .EVEN
 64 055256                      CURGLOB:
 65 055256  054770'             .WORD FUNB+6
 66                             .GLOBL CURGLOB
 67                             .GLOBL CURFUN
 68                             .GLOBL FUNB
 69                             .GLOBL NXTVAR
 70                             .GLOBL VARB
 71                             .GLOBL TOP
 72                             .GLOBL STACKB
 73                             .GLOBL APPLVL
 74                             .GLOBL ERR
 75                             .GLOBL BRAKE
 76                             .GLOBL LEAVE
 77                             .GLOBL PRUSED
 78                             .GLOBL PROGEND
 79                             .GLOBL ERRAT
 80                             .GLOBL LNAME
 81                             .GLOBL FNAME
 82                             .GLOBL CURSOR
 83                             .GLOBL PR
 84         000001              .END
```

```
                                        .TITLE TINY-C/11-01-03/MAIN
                                        .DSABL REG
                                        .ENABL GBL
                                        .PSECT DATA
                                        $3:
000000  103     157     160     .BYTE 103,157,160,171,162,151,147,150
000003  171     162     151
000006  147     150
000010  164     040     133     .BYTE 164,40,133,143,135,40,61,71
000013  143     135     040
000016  061     071
000020  067     071     040     .BYTE 67,71,40,142,171,40,164,151
000023  142     171     040
000026  164     151
000030  156     171     040     .BYTE 156,171,40,143,40,141,163,163
000033  143     040     141
000036  163     163
000040  157     143     151     .BYTE 157,143,151,141,164,145,163,0
000043  141     164     145
000046  163     000
000050                          $1:
000050  164     151     156     .BYTE 164,151,156,171,55,143,57,61
000053  171     055     143
000056  057     061
000060  061     040     111     .BYTE 61,40,111,156,164,145,162,160
000063  156     164     145
000066  162     160
000070  162     145     164     .BYTE 162,145,164,145,162,40,40,126
000073  145     162     040
000076  040     126
000100  145     162     163     .BYTE 145,162,163,151,157,156,40,61
000103  151     157     156
000106  040     061
000110  061     055     060     .BYTE 61,55,60,61,55,60,63,0
000113  061     055     060
000116  063     000
```

```
                                        .GLOBL ST
                                        .GLOBL NEWFUN
                                        .GLOBL LINK
                                        .GLOBL FLOAD
                                        .GLOBL MAIN
                                        .GLOBL CURGLOB
                                        .GLOBL CURFUN
                                        .GLOBL FUNB
                                        .GLOBL NXTVAR
                                        .GLOBL VARB
                                        .GLOBL TOP
                                        .GLOBL STACKB
                                        .GLOBL APPLVL
                                        .GLOBL ERR
                                        .GLOBL BRAKE
                                        .GLOBL LEAVE
                                        .GLOBL PRUSED
                                        .GLOBL PROGEND
                                        .GLOBL ERRAT
                                        .GLOBL LNAME
                                        .GLOBL FNAME
                                        .GLOBL CURSOR
                                        .GLOBL PR
000000                                  .PSECT TEXT
000000                                  MAIN:
000000  010546                          MOV %5,-(%6)
000002  010605                          MOV %6,%5
000004  062706  177764                  ADD #-14,%6
000010  012765  000050' 177770          MOV #$1,-10(%5)
000016  012765  000000' 177766          MOV #$3,-12(%5)
000024  004767  000000G                 JSR %7,FLOAD
000030  005700                          TST %0
000032  002423                          BLT $2
000034  016767  000000G 000000G        MOV PROGEND,PRUSED
000042  012767  000000G 000000G        MOV #PR,CURSOR
000050  016716  000000G                 MOV CURSOR,(%6)
000054  004767  000000G                 JSR %7,LINK
000060  004767  000000G                 JSR %7,NEWFUN
000064  012767  000000G 000000G        MOV #PR,CURSOR
000072  016716  000000G                 MOV CURSOR,(%6)
000076  004767  000000G                 JSR %7,ST
000102                                  $2:
000102  104350                          EMT ^0350
        000000'                         .END MAIN
```

```
 1                                      .TITLE TINY-C/11-01-03/ANALYZ
 2                                      .DSABL REG
 3                                      .ENABL GBL
 4 000000                               .PSECT DATA
 5 000000                               $3:
 6 000000      075     000              .BYTE 75,0
 7 000000                               .PSECT TEXT
 8 000000                               ASGN:
 9 000000  010546                       MOV %5,-(%6)
10 000002  010605                       MOV %6,%5
11 000004  005746                       TST -(%6)
12 000006  004767  001212               JSR %7,RELN
13 000012  012716  000000'              MOV #$3,(%6)
14 000016  004767  000000G              JSR %7,LIT
15 000022  005700                       TST %0
16 000024  001405                       BEQ $1
17 000026  004767  177746               JSR %7,ASGN
18 000032  010016                       MOV %0,(%6)
19 000034  004767  000000G              JSR %7,EQ
20 000040                               $1: ;17
21 000040  000167  000000G              JMP RETS
22 000002                               .PSECT DATA
23 000002                               $13:
24 000002      055     000              .BYTE 55,0
25 000004                               $32:
26 000004      053     000              .BYTE 53,0
27 000006                               $31:
28 000006      053     000              .BYTE 53,0
29 000010                               $7:
30 000010      055     000              .BYTE 55,0
31 000044                               .PSECT TEXT
32 000044                               EXPR:
33 000044  010546                       MOV %5,-(%6)
34 000046  010605                       MOV %6,%5
35 000050  062706  177766               ADD #-12,%6
36 000054  012716  000010'              MOV #$7,(%6)
37 000060  004767  000000G              JSR %7,LIT
38 000064  005700                       TST %0
39 000066  001411                       BEQ $5
40 000070  004767  003126               JSR %7,TERM
41 000074  004767  000000G              JSR %7,TOPTOI
42 000100  010016                       MOV %0,(%6)
43 000102  005416                       NEG (%6)
44 000104  004767  000000G              JSR %7,PUSHINT
45 000110  000406                       BR $51
46 000112                               $5: ;21
47 000112  012716  000006'              MOV #$31,(%6)
48 000116  004767  000000G              JSR %7,LIT
```

```
 49 000122  004767  003074          JSR %7,TERM
 50 000126                          $51: ; 27
 51 000126  012716  000004'         MOV #$32,(%6)
 52 000132  004767  000000G         JSR %7,LIT
 53 000136  005700                  TST %0
 54 000140  001413                  BEQ $12
 55 000142  004767  003054          JSR %7,TERM
 56 000146  004767  000000G         JSR %7,TOPTOI
 57 000152  010016                  MOV %0,(%6)
 58 000154  004767  000000G         JSR %7,TOPTOI
 59 000160  060016                  ADD %0,(%6)
 60 000162  004767  000000G         JSR %7,PUSHINT
 61 000166  000757                  BR $51
 62 000170                          $12: ;50
 63 000170  012716  000002'         MOV #$13,(%6)
 64 000174  004767  000000G         JSR %7,LIT
 65 000200  005700                  TST %0
 66 000202  001416                  BEQ $72
 67 000204  004767  003012          JSR %7,TERM
 68 000210  004767  000000G         JSR %7,TOPTOI
 69 000214  010065  177770          MOV %0,-10(%5)
 70 000220  004767  000000G         JSR %7,TOPTOI
 71 000224  010016                  MOV %0,(%6)
 72 000226  166516  177770          SUB -10(%5),(%6)
 73 000232  004767  000000G         JSR %7,PUSHINT
 74 000236  000733                  BR $51
 75 000240                          $72: ;74
 76 000240  000167  000000G         JMP RETS
 77 000012                          .PSECT DATA
 78                                 .EVEN
 79 000012                          $15:
 80 000012  000474'                 .WORD $16
 81 000014  000003                  .WORD 3
 82 000016  000432'                 .WORD $75
 83 000020  000002                  .WORD 2
 84 000022  000410'                 .WORD $55
 85 000024  000001                  .WORD 1
 86 000026  000000                  .WORD 0
 87 000030  000430'                 .WORD $35
 88 000032                          $701:
 89 000032     051     000          .BYTE 51,0
 90 000034                          $101:
 91 000034     050     000          .BYTE 50,0
 92 000036                          $34:
 93 000036     051     000          .BYTE 51,0
 94 000040                          $73:
 95 000040     050     000          .BYTE 50,0
```

```
 96 000244                              .PSECT TEXT
 97 000244                              FACTOR:
 98 000244  004567  000000G             JSR %5,SAVE
 99 000250  062706  177770              ADD #-10,%6
100 000254  012716  000040'             MOV #$73,(%6)
101 000260  004767  000000G             JSR %7,LIT
102 000264  005700                      TST %0
103 000266  001415                      BEQ $53
104 000270  004767  177504              JSR %7,ASGN
105 000274  012716  000036'             MOV #$34,(%6)
106 000300  004767  000000G             JSR %7,LIT
107 000304  005700                      TST %0
108 000306  001117                      BNE $54
109 000310  012716  000005              MOV #5,(%6)
110 000314  004767  000000G             JSR %7,ERROR
111 000320  000512                      BR $54
112 000322                              $53: ;25
113 000322  004767  000000G             JSR %7,CONST
114 000326  010065  177770              MOV %0,-10(%5)
115 000332  001406                      BEQ $74
116 000334  016500  177770              MOV -10(%5),%0
117 000340  012701  000012'             MOV #$15,%1
118 000344  000167  000000G             JMP SWITCH
119 000350                              $74: ;40
120 000350  004767  000000G             JSR %7,SYMNAME
121 000354  005700                      TST %0
122 000356  001467                      BEQ $56
123 000360  127727  000000G 000115      CMPB @FNAME,#115
124 000366  001070                      BNE $76
125 000370  127727  000000G 000103      CMPB @LNAME,#103
126 000376  001064                      BNE $76
127 000400  005016                      CLR (%6)
128 000402  004767  000000G             JSR %7,ENTER
129 000406  000457                      BR $54
130 000410                              $55: ;60
131 000410  016716  000000G             MOV FNAME,(%6)
132 000414  004767  000000G             JSR %7,ATOI
133 000420  010016                      MOV %0,(%6)
134 000422  004767  000000G             JSR %7,PUSHINT
135 000426  000447                      BR $54
136 000430                              $35: ;70
137 000430  000446                      BR $54
138 000432                              $75: ;71
139 000432  016765  000000G 177764      MOV FNAME,-14(%5)
140 000440  010516                      MOV %5,(%6)
141 000442  062716  177764              ADD #-14,(%6)
142 000446  012746  000001              MOV #1,-(%6)
143 000452  012746  000101              MOV #101,-(%6)
144 000456  012746  000001              MOV #1,-(%6)
145 000462  004767  000000G             JSR %7,PUSH
146 000466  062706  000006              ADD #6,%6
```

```
147 000472  000425                  BR $54
148 000474                          $16: ;112
149 000474  117700  000000G         MOVB @FNAME,%0
150 000500  010065  177764          MOV %0,-14(%5)
151 000504  010516                  MOV %5,(%6)
152 000506  062716  177764          ADD #-14,(%6)
153 000512  012746  000001          MOV #1,-(%6)
154 000516  012746  000101          MOV #101,-(%6)
155 000522  005046                  CLR -(%6)
156 000524  004767  000000G         JSR %7,PUSH
157 000530  062706  000006          ADD #6,%6
158 000534  000404                  BR $54
159 000536                          $56: ;133
160 000536  012716  000011          MOV #11,(%6)
161 000542  004767  000000G         JSR %7,ERROR
162 000546                          $54: ; 137
163 000546  000432                  BR $2
164 000550                          $76: ;140
165 000550  004767  000000G         JSR %7,ADDRVAL
166 000554  010065  177762          MOV %0,-16(%5)
167 000560  001427                  BEQ $37
168 000562  016500  177762          MOV -16(%5),%0
169 000566  126027  000010  000105  CMPB 10(%0),#105
170 000574  001426                  BEQ $57
171 000576  012716  000034'         MOV #$101,(%6)
172 000602  004767  000000G         JSR %7,LIT
173 000606  005700                  TST %0
174 000610  001427                  BEQ $77
175 000612  016500  177762          MOV -16(%5),%0
176 000616  105760  000010          TSTB 10(%0)
177 000622  001035                  BNE $301
178 000624  012716  000007          MOV #7,(%6)
179 000630  004767  000000G         JSR %7,ERROR
180 000634                          $2:
181 000634  000167  000000G         JMP RET
182 000640                          $37: ;174
183 000640  012716  000003          MOV #3,(%6)
184 000644  004767  000000G         JSR %7,ERROR
185 000650  000736                  BR $54
186 000652                          $57: ;201
187 000652  016500  177762          MOV -16(%5),%0
188 000656  016016  000014          MOV 14(%0),(%6)
189 000662  004767  000000G         JSR %7,ENTER
190 000666  000727                  BR $54
191 000670                          $77: ;210
192 000670  016500  177762          MOV -16(%5),%0
193 000674  105760  000010          TSTB 10(%0)
194 000700  001116                  BNE $711
195 000702  016500  177762          MOV -16(%5),%0
196 000706  016065  000014  177764  MOV 14(%0),-14(%5)
```

```
197 000714  000516                  BR $121
198 000716                          $301: ;223
199 000716  004767  177056          JSR %7,ASGN
200 000722  012716  000032'         MOV #$701,(%6)
201 000726  004767  000000G         JSR %7,LIT
202 000732  005700                  TST %0
203 000734  001005                  BNE $501
204 000736  012716  000005          MOV #5,(%6)
205 000742  004767  000000G         JSR %7,ERROR
206 000746  000732                  BR $2
207 000750                          $501: ;240
208 000750  004767  000000G         JSR %7,TOPTOI
209 000754  010065  177766          MOV %0,-12(%5)
210 000760  016500  177762          MOV -16(%5),%0
211 000764  026027  000012  000001  CMP 12(%0),#1
212 000772  003411                  BLE $111
213 000774  005765  177766          TST -12(%5)
214 001000  002451                  BLT $311
215 001002  016500  177762          MOV -16(%5),%0
216 001006  026560  177766  000012  CMP -12(%5),12(%0)
217 001014  002043                  BGE $311
218 001016                          $111: ;263
219 001016  016500  177762          MOV -16(%5),%0
220 001022  016000  000014          MOV 14(%0),%0
221 001026  016501  177762          MOV -16(%5),%1
222 001032  116103  000011          MOVB 11(%1),%3
223 001036  070365  177766          MUL -12(%5),%3
224 001042  060300                  ADD %3,%0
225 001044  010065  177764          MOV %0,-14(%5)
226 001050  010516                  MOV %5,(%6)
227 001052  062716  177764          ADD #-14,(%6)
228 001056  016500  177762          MOV -16(%5),%0
229 001062  116000  000011          MOVB 11(%0),%0
230 001066  010046                  MOV %0,-(%6)
231 001070  012746  000114          MOV #114,-(%6)
232 001074  016500  177762          MOV -16(%5),%0
233 001100  116000  000010          MOVB 10(%0),%0
234 001104  010046                  MOV %0,-(%6)
235 001106  062716  177777          ADD #-1,(%6)
236 001112  004767  000000G         JSR %7,PUSH
237 001116  062706  000006          ADD #6,%6
238 001122  000611                  BR $54
```

```
239 001124                              $311: ;326
240 001124  012716  000006              MOV #6,(%6)
241 001130  004767  000000G             JSR %7,ERROR
242 001134  000637                      BR $2
243 001136                              $711: ;333
244 001136  016500  177762              MOV -16(%5),%0
245 001142  062700  000014              ADD #14,%0
246 001146  010065  177764              MOV %0,-14(%5)
247 001152                              $121: ; 341
248 001152  010516                      MOV %5,(%6)
249 001154  062716  177764              ADD #-14,(%6)
250 001160  016500  177762              MOV -16(%5),%0
251 001164  116000  000011              MOVB 11(%0),%0
252 001170  010046                      MOV %0,-(%6)
253 001172  012746  000114              MOV #114,-(%6)
254 001176  016500  177762              MOV -16(%5),%0
255 001202  116000  000010              MOVB 10(%0),%0
256 001206  010046                      MOV %0,-(%6)
257 001210  004767  000000G             JSR %7,PUSH
258 001214  062706  000006              ADD #6,%6
259 001220  000167  177322              JMP $54
260 000042                              .PSECT DATA
261 000042                              $171:
262 000042     074     000              .BYTE 74,0
263 000044                              $361:
264 000044     076     000              .BYTE 76,0
265 000046                              $551:
266 000046     076     075     000      .BYTE 76,75,0
267 000051                              $741:
268 000051     074     075     000      .BYTE 74,75,0
269 000054                              $141:
270 000054     041     075     000      .BYTE 41,75,0
271 000057                              $331:
272 000057     075     075     000      .BYTE 75,75,0
```

```
273 001224                          .PSECT TEXT
274 001224                          RELN:
275 001224  010546                  MOV %5,-(%6)
276 001226  010605                  MOV %6,%5
277 001230  062706  177766          ADD #-12,%6
278 001234  004767  176604          JSR %7,EXPR
279 001240  012716  000057'         MOV #$331,(%6)
280 001244  004767  000000G         JSR %7,LIT
281 001250  005700                  TST %0
282 001252  001423                  BEQ $131
283 001254  004767  000000G         JSR %7,TOPTOI
284 001260  010065  177770          MOV %0,-10(%5)
285 001264  004767  176554          JSR %7,EXPR
286 001270  004767  000000G         JSR %7,TOPTOI
287 001274  020065  177770          CMP %0,-10(%5)
288 001300  001003                  BNE $4
289 001302  012700  000001          MOV #1,%0
290 001306  000401                  BR $6
291 001310                          $4:
292 001310  005000                  CLR %0
293 001312                          $6:
294 001312  010016                  MOV %0,(%6)
295 001314  004767  000000G         JSR %7,PUSHINT
296 001320  000430                  BR $531
297 001322                          $131: ;35
298 001322  012716  000054'         MOV #$141,(%6)
299 001326  004767  000000G         JSR %7,LIT
300 001332  005700                  TST %0
301 001334  001424                  BEQ $731
302 001336  004767  000000G         JSR %7,TOPTOI
303 001342  010065  177770          MOV %0,-10(%5)
304 001346  004767  176472          JSR %7,EXPR
305 001352  004767  000000G         JSR %7,TOPTOI
306 001356  020065  177770          CMP %0,-10(%5)
307 001362  001403                  BEQ $01
308 001364  012700  000001          MOV #1,%0
309 001370  000401                  BR $21
310 001372                          $01:
311 001372  005000                  CLR %0
312 001374                          $21:
313 001374  010016                  MOV %0,(%6)
314 001376  004767  000000G         JSR %7,PUSHINT
315 001402                          $531: ;65
316 001402  000167  000000G         JMP RETS
```

```
317 001406                          $731: ;67
318 001406  012716  000051'         MOV #$741,(%6)
319 001412  004767  000000G         JSR %7,LIT
320 001416  005700                  TST %0
321 001420  001423                  BEQ $541
322 001422  004767  000000G         JSR %7,TOPTOI
323 001426  010065  177770          MOV %0,-10(%5)
324 001432  004767  176406          JSR %7,EXPR
325 001436  004767  000000G         JSR %7,TOPTOI
326 001442  026500  177770          CMP -10(%5),%0
327 001446  003003                  BGT $41
328 001450  012700  000001          MOV #1,%0
329 001454  000401                  BR $61
330 001456                          $41:
331 001456  005000                  CLR %0
332 001460                          $61:
333 001460  010016                  MOV %0,(%6)
334 001462  004767  000000G         JSR %7,PUSHINT
335 001466  000745                  BR $531
336 001470                          $541: ;120
337 001470  012716  000046'         MOV #$551,(%6)
338 001474  004767  000000G         JSR %7,LIT
339 001500  005700                  TST %0
340 001502  001423                  BEQ $351
341 001504  004767  000000G         JSR %7,TOPTOI
342 001510  010065  177770          MOV %0,-10(%5)
343 001514  004767  176324          JSR %7,EXPR
344 001520  004767  000000G         JSR %7,TOPTOI
345 001524  026500  177770          CMP -10(%5),%0
346 001530  002403                  BLT $02
347 001532  012700  000001          MOV #1,%0
348 001536  000401                  BR $22
349 001540                          $02:
350 001540  005000                  CLR %0
351 001542                          $22:
352 001542  010016                  MOV %0,(%6)
353 001544  004767  000000G         JSR %7,PUSHINT
354 001550  000714                  BR $531
355 001552                          $351: ;151
356 001552  012716  000044'         MOV #$361,(%6)
357 001556  004767  000000G         JSR %7,LIT
358 001562  005700                  TST %0
359 001564  001423                  BEQ $161
360 001566  004767  000000G         JSR %7,TOPTOI
361 001572  010065  177770          MOV %0,-10(%5)
362 001576  004767  176242          JSR %7,EXPR
363 001602  004767  000000G         JSR %7,TOPTOI
364 001606  026500  177770          CMP -10(%5),%0
365 001612  003403                  BLE $42
366 001614  012700  000001          MOV #1,%0
367 001620  000401                  BR $62
```

```
368 001622                          $42:
369 001622  005000                  CLR %0
370 001624                          $62:
371 001624  010016                  MOV %0,(%6)
372 001626  004767  000000G         JSR %7,PUSHINT
373 001632  000663                  BR $531
374 001634                          $161: ;202
375 001634  012716  000042'         MOV #$171,(%6)
376 001640  004767  000000G         JSR %7,LIT
377 001644  005700                  TST %0
378 001646  001655                  BEQ $531
379 001650  004767  000000G         JSR %7,TOPTOI
380 001654  010065  177770          MOV %0,-10(%5)
381 001660  004767  176160          JSR %7,EXPR
382 001664  004767  000000G         JSR %7,TOPTOI
383 001670  026500  177770          CMP -10(%5),%0
384 001674  002003                  BGE $03
385 001676  012700  000001          MOV #1,%0
386 001702  000401                  BR $23
387 001704                          $03:
388 001704  005000                  CLR %0
389 001706                          $23:
390 001706  010016                  MOV %0,(%6)
391 001710  004767  000000G         JSR %7,PUSHINT
392 001714  000632                  BR $531
393 000062                          .PSECT DATA
394 000062                          $122:
395 000062     050     000          .BYTE 50,0
396 000064                          $712:
397 000064     167     150     151  .BYTE 167,150,151,154,145,0
    000067     154     145     000
398 000072                          $112:
399 000072     145     154     163  .BYTE 145,154,163,145,0
    000075     145     000
400 000077                          $502:
401 000077     050     000          .BYTE 50,0
402 000101                          $302:
403 000101     151     146     000  .BYTE 151,146,0
404 000104                          $571:
405 000104     133     000          .BYTE 133,0
```

```
406 001716                          .PSECT TEXT
407 001716                          SKIPST:
408 001716  004567  000000G         JSR %5,SAVE
409 001722  004767  000000G         JSR %7,REM
410 001726  012716  000104'         MOV #$571,(%6)
411 001732  004767  000000G         JSR %7,LIT
412 001736  005700                  TST %0
413 001740  001410                  BEQ $371
414 001742  012716  000135          MOV #135,(%6)
415 001746  012746  000133          MOV #133,-(%6)
416 001752  004767  000000G         JSR %7,SKIP
417 001756  005726                  TST (%6)+
418 001760  000514                  BR $771
419 001762                          $371: ;22
420 001762  012716  000101'         MOV #$302,(%6)
421 001766  004767  000000G         JSR %7,LIT
422 001772  005700                  TST %0
423 001774  001426                  BEQ $102
424 001776  012716  000077'         MOV #$502,(%6)
425 002002  004767  000000G         JSR %7,LIT
426 002006  012716  000051          MOV #51,(%6)
427 002012  012746  000050          MOV #50,-(%6)
428 002016  004767  000000G         JSR %7,SKIP
429 002022  005726                  TST (%6)+
430 002024  004767  177666          JSR %7,SKIPST
431 002030  012716  000072'         MOV #$112,(%6)
432 002034  004767  000000G         JSR %7,LIT
433 002040  005700                  TST %0
434 002042  001463                  BEQ $771
435 002044  004767  177646          JSR %7,SKIPST
436 002050  000460                  BR $771
437 002052                          $102: ;56
438 002052  012716  000064'         MOV #$712,(%6)
439 002056  004767  000000G         JSR %7,LIT
440 002062  005700                  TST %0
441 002064  001416                  BEQ $512
442 002066  012716  000062'         MOV #$122,(%6)
443 002072  004767  000000G         JSR %7,LIT
444 002076  012716  000051          MOV #51,(%6)
445 002102  012746  000050          MOV #50,-(%6)
446 002106  004767  000000G         JSR %7,SKIP
447 002112  005726                  TST (%6)+
448 002114  004767  177576          JSR %7,SKIPST
449 002120  000434                  BR $771
```

```
450 002122                              $512: ;102
451 002122  016704  000000G             MOV CURSOR,%4
452 002126                              $522: ; 104
453 002126  121427  000073              CMPB (%4),#73
454 002132  001413                      BEQ $722
455 002134  121427  000135              CMPB (%4),#135
456 002140  001410                      BEQ $722
457 002142  121427  000012              CMPB (%4),#12
458 002146  001405                      BEQ $722
459 002150  020467  000000G             CMP %4,PROGEND
460 002154  101002                      BHI $722
461 002156  005204                      INC %4
462 002160  000762                      BR $522
463 002162                              $722: ;122
464 002162  020467  000000G             CMP %4,PROGEND
465 002166  101405                      BLOS $132
466 002170  012716  000002              MOV #2,(%6)
467 002174  004767  000000G             JSR %7,ERROR
468 002200  000406                      BR $43
469 002202                              $132: ;132
470 002202  010400                      MOV %4,%0
471 002204  005200                      INC %0
472 002206  010067  000000G             MOV %0,CURSOR
473 002212                              $771: ; 136
474 002212  004767  000000G             JSR %7,REM
475 002216                              $43:
476 002216  000167  000000G             JMP RET
477 000106                              .PSECT DATA
478 000106                              $114:
479 000106     073     000              .BYTE 73,0
480 000110                              $504:
481 000110     142     162     145      .BYTE 142,162,145,141,153,0
    000113     141     153     000
482 000116                              $373:
483 000116     012     000              .BYTE 12,0
484 000120                              $173:
485 000120     073     000              .BYTE 73,0
486 000122                              $563:
487 000122     162     145     164      .BYTE 162,145,164,165,162,156,0
    000125     165     162     156
    000130     000
```

```
488 000131                              $753:
489 000131      073     000             .BYTE 73,0
490 000133                              $143:
491 000133      051     000             .BYTE 51,0
492 000135                              $733:
493 000135      050     000             .BYTE 50,0
494 000137                              $533:
495 000137      167     150     151     .BYTE 167,150,151,154,145,0
    000142      154     145     000
496 000145                              $723:
497 000145      145     154     163     .BYTE 145,154,163,145,0
    000150      145     000
498 000152                              $123:
499 000152      145     154     163     .BYTE 145,154,163,145,0
    000155      145     000
500 000157                              $313:
501 000157      051     000             .BYTE 51,0
502 000161                              $113:
503 000161      050     000             .BYTE 50,0
504 000163                              $703:
505 000163      151     146     000     .BYTE 151,146,0
506 000166                              $103:
507 000166      135     000             .BYTE 135,0
508 000170                              $372:
509 000170      133     000             .BYTE 133,0
510 000172                              $562:
511 000172      073     000             .BYTE 73,0
512 000174                              $362:
513 000174      054     000             .BYTE 54,0
514 000176                              $552:
515 000176      151     156     164     .BYTE 151,156,164,0
    000201      000
516 000202                              $742:
517 000202      073     000             .BYTE 73,0
518 000204                              $542:
519 000204      054     000             .BYTE 54,0
520 000206                              $732:
521 000206      143     150     141     .BYTE 143,150,141,162,0
    000211      162     000
```

```
522 002222                          .PSECT TEXT
523 002222                          ST:
524 002222  010546                  MOV %5,-(%6)
525 002224  010605                  MOV %6,%5
526 002226  062706  177764          ADD #-14,%6
527 002232  005767  000000G         TST ERR
528 002236  001402                  BEQ $332
529 002240  000167  000000G         JMP RETS
530 002244                          $332: ;7
531 002244  004767  000000G         JSR %7,REM
532 002250  004767  000000G         JSR %7,STBEGIN
533 002254  012716  000206'         MOV #$732,(%6)
534 002260  004767  000000G         JSR %7,LIT
535 002264  005700                  TST %0
536 002266  001423                  BEQ $532
537 002270  005016                  CLR (%6)
538 002272  012746  000103          MOV #103,-(%6)
539 002276  004767  001140          JSR %7,VALLOC
540 002302  005726                  TST (%6)+
541 002304                          $142: ; 27
542 002304  012716  000204'         MOV #$542,(%6)
543 002310  004767  000000G         JSR %7,LIT
544 002314  005700                  TST %0
545 002316  001440                  BEQ $342
546 002320  005016                  CLR (%6)
547 002322  012746  000103          MOV #103,-(%6)
548 002326  004767  001110          JSR %7,VALLOC
549 002332  005726                  TST (%6)+
550 002334  000763                  BR $142
551 002336                          $532: ;44
552 002336  012716  000176'         MOV #$552,(%6)
553 002342  004767  000000G         JSR %7,LIT
554 002346  005700                  TST %0
555 002350  001431                  BEQ $352
556 002352  005016                  CLR (%6)
557 002354  012746  000111          MOV #111,-(%6)
558 002360  004767  001056          JSR %7,VALLOC
559 002364  005726                  TST (%6)+
560 002366                          $752: ; 60
561 002366  012716  000174'         MOV #$362,(%6)
562 002372  004767  000000G         JSR %7,LIT
563 002376  005700                  TST %0
564 002400  001455                  BEQ $162
565 002402  005016                  CLR (%6)
566 002404  012746  000111          MOV #111,-(%6)
567 002410  004767  001026          JSR %7,VALLOC
568 002414  005726                  TST (%6)+
569 002416  000763                  BR $752
```

```
570 002420                          $342: ;75
571 002420  012716  000202'         MOV #$742,(%6)
572 002424  004767  000000G         JSR %7,LIT
573 002430  000167  000546          JMP $152
574 002434                          $352: ;103
575 002434  012716  000170'         MOV #$372,(%6)
576 002440  004767  000000G         JSR %7,LIT
577 002444  005700                  TST %0
578 002446  001440                  BEQ $172
579 002450                          $572: ; 111
580 002450  005767  000000G         TST LEAVE
581 002454  001402                  BEQ .+6
582 002456  000167  000520          JMP $152
583 002462  005767  000000G         TST BRAKE
584 002466  001402                  BEQ .+6
585 002470  000167  000506          JMP $152
586 002474  012716  000166'         MOV #$103,(%6)
587 002500  004767  000000G         JSR %7,LIT
588 002504  005700                  TST %0
589 002506  001402                  BEQ .+6
590 002510  000167  000466          JMP $152
591 002514  005767  000000G         TST ERR
592 002520  001402                  BEQ .+6
593 002522  000167  000454          JMP $152
594 002526  004767  177470          JSR %7,ST
595 002532  000746                  BR $572
596 002534                          $162: ;143
597 002534  012716  000172'         MOV #$562,(%6)
598 002540  004767  000000G         JSR %7,LIT
599 002544  000167  000432          JMP $152
600 002550                          $172: ;151
601 002550  012716  000163'         MOV #$703,(%6)
602 002554  004767  000000G         JSR %7,LIT
603 002560  005700                  TST %0
604 002562  001431                  BEQ $503
605 002564  012716  000161'         MOV #$113,(%6)
606 002570  004767  000000G         JSR %7,LIT
607 002574  004767  175200          JSR %7,ASGN
608 002600  012716  000157'         MOV #$313,(%6)
609 002604  004767  000000G         JSR %7,LIT
610 002610  004767  000000G         JSR %7,TOPTOI
611 002614  005700                  TST %0
612 002616  001465                  BEQ $513
613 002620  004767  177376          JSR %7,ST
614 002624  012716  000152'         MOV #$123,(%6)
615 002630  004767  000000G         JSR %7,LIT
616 002634  005700                  TST %0
617 002636  001561                  BEQ $152
618 002640  004767  177052          JSR %7,SKIPST
619 002644  000556                  BR $152
```

```
620 002646                          $503: ;210
621 002646  012716  000137'         MOV #$533,(%6)
622 002652  004767  000000G         JSR %7,LIT
623 002656  005700                  TST %0
624 002660  001457                  BEQ $333
625 002662  016700  000000G         MOV CURSOR,%0
626 002666  062700  177773          ADD #-5,%0
627 002672  010065  177770          MOV %0,-10(%5)
628 002676  012716  000135'         MOV #$733,(%6)
629 002702  004767  000000G         JSR %7,LIT
630 002706  004767  175066          JSR %7,ASGN
631 002712  012716  000133'         MOV #$143,(%6)
632 002716  004767  000000G         JSR %7,LIT
633 002722  004767  000000G         JSR %7,TOPTOI
634 002726  005700                  TST %0
635 002730  001442                  BEQ $343
636 002732  016765  000000G 177766  MOV CURSOR,-12(%5)
637 002740  004767  177256          JSR %7,ST
638 002744  005767  000000G         TST BRAKE
639 002750  001435                  BEQ $543
640 002752  016567  177766  000000G MOV -12(%5),CURSOR
641 002760  004767  176732          JSR %7,SKIPST
642 002764  005067  000000G         CLR BRAKE
643 002770  000504                  BR $152
644 002772                          $513: ;262
645 002772  004767  176720          JSR %7,SKIPST
646 002776  012716  000145'         MOV #$723,(%6)
647 003002  004767  000000G         JSR %7,LIT
648 003006  005700                  TST %0
649 003010  001474                  BEQ $152
650 003012  004767  177204          JSR %7,ST
651 003016  000471                  BR $152
652 003020                          $333: ;275
653 003020  012716  000131'         MOV #$753,(%6)
654 003024  004767  000000G         JSR %7,LIT
655 003030  005700                  TST %0
656 003032  001410                  BEQ $553
657 003034  000462                  BR $152
658 003036                          $343: ;304
659 003036  004767  176654          JSR %7,SKIPST
660 003042  000457                  BR $152
661 003044                          $543: ;307
662 003044  016567  177770  000000G MOV -10(%5),CURSOR
663 003052  000453                  BR $152
```

```
664 003054                              $553: ;313
665 003054  012716  000122'             MOV #$563,(%6)
666 003060  004767  000000G             JSR %7,LIT
667 003064  005700                      TST %0
668 003066  001415                      BEQ $363
669 003070  012716  000120'             MOV #$173,(%6)
670 003074  004767  000000G             JSR %7,LIT
671 003100  005700                      TST %0
672 003102  001021                      BNE $573
673 003104  012716  000116'             MOV #$373,(%6)
674 003110  004767  000000G             JSR %7,LIT
675 003114  005700                      TST %0
676 003116  001422                      BEQ $763
677 003120  000412                      BR $573
678 003122                              $363: ;336
679 003122  012716  000110'             MOV #$504,(%6)
680 003126  004767  000000G             JSR %7,LIT
681 003132  005700                      TST %0
682 003134  001416                      BEQ $304
683 003136  012767  000001  000000G     MOV #1,BRAKE
684 003144  000416                      BR $152
685 003146                              $573: ;350
686 003146  005016                      CLR (%6)
687 003150  004767  000000G             JSR %7,PUSHINT
688 003154                              $773: ; 353
689 003154  012767  000001  000000G     MOV #1,LEAVE
690 003162  000407                      BR $152
691 003164                              $763: ;357
692 003164  004767  174610              JSR %7,ASGN
693 003170  000771                      BR $773
694 003172                              $304: ;362
695 003172  004767  174602              JSR %7,ASGN
696 003176  004767  000000G             JSR %7,TOPTOI
697 003202                              $152: ; 366
698 003202  012716  000106'             MOV #$114,(%6)
699 003206  004767  000000G             JSR %7,LIT
700 003212  004767  000000G             JSR %7,REM
701 003216  000167  000000G             JMP RETS
702 000213                              .PSECT DATA
703 000213                              $534:
704 000213     045     000              .BYTE 45,0
705 000215                              $724:
706 000215     057     000              .BYTE 57,0
707 000217                              $124:
708 000217     052     000              .BYTE 52,0
709 003222                              .PSECT TEXT
710 003222                              TERM:
711 003222  004567  000000G             JSR %5,SAVE
712 003226  005746                      TST -(%6)
713 003230  004767  175010              JSR %7,FACTOR
```

```
714 003234                          $314: ; 4
715 003234  012716  000217'         MOV #$124,(%6)
716 003240  004767  000000G         JSR %7,LIT
717 003244  005700                  TST %0
718 003246  001417                  BEQ $714
719 003250  004767  000000G         JSR %7,TOPTOI
720 003254  010065  177770          MOV %0,-10(%5)
721 003260  004767  174760          JSR %7,FACTOR
722 003264  004767  000000G         JSR %7,TOPTOI
723 003270  010001                  MOV %0,%1
724 003272  070165  177770          MUL -10(%5),%1
725 003276  010116                  MOV %1,(%6)
726 003300  004767  000000G         JSR %7,PUSHINT
727 003304  000753                  BR $314
728 003306                          $714: ;31
729 003306  012716  000215'         MOV #$724,(%6)
730 003312  004767  000000G         JSR %7,LIT
731 003316  005700                  TST %0
732 003320  001420                  BEQ $524
733 003322  004767  000000G         JSR %7,TOPTOI
734 003326  010065  177770          MOV %0,-10(%5)
735 003332  004767  174706          JSR %7,FACTOR
736 003336  004767  000000G         JSR %7,TOPTOI
737 003342  016503  177770          MOV -10(%5),%3
738 003346  006702                  SXT %2
739 003350  071200                  DIV %0,%2
740 003352  010216                  MOV %2,(%6)
741 003354  004767  000000G         JSR %7,PUSHINT
742 003360  000725                  BR $314
743 003362                          $524: ;57
744 003362  012716  000213'         MOV #$534,(%6)
745 003366  004767  000000G         JSR %7,LIT
746 003372  005700                  TST %0
747 003374  001420                  BEQ $334
748 003376  004767  000000G         JSR %7,TOPTOI
749 003402  010065  177770          MOV %0,-10(%5)
750 003406  004767  174632          JSR %7,FACTOR
751 003412  004767  000000G         JSR %7,TOPTOI
752 003416  016503  177770          MOV -10(%5),%3
753 003422  006702                  SXT %2
754 003424  071200                  DIV %0,%2
755 003426  010316                  MOV %3,(%6)
756 003430  004767  000000G         JSR %7,PUSHINT
757 003434  000677                  BR $314
758 003436                          $334: ;105
759 003436  000167  000000G         JMP RET
760 000221                          .PSECT DATA
761 000221                          $744:
762 000221      051      000        .BYTE 51,0
763 000223                          $544:
764 000223      050      000        .BYTE 50,0
```

```
765                     .GLOBL NEWVAR
766                     .GLOBL VALLOC
767                     .GLOBL STBEGIN
768                     .GLOBL ST
769                     .GLOBL SKIP
770                     .GLOBL REM
771                     .GLOBL SKIPST
772                     .GLOBL ENTER
773                     .GLOBL ADDRVAL
774                     .GLOBL SYMNAME
775                     .GLOBL PUSH
776                     .GLOBL ATOI
777                     .GLOBL CONST
778                     .GLOBL ERROR
779                     .GLOBL FACTOR
780                     .GLOBL TERM
781                     .GLOBL PUSHINT
782                     .GLOBL TOPTOI
783                     .GLOBL EXPR
784                     .GLOBL EQ
785                     .GLOBL LIT
786                     .GLOBL RELN
787                     .GLOBL ASGN
788                     .GLOBL CURGLOB
789                     .GLOBL CURFUN
790                     .GLOBL FUNB
791                     .GLOBL NXTVAR
792                     .GLOBL VARB
793                     .GLOBL TOP
794                     .GLOBL STACKB
795                     .GLOBL APPLVL
796                     .GLOBL ERR
797                     .GLOBL BRAKE
798                     .GLOBL LEAVE
799                     .GLOBL PRUSED
800                     .GLOBL PROGEND
801                     .GLOBL ERRAT
802                     .GLOBL LNAME
803                     .GLOBL FNAME
804                     .GLOBL CURSOR
805                     .GLOBL PR
```

```
806 003442                          .PSECT TEXT
807 003442                          VALLOC:
808 003442  010546                  MOV %5,-(%6)
809 003444  010605                  MOV %6,%5
810 003446  062706  177746          ADD #-32,%6
811 003452  004767  000000G         JSR %7,SYMNAME
812 003456  005700                  TST %0
813 003460  001443                  BEQ $144
814 003462  105065  177760          CLRB -20(%5)
815 003466  012716  000223'        MOV #$544,(%6)
816 003472  004767  000000G         JSR %7,LIT
817 003476  005700                  TST %0
818 003500  001441                  BEQ $344
819 003502  016765  000000G 177770  MOV FNAME,-10(%5)
820 003510  016765  000000G 177766  MOV LNAME,-12(%5)
821 003516  112765  000001  177760  MOVB #1,-20(%5)
822 003524  004767  174250          JSR %7,ASGN
823 003530  016567  177770  000000G MOV -10(%5),FNAME
824 003536  016567  177766  000000G MOV -12(%5),LNAME
825 003544  012716  000221'        MOV #$744,(%6)
826 003550  004767  000000G         JSR %7,LIT
827 003554  004767  000000G         JSR %7,TOPTOI
828 003560  005200                  INC %0
829 003562  010065  177762          MOV %0,-16(%5)
830 003566  000411                  BR $154
831 003570                          $144: ;51
832 003570  012716  000003          MOV #3,(%6)
833 003574  004767  000000G         JSR %7,ERROR
834 003600                          $354: ; 55
835 003600  000167  000000G         JMP RETS
836 003604                          $344: ;57
837 003604  012765  000001  177762  MOV #1,-16(%5)
838 003612                          $154: ; 62
839 003612  026527  000004  000103  CMP 4(%5),#103
840 003620  001003                  BNE $63
841 003622  112700  000001          MOVB #1,%0
842 003626  000402                  BR $04
843 003630                          $63:
844 003630  112700  000002          MOVB #2,%0
845 003634                          $04:
846 003634  110065  177761          MOVB %0,-17(%5)
847 003640  016565  000006  177764  MOV 6(%5),-14(%5)
848 003646  010516                  MOV %5,(%6)
849 003650  062716  177750          ADD #-30,(%6)
850 003654  004767  000000G         JSR %7,NEWVAR
851 003660  000747                  BR $354
852         000001                  .END
```

```
                                        .TITLE TINY-C/11-01-03/ENTER
                                        .DSABL REG
                                        .ENABL GBL
000000                                  .PSECT DATA
000000                                  $56:
000000      073     000                 .BYTE 73,0
000002                                  $36:
000002      054     000                 .BYTE 54,0
000004                                  $55:
000004      143     150     141         .BYTE 143,150,141,162,0
000007      162     000
000011                                  $74:
000011      073     000                 .BYTE 73,0
000013                                  $54:
000013      054     000                 .BYTE 54,0
000015                                  $73:
000015      151     156     164         .BYTE 151,156,164,0
000020      000
000021                                  $52:
000021      051     000                 .BYTE 51,0
000023                                  $51:
000023      054     000                 .BYTE 54,0
000025                                  $5:
000025      051     000                 .BYTE 51,0
000027                                  $1:
000027      050     000                 .BYTE 50,0
000000                                  .PSECT TEXT
000000                                  ENTER:
000000  010546                          MOV %5,-(%6)
000002  010605                          MOV %6,%5
000004  062706  177762                  ADD #-16,%6
000010  005065  177766                  CLR -12(%5)
000014  016700  000000G                 MOV TOP,%0
000020  062700  000006                  ADD #6,%0
000024  010065  177764                  MOV %0,-14(%5)
000030  012716  000027'                 MOV #$1,(%6)
000034  004767  000000G                 JSR %7,LIT
000040  012716  000025'                 MOV #$5,(%6)
000044  004767  000000G                 JSR %7,LIT
000050  005700                          TST %0
000052  001042                          BNE $3
000054  127727  000000G 000073         CMPB @CURSOR,#73
000062  001436                          BEQ $3
000064  127727  000000G 000012         CMPB @CURSOR,#12
000072  001432                          BEQ $3
000074  127727  000000G 000057         CMPB @CURSOR,#57
000102  001426                          BEQ $3
```

```
 46 000104                          $7: ; 40
 47 000104  004767  000000G         JSR %7,ASGN
 48 000110  005265  177766          INC -12(%5)
 49 000114  012716  000023'         MOV #$51,(%6)
 50 000120  004767  000000G         JSR %7,LIT
 51 000124  005700                  TST %0
 52 000126  001366                  BNE $7
 53 000130  005767  000000G         TST ERR
 54 000134  001421                  BEQ $71
 55 000136                          $12: ; 55
 56 000136  016500  177766          MOV -12(%5),%0
 57 000142  005365  177766          DEC -12(%5)
 58 000146  005700                  TST %0
 59 000150  001420                  BEQ $32
 60 000152  004767  000000G         JSR %7,POP
 61 000156  000767                  BR $12
 62 000160                          $3: ;66
 63 000160  005765  000004          TST 4(%5)
 64 000164  001017                  BNE $72
 65 000166  016516  177766          MOV -12(%5),(%6)
 66 000172  004767  000000G         JSR %7,MC
 67 000176  000410                  BR $2
 68 000200                          $71: ;76
 69 000200  012716  000021'         MOV #$52,(%6)
 70 000204  004767  000000G         JSR %7,LIT
 71 000210  000763                  BR $3
 72 000212                          $32: ;103
 73 000212  005016                  CLR (%6)
 74 000214  004767  000000G         JSR %7,PUSHINT
 75 000220                          $2:
 76 000220  000167  000000G         JMP RETS
 77 000224                          $72: ;110
 78 000224  016765  000000G 177770  MOV CURSOR,-10(%5)
 79 000232  016567  000004  000000G MOV 4(%5),CURSOR
 80 000240  004767  000000G         JSR %7,NEWFUN
 81 000244                          $13: ; 120
 82 000244  004767  000000G         JSR %7,REM
 83 000250  012716  000015'         MOV #$73,(%6)
 84 000254  004767  000000G         JSR %7,LIT
 85 000260  005700                  TST %0
 86 000262  001433                  BEQ $53
 87 000264  016516  177764          MOV -14(%5),(%6)
 88 000270  062765  000006  177764  ADD #6,-14(%5)
 89 000276  012746  000111          MOV #111,-(%6)
 90 000302  004767  001012          JSR %7,SETARG
 91 000306  005726                  TST (%6)+
```

```
 92 000310                          $14: ; 142
 93 000310  012716  000013'         MOV #$54,(%6)
 94 000314  004767  000000G         JSR %7,LIT
 95 000320  005700                  TST %0
 96 000322  001454                  BEQ $34
 97 000324  016516  177764          MOV -14(%5),(%6)
 98 000330  062765  000006  177764  ADD #6,-14(%5)
 99 000336  012746  000111          MOV #111,-(%6)
100 000342  004767  000752          JSR %7,SETARG
101 000346  005726                  TST (%6)+
102 000350  000757                  BR $14
103 000352                          $53: ;163
104 000352  012716  000004'         MOV #$55,(%6)
105 000356  004767  000000G         JSR %7,LIT
106 000362  005700                  TST %0
107 000364  001445                  BEQ $33
108 000366  016516  177764          MOV -14(%5),(%6)
109 000372  062765  000006  177764  ADD #6,-14(%5)
110 000400  012746  000103          MOV #103,-(%6)
111 000404  004767  000710          JSR %7,SETARG
112 000410  005726                  TST (%6)+
113 000412                          $75: ; 203
114 000412  012716  000002'         MOV #$36,(%6)
115 000416  004767  000000G         JSR %7,LIT
116 000422  005700                  TST %0
117 000424  001420                  BEQ $16
118 000426  016516  177764          MOV -14(%5),(%6)
119 000432  062765  000006  177764  ADD #6,-14(%5)
120 000440  012746  000103          MOV #103,-(%6)
121 000444  004767  000650          JSR %7,SETARG
122 000450  005726                  TST (%6)+
123 000452  000757                  BR $75
124 000454                          $34: ;224
125 000454  012716  000011'         MOV #$74,(%6)
126 000460  004767  000000G         JSR %7,LIT
127 000464  000667                  BR $13
128 000466                          $16: ;231
129 000466  012716  000000'         MOV #$56,(%6)
130 000472  004767  000000G         JSR %7,LIT
131 000476  000662                  BR $13
132 000500                          $33: ;236
133 000500  016700  000000G         MOV TOP,%0
134 000504  062700  000006          ADD #6,%0
135 000510  020065  177764          CMP %0,-14(%5)
136 000514  001404                  BEQ $37
137 000516  012716  000025          MOV #25,(%6)
138 000522  004767  000000G         JSR %7,ERROR
```

```
139 000526                              $37: ;251
140 000526  016500  177766              MOV -12(%5),%0
141 000532  005365  177766              DEC -12(%5)
142 000536  005700                      TST %0
143 000540  001403                      BEQ $57
144 000542  004767  000000G             JSR %7,POP
145 000546  000767                      BR $37
146 000550                              $57: ;262
147 000550  005767  000000G             TST ERR
148 000554  001002                      BNE $77
149 000556  004767  000000G             JSR %7,ST
150 000562                              $77: ;267
151 000562  005767  000000G             TST LEAVE
152 000566  001004                      BNE $101
153 000570  005016                      CLR (%6)
154 000572  004767  000000G             JSR %7,PUSHINT
155 000576  000402                      BR $301
156 000600                              $101: ;276
157 000600  005067  000000G             CLR LEAVE
158 000604                              $301: ; 300
159 000604  016567  177770  000000G     MOV -10(%5),CURSOR
160 000612  004767  000000G             JSR %7,FUNDONE
161 000616  000600                      BR $2
162 000031                              .PSECT DATA
163 000031                              $551:
164 000031     145     156     144      .BYTE 145,156,144,154,151,142,162,141
    000034     154     151     142
    000037     162     141
165 000041     162     171     000      .BYTE 162,171,0
166 000044                              $741:
167 000044     073     000              .BYTE 73,0
168 000046                              $541:
169 000046     054     000              .BYTE 54,0
170 000050                              $731:
171 000050     143     150     141      .BYTE 143,150,141,162,0
    000053     162     000
172 000055                              $131:
173 000055     073     000              .BYTE 73,0
174 000057                              $721:
175 000057     054     000              .BYTE 54,0
176 000061                              $121:
177 000061     151     156     164      .BYTE 151,156,164,0
    000064     000
178 000065                              $311:
179 000065     133     000              .BYTE 133,0
```

```
180 000620                          .PSECT TEXT
181 000620                          LINK:
182 000620  010546                  MOV %5,-(%6)
183 000622  010605                  MOV %6,%5
184 000624  062706  177752          ADD #-26,%6
185 000630  004767  000000G         JSR %7,NEWFUN
186 000634                          $501: ; 4
187 000634  016700  000000G         MOV PROGEND,%0
188 000640  005300                  DEC %0
189 000642  026700  000000G         CMP CURSOR,%0
190 000646  103023                  BHIS $701
191 000650  005767  000000G         TST ERR
192 000654  001020                  BNE $701
193 000656  004767  000000G         JSR %7,REM
194 000662  012716  000065'         MOV #$311,(%6)
195 000666  004767  000000G         JSR %7,LIT
196 000672  005700                  TST %0
197 000674  001411                  BEQ $111
198 000676  012716  000135          MOV #135,(%6)
199 000702  012746  000133          MOV #133,-(%6)
200 000706  004767  000000G         JSR %7,SKIP
201 000712  005726                  TST (%6)+
202 000714  000747                  BR $501
203 000716                          $701: ;35
204 000716  000563                  BR $4
205 000720                          $111: ;36
206 000720  012716  000061'         MOV #$121,(%6)
207 000724  004767  000000G         JSR %7,LIT
208 000730  005700                  TST %0
209 000732  001423                  BEQ $711
210 000734  005016                  CLR (%6)
211 000736  012746  000111          MOV #111,-(%6)
212 000742  004767  000000G         JSR %7,VALLOC
213 000746  005726                  TST (%6)+
214 000750                          $321: ; 52
215 000750  012716  000057'         MOV #$721,(%6)
216 000754  004767  000000G         JSR %7,LIT
217 000760  005700                  TST %0
218 000762  001440                  BEQ $521
219 000764  005016                  CLR (%6)
220 000766  012746  000111          MOV #111,-(%6)
221 000772  004767  000000G         JSR %7,VALLOC
222 000776  005726                  TST (%6)+
223 001000  000763                  BR $321
```

```
224 001002                              $711: ;67
225 001002  012716  000050'             MOV #$731,(%6)
226 001006  004767  000000G             JSR %7,LIT
227 001012  005700                      TST %0
228 001014  001430                      BEQ $531
229 001016  005016                      CLR (%6)
230 001020  012746  000103              MOV #103,-(%6)
231 001024  004767  000000G             JSR %7,VALLOC
232 001030  005726                      TST (%6)+
233 001032                              $141: ; 103
234 001032  012716  000046'             MOV #$541,(%6)
235 001036  004767  000000G             JSR %7,LIT
236 001042  005700                      TST %0
237 001044  001425                      BEQ $341
238 001046  005016                      CLR (%6)
239 001050  012746  000103              MOV #103,-(%6)
240 001054  004767  000000G             JSR %7,VALLOC
241 001060  005726                      TST (%6)+
242 001062  000763                      BR $141
243 001064                              $521: ;120
244 001064  012716  000055'             MOV #$131,(%6)
245 001070  004767  000000G             JSR %7,LIT
246 001074  000657                      BR $501
247 001076                              $531: ;125
248 001076  012716  000031'             MOV #$551,(%6)
249 001102  004767  000000G             JSR %7,LIT
250 001106  005700                      TST %0
251 001110  001410                      BEQ $351
252 001112  004767  000000G             JSR %7,NEWFUN
253 001116  000646                      BR $501
254 001120                              $341: ;136
255 001120  012716  000044'             MOV #$741,(%6)
256 001124  004767  000000G             JSR %7,LIT
257 001130  000641                      BR $501
258 001132                              $351: ;143
259 001132  004767  000000G             JSR %7,SYMNAME
260 001136  005700                      TST %0
261 001140  001434                      BEQ $161
262 001142  112765  000105  177764      MOVB #105,-14(%5)
263 001150  112765  000002  177765      MOVB #2,-13(%5)
264 001156  012765  000001  177766      MOV #1,-12(%5)
265 001164  016765  000000G 177770      MOV CURSOR,-10(%5)
266 001172  010516                      MOV %5,(%6)
267 001174  062716  177754              ADD #-24,(%6)
268 001200  004767  000000G             JSR %7,NEWVAR
```

```
269 001204                              $361: ; 170
270 001204  127727  000000G 000133      CMPB @CURSOR,#133
271 001212  001415                      BEQ $561
272 001214  026767  000000G 000000G     CMP CURSOR,PROGEND
273 001222  103011                      BHIS $561
274 001224  005267  000000G             INC CURSOR
275 001230  000765                      BR $361
276 001232                              $161: ;203
277 001232  012716  000024              MOV #24,(%6)
278 001236  004767  000000G             JSR %7,ERROR
279 001242  000167  177366              JMP $501
280 001246                              $561: ;211
281 001246  127727  000000G 000133      CMPB @CURSOR,#133
282 001254  001406                      BEQ $761
283 001256  012716  000026              MOV #26,(%6)
284 001262  004767  000000G             JSR %7,ERROR
285 001266                              $4:
286 001266  000167  000000G             JMP RETS
287 001272                              $761: ;223
288 001272  005267  000000G             INC CURSOR
289 001276  012716  000135              MOV #135,(%6)
290 001302  012746  000133              MOV #133,-(%6)
291 001306  004767  000000G             JSR %7,SKIP
292 001312  005726                      TST (%6)+
293 001314  000167  177314              JMP $501
294                                     .GLOBL SKIP
295                                     .GLOBL VALLOC
296                                     .GLOBL SYMNAME
297                                     .GLOBL NEWVAR
298                                     .GLOBL LINK
299                                     .GLOBL FUNDONE
300                                     .GLOBL ST
301                                     .GLOBL ERROR
302                                     .GLOBL REM
303                                     .GLOBL SETARG
304                                     .GLOBL NEWFUN
305                                     .GLOBL MC
306                                     .GLOBL ASGN
307                                     .GLOBL PUSHINT
308                                     .GLOBL POP
309                                     .GLOBL LIT
310                                     .GLOBL ENTER
311                                     .GLOBL CURGLOB
312                                     .GLOBL CURFUN
313                                     .GLOBL FUNB
314                                     .GLOBL NXTVAR
315                                     .GLOBL VARB
316                                     .GLOBL TOP
317                                     .GLOBL STACKB
318                                     .GLOBL APPLVL
```

```
                                        .GLOBL ERR
                                        .GLOBL BRAKE
                                        .GLOBL LEAVE
                                        .GLOBL PRUSED
                                        .GLOBL PROGEND
                                        .GLOBL ERRAT
                                        .GLOBL LNAME
                                        .GLOBL FNAME
                                        .GLOBL CURSOR
                                        .GLOBL PR
001320                                  SETARG:
001320  004567  000000G                 JSR %5,SAVE
001324  016500  000006                  MOV 6(%5),%0
001330  126027  000001  000101          CMPB 1(%0),#101
001336  001012                          BNE $571
001340  016500  000006                  MOV 6(%5),%0
001344  016016  000004                  MOV 4(%0),(%6)
001350  016546  000004                  MOV 4(%5),-(%6)
001354  004767  000000G                 JSR %7,VALLOC
001360  005726                          TST (%6)+
001362  000453                          BR $771
001364                                  $571: ;22
001364  016500  000006                  MOV 6(%5),%0
001370  126027  000002  000001          CMPB 2(%0),#1
001376  001016                          BNE $102
001400  105775  000006                  TSTB @6(%5)
001404  001013                          BNE $102
001406  016500  000006                  MOV 6(%5),%0
001412  117000  000004                  MOVB @4(%0),%0
001416  010016                          MOV %0,(%6)
001420  016546  000004                  MOV 4(%5),-(%6)
001424  004767  000000G                 JSR %7,VALLOC
001430  005726                          TST (%6)+
001432  000427                          BR $771
001434                                  $102: ;46
001434  016500  000006                  MOV 6(%5),%0
001440  016000  000004                  MOV 4(%0),%0
001444  116004  000001                  MOVB 1(%0),%4
001450  010400                          MOV %4,%0
001452  072027  000010                  ASH #10,%0
001456  016501  000006                  MOV 6(%5),%1
001462  117101  000004                  MOVB @4(%1),%1
001466  042701  177400                  BIC #-400,%1
001472  050100                          BIS %1,%0
001474  010004                          MOV %0,%4
001476  010416                          MOV %4,(%6)
001500  016546  000004                  MOV 4(%5),-(%6)
001504  004767  000000G                 JSR %7,VALLOC
001510  005726                          TST (%6)+
001512                                  $771: ; 75
001512  000167  000000G                 JMP RET
        000001                          .END
```

```
                                .TITLE TINY-C/11-01-03/SCAN
                                .DSABL REG
                                .ENABL GBL
000000                          .PSECT TEXT
000000                          BLANKS:
000000  016700  000000G         MOV CURSOR,%0
000004                          $1: ; 4
000004  121027  000040          CMPB (%0),#40
000010  001406                  BEQ $5
000012  121027  000011          CMPB (%0),#11
000016  001403                  BEQ $5
000020  010067  000000G         MOV %0,CURSOR
000024  000207                  RTS %7
000026                          $5: ;12
000026  005200                  INC %0
000030  000765                  BR $1
000032                          CONST:
000032  004567  000000G         JSR %5,SAVE
000036  004767  177736          JSR %7,BLANKS
000042  010004                  MOV %0,%4
000044  121427  000053          CMPB (%4),#53
000050  001422                  BEQ $11
000052  121427  000055          CMPB (%4),#55
000056  001417                  BEQ $11
000060  122714  000060          CMPB #60,(%4)
000064  003007                  BGT $2
000066  121427  000071          CMPB (%4),#71
000072  003004                  BGT $2
000074  111400                  MOVB (%4),%0
000076  062700  177720          ADD #-60,%0
000102  000402                  BR $4
000104                          $2:
000104  112700  177777          MOVB #-1,%0
000110                          $4:
000110  120027  177777          CMPB %0,#-1
000114  001424                  BEQ $7
000116                          $11: ;32
000116  016767  000000G 000000G MOV CURSOR,FNAME
000124                          $31: ; 35
000124  122714  000060          CMPB #60,(%4)
000130  003007                  BGT $6
000132  121427  000071          CMPB (%4),#71
000136  003004                  BGT $6
000140  111400                  MOVB (%4),%0
000142  062700  177720          ADD #-60,%0
000146  000402                  BR $01
```

```
47 000150                              $6:
48 000150  112700  177777              MOVB #-1,%0
49 000154                              $01:
50 000154  120027  177777              CMPB %0,#-1
51 000160  001423                      BEQ $51
52 000162  005204                      INC %4
53 000164  000757                      BR $31
54 000166                              $7: ;56
55 000166  121427  000042              CMPB (%4),#42
56 000172  001027                      BNE $12
57 000174  005204                      INC %4
58 000176  010467  000000G             MOV %4,FNAME
59 000202                              $32: ; 64
60 000202  105714                      TSTB (%4)
61 000204  001441                      BEQ $52
62 000206  121427  000042              CMPB (%4),#42
63 000212  001436                      BEQ $52
64 000214  010400                      MOV %4,%0
65 000216  005204                      INC %4
66 000220  020067  000000G             CMP %0,PROGEND
67 000224  103031                      BHIS $52
68 000226  000765                      BR $32
69 000230                              $51: ;77
70 000230  010400                      MOV %4,%0
71 000232  005300                      DEC %0
72 000234  010067  000000G             MOV %0,LNAME
73 000240  010467  000000G             MOV %4,CURSOR
74 000244  112700  000001              MOVB #1,%0
75 000250  000434                      BR $21
76 000252                              $12: ;110
77 000252  121427  000047              CMPB (%4),#47
78 000256  001033                      BNE $33
79 000260  005204                      INC %4
80 000262  010467  000000G             MOV %4,FNAME
81 000266                              $53: ; 116
82 000266  121427  000047              CMPB (%4),#47
83 000272  001427                      BEQ $73
84 000274  010400                      MOV %4,%0
85 000276  005204                      INC %4
86 000300  020067  000000G             CMP %0,PROGEND
87 000304  103022                      BHIS $73
88 000306  000767                      BR $53
89 000310                              $52: ;127
90 000310  020467  000000G             CMP %4,PROGEND
91 000314  101041                      BHI $71
92 000316  105024                      CLRB (%4)+
93 000320  010467  000000G             MOV %4,CURSOR
94 000324  010400                      MOV %4,%0
95 000326  062700  177776              ADD #-2,%0
96 000332  010067  000000G             MOV %0,LNAME
97 000336  112700  000002              MOVB #2,%0
```

```
 98 000342                          $21:
 99 000342  000167  000000G         JMP RET
100 000346                          $33: ;146
101 000346  005000                  CLR %0
102 000350  000774                  BR $21
103 000352                          $73: ;150
104 000352  020467  000000G         CMP %4,PROGEND
105 000356  101013                  BHI $14
106 000360  005204                  INC %4
107 000362  010467  000000G         MOV %4,CURSOR
108 000366  010400                  MOV %4,%0
109 000370  062700  177776          ADD #-2,%0
110 000374  010067  000000G         MOV %0,LNAME
111 000400  112700  000003          MOVB #3,%0
112 000404  000756                  BR $21
113 000406                          $14: ;166
114 000406  012716  000002          MOV #2,(%6)
115 000412  004767  000000G         JSR %7,ERROR
116 000416  000751                  BR $21
117 000420                          $71: ;173
118 000420  000750                  BR $21
119 000422                          LIT:
120 000422  016700  000000G         MOV CURSOR,%0
121 000426  121027  000040          CMPB (%0),#40
122 000432  001403                  BEQ $15
123 000434  121027  000011          CMPB (%0),#11
124 000440  001002                  BNE $74
125 000442                          $15: ;14
126 000442  004767  177332          JSR %7,BLANKS
127 000446                          $74: ; 17
128 000446  016601  000002          MOV 2(%6),%1
129 000452                          $55:
130 000452  122021                  CMPB (%0)+,(%1)+
131 000454  001402                  BEQ $41
132 000456  005000                  CLR %0
133 000460  000207                  RTS %7
134 000462                          $41:
135 000462  105711                  TSTB (%1)
136 000464  001372                  BNE $55
137 000466  010067  000000G         MOV %0,CURSOR
138 000472  112700  000001          MOVB #1,%0
139 000476  000207                  RTS %7
140 000500                          REM:
141 000500  004567  000000G         JSR %5,SAVE
142 000504                          $36: ; 2
143 000504  004767  177270          JSR %7,BLANKS
144 000510  010004                  MOV %0,%4
145 000512  020467  000000G         CMP %4,PROGEND
146 000516  101005                  BHI $56
147 000520  121427  000012          CMPB (%4),#12
148 000524  001005                  BNE $76
149 000526  005204                  INC %4
150 000530  000413                  BR $17
```

```
151 000532                              $56: ;15
152 000532  016700  000000G             MOV PROGEND,%0
153 000536  000417                      BR $61
154 000540                              $76: ;20
155 000540  121427  000057              CMPB (%4),#57
156 000544  001010                      BNE $57
157 000546  126427  000001  000052      CMPB 1(%4),#52
158 000554  001412                      BEQ $101
159 000556  000403                      BR $57
160 000560                              $17: ;30
161 000560  010467  000000G             MOV %4,CURSOR
162 000564  000747                      BR $36
163 000566                              $57: ;33
164 000566  010467  000000G             MOV %4,CURSOR
165 000572  016700  000000G             MOV CURSOR,%0
166 000576                              $61:
167 000576  000167  000000G             JMP RET
168 000602                              $101: ;41
169 000602  122427  000012              CMPB (%4)+,#12
170 000606  001764                      BEQ $17
171 000610  020467  000000G             CMP %4,PROGEND
172 000614  101361                      BHI $17
173 000616  000771                      BR $101
174 000620                              SKIP:
175 000620  004567  000000G             JSR %5,SAVE
176 000624  112702  000001              MOVB #1,%2
177 000630  016704  000000G             MOV CURSOR,%4
178 000634                              $501: ; 6
179 000634  020467  000000G             CMP %4,PROGEND
180 000640  101006                      BHI $701
181 000642  111400                      MOVB (%4),%0
182 000644  020065  000004              CMP %0,4(%5)
183 000650  001011                      BNE $511
184 000652  005202                      INC %2
185 000654  000414                      BR $711
186 000656                              $701: ;17
187 000656  012716  000002              MOV #2,(%6)
188 000662  004767  000000G             JSR %7,ERROR
189 000666  000416                      BR $02
190 000670                              $111: ;24
191 000670  005204                      INC %4
192 000672  000760                      BR $501
193 000674                              $511: ;26
194 000674  111400                      MOVB (%4),%0
195 000676  020065  000006              CMP %0,6(%5)
196 000702  001001                      BNE $711
197 000704  005302                      DEC %2
```

```
198 000706                          $711: ;33
199 000706  005702                  TST %2
200 000710  001367                  BNE $111
201 000712  005204                  INC %4
202 000714  010467  000000G         MOV %4,CURSOR
203 000720  016700  000000G         MOV CURSOR,%0
204 000724                          $02:
205 000724  000167  000000G         JMP RET
206                                 .GLOBL SYMNAME
207                                 .GLOBL SKIP
208                                 .GLOBL REM
209                                 .GLOBL LIT
210                                 .GLOBL ERROR
211                                 .GLOBL CONST
212                                 .GLOBL BLANKS
213                                 .GLOBL CURGLOB
214                                 .GLOBL CURFUN
215                                 .GLOBL FUNB
216                                 .GLOBL NXTVAR
217                                 .GLOBL VARB
218                                 .GLOBL TOP
219                                 .GLOBL STACKB
220                                 .GLOBL APPLVL
221                                 .GLOBL ERR
222                                 .GLOBL BRAKE
223                                 .GLOBL LEAVE
224                                 .GLOBL ERRAT
225                                 .GLOBL LNAME
226                                 .GLOBL FNAME
227                                 .GLOBL CURSOR
228                                 .GLOBL PRUSED
229                                 .GLOBL PROGEND
230                                 .GLOBL PR
```

```
231 000730                          SYMNAME:
232 000730  004567  000000G         JSR %5,SAVE
233 000734  004767  177040          JSR %7,BLANKS
234 000740  010004                  MOV %0,%4
235 000742  122714  000141          CMPB #141,(%4)
236 000746  003003                  BGT $721
237 000750  121427  000172          CMPB (%4),#172
238 000754  003406                  BLE $521
239 000756                          $721: ;13
240 000756  122714  000101          CMPB #101,(%4)
241 000762  003016                  BGT $131
242 000764  121427  000132          CMPB (%4),#132
243 000770  003013                  BGT $131
244 000772                          $521: ;21
245 000772  010400                  MOV %4,%0
246 000774  005204                  INC %4
247 000776  010067  000000G         MOV %0,FNAME
248 001002                          $331: ; 25
249 001002  122714  000141          CMPB #141,(%4)
250 001006  003011                  BGT $731
251 001010  121427  000172          CMPB (%4),#172
252 001014  003414                  BLE $141
253 001016  000405                  BR $731
254 001020                          $131: ;34
255 001020  121427  000137          CMPB (%4),#137
256 001024  001762                  BEQ $521
257 001026  005000                  CLR %0
258 001030  000431                  BR $22
259 001032                          $731: ;41
260 001032  122714  000101          CMPB #101,(%4)
261 001036  003005                  BGT $341
262 001040  121427  000132          CMPB (%4),#132
263 001044  003002                  BGT $341
264 001046                          $141: ;47
265 001046  005204                  INC %4
266 001050  000754                  BR $331
267 001052                          $341: ;51
268 001052  121427  000137          CMPB (%4),#137
269 001056  001773                  BEQ $141
270 001060  122714  000060          CMPB #60,(%4)
271 001064  003003                  BGT $531
272 001066  121427  000071          CMPB (%4),#71
273 001072  003765                  BLE $141
274 001074                          $531: ;62
275 001074  010400                  MOV %4,%0
276 001076  005300                  DEC %0
277 001100  010067  000000G         MOV %0,LNAME
278 001104  010467  000000G         MOV %4,CURSOR
279 001110  112700  000001          MOVB #1,%0
280 001114                          $22:
281 001114  000167  000000G         JMP RET
282         000001                  .END
```

```
                                        .TITLE TINY-C/11-01-03/STACK
                                        .DSABL REG
                                        .ENABL GBL
000000                                  .PSECT TEXT
000000                                  EQ:
000000  010546                         MOV %5,-(%6)
000002  010605                         MOV %6,%5
000004  062706  177766                 ADD #-12,%6
000010  004767  000466                 JSR %7,TOPTOI
000014  010065  177770                 MOV %0,-10(%5)
000020  016700  000000G                MOV TOP,%0
000024  126027  000001  000114         CMPB 1(%0),#114
000032  001023                         BNE $1
000034  105777  000000G                TSTB @TOP
000040  001026                         BNE $3
000042  016700  000000G                MOV TOP,%0
000046  116000  000002                 MOVB 2(%0),%0
000052  010016                         MOV %0,(%6)
000054  012746  177770                 MOV #-10,-(%6)
000060  060516                         ADD %5,(%6)
000062  016700  000000G                MOV TOP,%0
000066  016046  000004                 MOV 4(%0),-(%6)
000072  004767  000000G                JSR %7,MOVN
000076  022626                         CMP (%6)+,(%6)+
000100  000422                         BR $5
000102                                  $1: ;37
000102  012716  000016                 MOV #16,(%6)
000106  004767  000000G                JSR %7,ERROR
000112                                  $7: ; 43
000112  000167  000000G                JMP RETS
000116                                  $3: ;45
000116  012716  000002                 MOV #2,(%6)
000122  012746  177770                 MOV #-10,-(%6)
000126  060516                         ADD %5,(%6)
000130  016700  000000G                MOV TOP,%0
000134  016046  000004                 MOV 4(%0),-(%6)
000140  004767  000000G                JSR %7,MOVN
000144  022626                         CMP (%6)+,(%6)+
000146                                  $5: ; 61
000146  004767  000012                 JSR %7,POP
000152  016516  177770                 MOV -10(%5),(%6)
000156  004767  000214                 JSR %7,PUSHINT
000162  000753                         BR $7
```

```
 44 000164                              .  POP:
 45 000164  010546                         MOV %5,-(%6)
 46 000166  010605                         MOV %6,%5
 47 000170  005746                         TST -(%6)
 48 000172  026727  000000G 000000G       CMP TOP,#STACKB
 49 000200  103410                         BLO $11
 50 000202  016700  000000G               MOV TOP,%0
 51 000206  162767  000006  000000G       SUB #6,TOP
 52 000214  016000  000004               MOV 4(%0),%0
 53 000220  000404                         BR $2
 54 000222                                 $11: ;16
 55 000222  012716  000017               MOV #17,(%6)
 56 000226  004767  000000G               JSR %7,ERROR
 57 000232                                 $2:
 58 000232  000167  000000G               JMP RETS
 59 000236                                 PUSH:
 60 000236  010546                         MOV %5,-(%6)
 61 000240  010605                         MOV %6,%5
 62 000242  005746                         TST -(%6)
 63 000244  062767  000006  000000G       ADD #6,TOP
 64 000252  026727  000000G 000264G       CMP TOP,#STACKB+264
 65 000260  101032                         BHI $51
 66 000262  116577  000004  000000G       MOVB 4(%5),@TOP
 67 000270  016700  000000G               MOV TOP,%0
 68 000274  116560  000006  000001       MOVB 6(%5),1(%0)
 69 000302  016700  000000G               MOV TOP,%0
 70 000306  116560  000010  000002       MOVB 10(%5),2(%0)
 71 000314  026527  000004  000101       CMP 4(%5),#101
 72 000322  001017                         BNE $71
 73 000324  005765  000006               TST 6(%5)
 74 000330  001014                         BNE $71
 75 000332  016700  000000G               MOV TOP,%0
 76 000336  017560  000012  000004       MOV @12(%5),4(%0)
 77 000344  000404                         BR $32
 78 000346                                 $51: ;43
 79 000346  012716  000020               MOV #20,(%6)
 80 000352  004767  000000G               JSR %7,ERROR
 81 000356                                 $32: ; 47
 82 000356  000167  000000G               JMP RETS
 83 000362                                 $71: ;51
 84 000362  016700  000000G               MOV TOP,%0
 85 000366  017560  000012  000004       MOV @12(%5),4(%0)
 86 000374  000770                         BR $32
```

```
 87 000376                          PUSHINT:
 88 000376  010546                  MOV %5,-(%6)
 89 000400  010605                  MOV %6,%5
 90 000402  005746                  TST -(%6)
 91 000404  062767  000006  000000G ADD #6,TOP
 92 000412  026727  000000G 000264G CMP TOP,#STACKB+264
 93 000420  101022                  BHI $52
 94 000422  105077  000000G         CLRB @TOP
 95 000426  016700  000000G         MOV TOP,%0
 96 000432  112760  000101  000001  MOVB #101,1(%0)
 97 000440  016700  000000G         MOV TOP,%0
 98 000444  112760  000002  000002  MOVB #2,2(%0)
 99 000452  016700  000000G         MOV TOP,%0
100 000456  016560  000004  000004  MOV 4(%5),4(%0)
101 000464  000404                  BR $72
102 000466                          $52: ;33
103 000466  012716  000020          MOV #20,(%6)
104 000472  004767  000000G         JSR %7,ERROR
105 000476                          $72: ; 37
106 000476  000167  000000G         JMP RETS
107                                 .GLOBL PUSH
108                                 .GLOBL ERROR
109                                 .GLOBL MOVN
110                                 .GLOBL POP
111                                 .GLOBL PUSHINT
112                                 .GLOBL TOPTOI
113                                 .GLOBL EQ
114                                 .GLOBL CURGLOB
115                                 .GLOBL CURFUN
116                                 .GLOBL FUNB
117                                 .GLOBL NXTVAR
118                                 .GLOBL VARB
119                                 .GLOBL TOP
120                                 .GLOBL STACKB
121                                 .GLOBL APPLVL
122                                 .GLOBL ERR
123                                 .GLOBL BRAKE
124                                 .GLOBL LEAVE
125                                 .GLOBL PRUSED
126                                 .GLOBL PROGEND
127                                 .GLOBL ERRAT
128                                 .GLOBL LNAME
129                                 .GLOBL FNAME
130                                 .GLOBL CURSOR
131                                 .GLOBL PR
```

```
132 000502                              TOPTOI:
133 000502  004567  000000G             JSR %5,SAVE
134 000506  016700  000000G             MOV TOP,%0
135 000512  126027  000001  000101      CMPB 1(%0),#101
136 000520  001003                      BNE $13
137 000522  004767  177436              JSR %7,POP
138 000526  000442                      BR $4
139 000530                              $13: ;13
140 000530  016700  000000G             MOV TOP,%0
141 000534  126027  000002  000001      CMPB 2(%0),#1
142 000542  001010                      BNE $53
143 000544  105777  000000G             TSTB @TOP
144 000550  001005                      BNE $53
145 000552  016700  000000G             MOV TOP,%0
146 000556  117004  000004              MOVB @4(%0),%4
147 000562  000421                      BR $33
148 000564                              $53: ;31
149 000564  016700  000000G             MOV TOP,%0
150 000570  016000  000004              MOV 4(%0),%0
151 000574  116004  000001              MOVB 1(%0),%4
152 000600  010400                      MOV %4,%0
153 000602  072027  000010              ASH #10,%0
154 000606  016701  000000G             MOV TOP,%1
155 000612  117101  000004              MOVB @4(%1),%1
156 000616  042701  177400              BIC #-400,%1
157 000622  050100                      BIS %1,%0
158 000624  010004                      MOV %0,%4
159 000626                              $33: ; 52
160 000626  004767  177332              JSR %7,POP
161 000632  010400                      MOV %4,%0
162 000634                              $4:
163 000634  000167  000000G             JMP RET
164         000001                      .END
```

```
                                        .TITLE TINY-C/11-01-03/SYMBOL
                                        .DSABL REG
                                        .ENABL GBL
000000                                  .PSECT TEXT
000000                                  ADDRVAL:
000000  004567  000000G                 JSR %5,SAVE
000004  062706  177770                  ADD #-10,%6
000010  010516                          MOV %5,(%6)
000012  062716  177762                  ADD #-16,(%6)
000016  004767  000124                  JSR %7,CANON
000022  016704  000000G                 MOV CURFUN,%4
000026                                  $7: ; 11
000026  011402                          MOV (%4),%2
000030                                  $11: ; 12
000030  020264  000002                  CMP %2,2(%4)
000034  101027                          BHI $31
000036  012716  000010                  MOV #10,(%6)
000042  010246                          MOV %2,-(%6)
000044  012746  177762                  MOV #-16,-(%6)
000050  060516                          ADD %5,(%6)
000052  004767  000000G                 JSR %7,CEQN
000056  022626                          CMP (%6)+,(%6)+
000060  005700                          TST %0
000062  001426                          BEQ $51
000064  010200                          MOV %2,%0
000066  000422                          BR $6
000070                                  $5: ;32
000070  020467  000000G                 CMP %4,CURFUN
000074  001003                          BNE $2
000076  016700  000000G                 MOV CURGLOB,%0
000102  000402                          BR $4
000104                                  $2:
000104  012700  000000G                 MOV #FUNB,%0
000110                                  $4:
000110  010004                          MOV %0,%4
000112  000745                          BR $7
000114                                  $31: ;44
000114  020427  000000G                 CMP %4,#FUNB
000120  001363                          BNE $5
000122  012716  000032                  MOV #32,(%6)
000126  004767  000000G                 JSR %7,ERROR
000132  005000                          CLR %0
000134                                  $6:
000134  000167  000000G                 JMP RET
000140                                  $51: ;56
000140  062702  000016                  ADD #16,%2
000144  000731                          BR $11
```

```
 48 000146                              CANON:
 49 000146  004567  000000G            JSR %5,SAVE
 50 000152  016504  000004             MOV 4(%5),%4
 51 000156  112702  000010             MOVB #10,%2
 52 000162                              $52: ; 6
 53 000162  005702                      TST %2
 54 000164  003413                      BLE $72
 55 000166  026767  000000G 000000G    CMP FNAME,LNAME
 56 000174  101007                      BHI $72
 57 000176  016700  000000G            MOV FNAME,%0
 58 000202  005267  000000G            INC FNAME
 59 000206  111024                      MOVB (%0),(%4)+
 60 000210  005302                      DEC %2
 61 000212  000763                      BR $52
 62 000214                              $72: ;23
 63 000214  005702                      TST %2
 64 000216  001004                      BNE $53
 65 000220  117764  000000G 177777     MOVB @LNAME,-1(%4)
 66 000226  000401                      BR $73
 67 000230                              $53: ;31
 68 000230  105014                      CLRB (%4)
 69 000232                              $73: ; 32
 70 000232  000167  000000G            JMP RET
 71 000236                              FUNDON:
 72 000236  010546                      MOV %5,-(%6)
 73 000240  010605                      MOV %6,%5
 74 000242  017767  000000G 000000G    MOV @CURFUN,NXTVAR
 75 000250  016700  000000G            MOV CURFUN,%0
 76 000254  162767  000006  000000G    SUB #6,CURFUN
 77 000262  016067  000004  000000G    MOV 4(%0),PRUSED
 78 000270  000167  000000G            JMP RETS
 79 000274                              NEWFUN:
 80 000274  010546                      MOV %5,-(%6)
 81 000276  010605                      MOV %6,%5
 82 000300  005746                      TST -(%6)
 83 000302  062767  000006  000000G    ADD #6,CURFUN
 84 000310  026727  000000G 000264G    CMP CURFUN,#FUNB+264
 85 000316  101021                      BHI $14
 86 000320  016777  000000G 000000G    MOV NXTVAR,@CURFUN
 87 000326  016700  000000G            MOV CURFUN,%0
 88 000332  016701  000000G            MOV NXTVAR,%1
 89 000336  062701  177762             ADD #-16,%1
 90 000342  010160  000002             MOV %1,2(%0)
 91 000346  016700  000000G            MOV CURFUN,%0
 92 000352  016760  000000G 000004     MOV PRUSED,4(%0)
 93 000360  000404                      BR $34
 94 000362                              $14: ;32
 95 000362  012716  000021             MOV #21,(%6)
 96 000366  004767  000000G            JSR %7,ERROR
 97 000372                              $34: ; 36
 98 000372  000167  000000G            JMP RETS
 99                                     .GLOBL MOVN
100                                     .GLOBL NEWVAR
101                                     .GLOBL NEWFUN
102                                     .GLOBL FUNDON
```

```
                                        .GLOBL ERROR
                                        .GLOBL CEQN
                                        .GLOBL CANON
                                        .GLOBL ADDRVAL
                                        .GLOBL CURGLOB
                                        .GLOBL CURFUN
                                        .GLOBL FUNB
                                        .GLOBL NXTVAR
                                        .GLOBL VARB
                                        .GLOBL TOP
                                        .GLOBL STACKB
                                        .GLOBL APPLVL
                                        .GLOBL ERR
                                        .GLOBL BRAKE
                                        .GLOBL LEAVE
                                        .GLOBL PRUSED
                                        .GLOBL PROGEND
                                        .GLOBL ERRAT
                                        .GLOBL LNAME
                                        .GLOBL FNAME
                                        .GLOBL CURSOR
                                        .GLOBL PR
000376                                  NEWVAR:
000376  004567  000000G                 JSR %5,SAVE
000402  016504  000004                  MOV 4(%5),%4
000406  026727  000000G 005360G         CMP NXTVAR,#VARB+5360
000414  101037                          BHI $54
000416  016716  000000G                 MOV NXTVAR,(%6)
000422  004767  177520                  JSR %7,CANON
000426  016700  000000G                 MOV NXTVAR,%0
000432  116460  000010  000010          MOVB 10(%4),10(%0)
000440  016700  000000G                 MOV NXTVAR,%0
000444  116460  000011  000011          MOVB 11(%4),11(%0)
000452  016700  000000G                 MOV NXTVAR,%0
000456  016460  000012  000012          MOV 12(%4),12(%0)
000464  005764  000014                  TST 14(%4)
000470  001417                          BEQ $74
000472  105764  000010                  TSTB 10(%4)
000476  001414                          BEQ $74
000500  016700  000000G                 MOV NXTVAR,%0
000504  016460  000014  000014          MOV 14(%4),14(%0)
000512  000506                          BR $15
000514                                  $54: ;47
000514  012716  000022                  MOV #22,(%6)
000520  004767  000000G                 JSR %7,ERROR
000524                                  $37: ; 53
000524  000167  000000G                 JMP RET
000530                                  $74: ;55
000530  016700  000000G                 MOV NXTVAR,%0
000534  016701  000000G                 MOV PRUSED,%1
000540  005201                          INC %1
000542  010160  000014                  MOV %1,14(%0)
000546  012701  000000G                 MOV #PR,%1
000552  005000                          CLR %0
000554  062700  000000                  ADD #0,%0
```

```
158 000560  062701  047040          ADD #47040,%1
159 000564  005500                  ADC %0
160 000566  010146                  MOV %1,-(%6)
161 000570  010046                  MOV %0,-(%6)
162 000572  116401  000011          MOVB 11(%4),%1
163 000576  070164  000012          MUL 12(%4),%1
164 000602  010101                  MOV %1,%1
165 000604  006700                  SXT %0
166 000606  010146                  MOV %1,-(%6)
167 000610  010046                  MOV %0,-(%6)
168 000612  016701  000000G         MOV PRUSED,%1
169 000616  005000                  CLR %0
170 000620  062600                  ADD (%6)+,%0
171 000622  062601                  ADD (%6)+,%1
172 000624  005500                  ADC %0
173 000626  162600                  SUB (%6)+,%0
174 000630  162601                  SUB (%6)+,%1
175 000632  005600                  SBC %0
176 000634  073027  000000          ASHC #0,%0
177 000640  003027                  BGT $35
178 000642  116401  000011          MOVB 11(%4),%1
179 000646  070164  000012          MUL 12(%4),%1
180 000652  060167  000000G         ADD %1,PRUSED
181 000656  005764  000014          TST 14(%4)
182 000662  001434                  BEQ $55
183 000664  116400  000011          MOVB 11(%4),%0
184 000670  010016                  MOV %0,(%6)
185 000672  012746  000014          MOV #14,-(%6)
186 000676  060416                  ADD %4,(%6)
187 000700  016700  000000G         MOV NXTVAR,%0
188 000704  016046  000014          MOV 14(%0),-(%6)
189 000710  004767  000000G         JSR %7,MOVN
190 000714  022626                  CMP (%6)+,(%6)+
191 000716  000404                  BR $15
192 000720                          $35: ;150
193 000720  012716  000023          MOV #23,(%6)
194 000724  004767  000000G         JSR %7,ERROR
195 000730                          $15: ; 154
196 000730  016700  000000G         MOV CURFUN,%0
197 000734  016701  000000G         MOV NXTVAR,%1
198 000740  062767  000016  000000G ADD #16,NXTVAR
199 000746  010160  000002          MOV %1,2(%0)
200 000752  000664                  BR $37
201 000754                          $55: ;166
202 000754  016700  000000G         MOV NXTVAR,%0
203 000760  016002  000014          MOV 14(%0),%2
204 000764                          $16: ; 172
205 000764  020267  000000G         CMP %2,PRUSED
206 000770  101357                  BHI $15
207 000772                          $56: ; 175
208 000772  105022                  CLRB (%2)+
209 000774  000773                  BR $16
210         000001                  .END
```

```
                                        .TITLE TINY-C/11-01-03/UTIL
                                        .DSABL REG
                                        .ENABL GBL
000000                                  .PSECT TEXT
000000                                  ATOI:
000000  004567  000400                  JSR %5,SAVE
000004  005746                          TST -(%6)
000006  016504  000004                  MOV 4(%5),%4
000012  005065  177770                  CLR -10(%5)
000016  121427  000053                  CMPB (%4),#53
000022  001403                          BEQ $3
000024  121427  000055                  CMPB (%4),#55
000030  001011                          BNE $1
000032                                  $3: ;14
000032  122427  000055                  CMPB (%4)+,#55
000036  001003                          BNE $2
000040  112700  000001                  MOVB #1,%0
000044  000401                          BR $4
000046                                  $2:
000046  005000                          CLR %0
000050                                  $4:
000050  010065  177770                  MOV %0,-10(%5)
000054                                  $1: ; 25
000054  005003                          CLR %3
000056                                  $5: ; 26
000056  122714  000060                  CMPB #60,(%4)
000062  003007                          BGT $6
000064  121427  000071                  CMPB (%4),#71
000070  003004                          BGT $6
000072  111400                          MOVB (%4),%0
000074  062700  177720                  ADD #-60,%0
000100  000402                          BR $01
000102                                  $6:
000102  112700  177777                  MOVB #-1,%0
000106                                  $01:
000106  110002                          MOVB %0,%2
000110  002407                          BLT $7
000112  010301                          MOV %3,%1
000114  070127  000012                  MUL #12,%1
000120  060201                          ADD %2,%1
000122  010103                          MOV %1,%3
000124  005204                          INC %4
000126  000753                          BR $5
000130                                  $7: ;53
000130  005765  177770                  TST -10(%5)
000134  001403                          BEQ $21
000136  010300                          MOV %3,%0
000140  005400                          NEG %0
000142  000401                          BR $41
000144                                  $21:
000144  010300                          MOV %3,%0
000146                                  $41:
000146  000167  000246                  JMP RET
```

```
 54 000152                                CEQ:
 55 000152  004567  000226                JSR %5,SAVE
 56 000156  016504  000004                MOV 4(%5),%4
 57 000162  016502  000006                MOV 6(%5),%2
 58 000166                                $51: ; 6
 59 000166  122214                        CMPB (%2)+,(%4)
 60 000170  001005                        BNE $71
 61 000172  105724                        TSTB (%4)+
 62 000174  001374                        BNE $51
 63 000176  112700  000001                MOVB #1,%0
 64 000202  000401                        BR $61
 65 000204                                $71: ;15
 66 000204  005000                        CLR %0
 67 000206                                $61:
 68 000206  000167  000206                JMP RET
 69 000212                                CEQN:
 70 000212  004567  000166                JSR %5,SAVE
 71 000216  016504  000004                MOV 4(%5),%4
 72 000222  016502  000006                MOV 6(%5),%2
 73 000226  016503  000010                MOV 10(%5),%3
 74 000232                                $32: ; 10
 75 000232  010300                        MOV %3,%0
 76 000234  005303                        DEC %3
 77 000236  005700                        TST %0
 78 000240  003404                        BLE $52
 79 000242  122214                        CMPB (%2)+,(%4)
 80 000244  001406                        BEQ $72
 81 000246  005000                        CLR %0
 82 000250  000402                        BR $02
 83 000252                                $52: ;20
 84 000252  112700  000001                MOVB #1,%0
 85 000256                                $02:
 86 000256  000167  000136                JMP RET
 87 000262                                $72: ;24
 88 000262  105724                        TSTB (%4)+
 89 000264  001362                        BNE $32
 90 000266  112700  000001                MOVB #1,%0
 91 000272  000771                        BR $02
 92 000274                                ERROR:
 93 000274  010546                        MOV %5,-(%6)
 94 000276  010605                        MOV %6,%5
 95 000300  005767  000000G               TST ERR
 96 000304  001006                        BNE $33
 97 000306  016567  000004  000000G       MOV 4(%5),ERR
 98 000314  016767  000000G 000000G       MOV CURSOR,ERRAT
 99 000322                                $33: ;13
100 000322  000167  000102                JMP RETS
```

```
101                     .GLOBL MOVN
102                     .GLOBL ERROR
103                     .GLOBL CEQN
104                     .GLOBL CEQ
105                     .GLOBL ATOI
106                     .GLOBL CURGLOB
107                     .GLOBL CURFUN
108                     .GLOBL FUNB
109                     .GLOBL NXTVAR
110                     .GLOBL VARB
111                     .GLOBL TOP
112                     .GLOBL STACKB
113                     .GLOBL APPLVL
114                     .GLOBL ERR
115                     .GLOBL BRAKE
116                     .GLOBL LEAVE
117                     .GLOBL PRUSED
118                     .GLOBL PROGEND
119                     .GLOBL ERRAT
120                     .GLOBL LNAME
121                     .GLOBL FNAME
122                     .GLOBL CURSOR
123                     .GLOBL PR
```

```
124 000326                          MOVN:
125 000326  004567  000052          JSR %5,SAVE
126 000332  016504  000004          MOV 4(%5),%4
127 000336  016502  000006          MOV 6(%5),%2
128 000342  016503  000010          MOV 10(%5),%3
129 000346                          $53: ; 10
130 000346  010300                  MOV %3,%0
131 000350  005303                  DEC %3
132 000352  005700                  TST %0
133 000354  003402                  BLE $73
134 000356  112224                  MOVB (%2)+,(%4)+
135 000360  000772                  BR $53
136 000362                          $73: ;16
137 000362  000167  000032          JMP RET
138 000366                          SWITCH:
139 000366  005721                  TST (%1)+
140 000370  001404                  BEQ .+12
141 000372  020021                  CMP %0,(%1)+
142 000374  001374                  BNE .-6
143 000376  000171  177774          JMP @-4(%1)
144 000402  000131                  JMP @(%1)+
145 000404                          SAVE:
146 000404  010500                  MOV %5,%0
147 000406  010605                  MOV %6,%5
148 000410  010446                  MOV %4,-(%6)
149 000412  010346                  MOV %3,-(%6)
150 000414  010246                  MOV %2,-(%6)
151 000416  004710                  JSR %7,(%0)
152                                 .GLOBL SWITCH
153                                 .GLOBL SAVE
154                                 .GLOBL RET
155                                 .GLOBL RETS
156 000420                          RET:
157 000420  010502                  MOV %5,%2
158 000422  014204                  MOV -(%2),%4
159 000424  014203                  MOV -(%2),%3
160 000426  014202                  MOV -(%2),%2
161 000430                          RETS:
162 000430  010506                  MOV %5,%6
163 000432  012605                  MOV (%6)+,%5
164 000434  000207                  RTS %7
165         000001                  .END
```

```
                                        .TITLE  tiny-c/11-01-03/MC
000000                                  .PSECT  TCMC
                                        .ENABL  GBL
        000030                          MCERR = 24.
                                ;
                                ; machine call (MC) subroutines for tiny-c/11
                                ;
                                        .GLOBL  MC
000000  004567  000000G        MC:     JSR     R5,SAVE
000004  005765  000004                  TST     4(R5)
000010  003002                          BGT     1$
000012  000167  002670                  JMP     MC99
000016  162706  000030        1$:     SUB     #30,SP          ; make some room for locals
000022  004767  000000G                 JSR     PC,POP          ; get the function number off the stack
000026  010065  177770                  MOV     R0,-10(R5)
000032  005365  000004                  DEC     4(R5)
000036  022700  001750                  CMP     #1000.,R0       ; test for Private MC
000042  101012                          BHI     2$
000044  016516  000004                  MOV     4(R5),(SP)      ; push number of arguments
000050  016546  177770                  MOV     -10(R5),-(SP)   ; push Private MC number
000054  162716  001750                  SUB     #1000.,(SP)
000060  004767  000000G                 JSR     PC,USERMC       ; dispatch to Private MC handler
000064  000167  000000G                 JMP     RET
000070  020027  000017        2$:     CMP     R0,#MCMAX
000074  003402                          BLE     3$
000076  000167  002564                  JMP     MC98
000102  005300                3$:     DEC     R0
000104  006300                          ASL     R0
000106  000170  000112'                 JMP     @MCTAB(R0)      ; dispatch to Standard MC
                                ;
                                ; Standard Machine Call Table
                                ;
000112  000150'               MCTAB:  .WORD   MC1     ; send ASCII character to standard output
000114  000210'                       .WORD   MC2     ; get ASCII character from standard input
000116  000252'                       .WORD   MC3     ; open a file
000120  000334'                       .WORD   MC4     ; read from a file
000122  000402'                       .WORD   MC5     ; write to a file
000124  000456'                       .WORD   MC6     ; close a file
000126  000516'                       .WORD   MC7     ; move a block of RAM
000130  000674'                       .WORD   MC8     ; count character instances
000132  001014'                       .WORD   MC9     ; find a specific character instance
000134  001210'                       .WORD   MC10    ; execute a processor halt
000136  001216'                       .WORD   MC11    ; enter an application program
000140  001602'                       .WORD   MC12    ; test character ready on standard input
000142  001632'                       .WORD   MC13    ; send character block to standard output
000144  001724'                       .WORD   MC14    ; send signed integer value to standard output
000146  002626'                       .WORD   MC15    ; get character string from standard input
        000036                  MCMAX=.-MCTAB
        000017                  MCMAX=MCMAX/2
```

```
50                                      ;
51                                      ; MC 1 ... put a character to the standard output
52                                      ;
53 000150  022765  000001  000004  MC1:    CMP     #1,4(R5)
54 000156  001402                         BEQ     1$
55 000160  000167  002502                 JMP     MC98
56 000164  004767  000000G        1$:     JSR     PC,TOPTOI
57 000170  110016                         MOVB    R0,(SP)
58 000172  004737  000000G                JSR     PC,@#OUTCH
59 000176  005016                         CLR     (SP)
60 000200  004737  000000G                JSR     PC,@#PUSHINT
61 000204  000167  000000G                JMP     RET
62                                      ;
63                                      ; MC 2 ... get a character from the standard input
64                                      ;
65 000210  005765  000004         MC2:    TST     4(R5)
66 000214  001402                         BEQ     1$
67 000216  000167  002444                 JMP     MC98
68 000222  004767  000000G        1$:     JSR     PC,INCH
69 000226  010016                         MOV     R0,(SP)
70 000230  012746  000001                 MOV     #1,-(SP)
71 000234  012746  000101                 MOV     #'A,-(SP)
72 000240  005046                         CLR     -(SP)
73 000242  004737  000000G                JSR     PC,@#PUSH
74 000246  000167  000000G                JMP     RET
75                                      ;
76                                      ; MC 3 ... file open(r/w, name, size, channel)
77                                      ;
78 000252  022765  000004  000004  MC3:    CMP     #4,4(R5)
79 000260  001402                         BEQ     1$
80 000262  000167  002400                 JMP     MC98
81 000266  004767  000000G        1$:     JSR     PC,TOPTOI
82 000272  010016                         MOV     R0,(SP)
83 000274  004767  000000G                JSR     PC,TOPTOI
84 000300  010046                         MOV     R0,-(SP)
85 000302  004767  000000G                JSR     PC,TOPTOI
86 000306  010046                         MOV     R0,-(SP)
87 000310  004767  000000G                JSR     PC,TOPTOI
88 000314  010046                         MOV     R0,-(SP)
89 000316  004737  000000G                JSR     PC,@#FOPEN
90 000322  010016                         MOV     R0,(SP)
91 000324  004737  000000G                JSR     PC,@#PUSHINT
92 000330  000167  000000G                JMP     RET
```

```
 93                                     ;
 94                                     ; MC 4 ... file read(to, channel)
 95                                     ;
 96 000334  022765  000002  000004  MC4:    CMP     #2,4(R5)
 97 000342  001402                          BEQ     1$
 98 000344  000167  002316                  JMP     MC98
 99 000350  004767  000000G         1$:     JSR     PC,TOPTOI
100 000354  010016                         MOV     R0,(SP)
101 000356  004767  000000G                 JSR     PC,TOPTOI
102 000362  010046                         MOV     R0,-(SP)
103 000364  004737  000000G                 JSR     PC,@#FREAD
104 000370  010016                         MOV     R0,(SP)
105 000372  004737  000000G                 JSR     PC,@#PUSHINT
106 000376  000167  000000G                 JMP     RET
107                                     ;
108                                     ; MC 5 ... file write(from, to, channel)
109                                     ;
110 000402  022765  000003  000004  MC5:    CMP     #3,4(R5)
111 000410  001402                          BEQ     1$
112 000412  000167  002250                  JMP     MC98
113 000416  004767  000000G         1$:     JSR     PC,TOPTOI
114 000422  010016                         MOV     R0,(SP)
115 000424  004767  000000G                 JSR     PC,TOPTOI
116 000430  010046                         MOV     R0,-(SP)
117 000432  004767  000000G                 JSR     PC,TOPTOI
118 000436  010046                         MOV     R0,-(SP)
119 000440  004737  000000G                 JSR     PC,@#FWRITE
120 000444  010016                         MOV     R0,(SP)
121 000446  004737  000000G                 JSR     PC,@#PUSHINT
122 000452  000167  000000G                 JMP     RET
123                                     ;
124                                     ; MC 6 ... file close(channel)
125                                     ;
126 000456  022765  000001  000004  MC6:    CMP     #1,4(R5)
127 000464  001402                          BEQ     1$
128 000466  000167  002174                  JMP     MC98
129 000472  004767  000000G         1$:     JSR     PC,TOPTOI
130 000476  010016                         MOV     R0,(SP)
131 000500  004767  000000G                 JSR     PC,FCLOSE
132 000504  010016                         MOV     R0,(SP)
133 000506  004737  000000G                 JSR     PC,@#PUSHINT
134 000512  000167  000000G                 JMP     RET
```

```
135                                     ;
136                                     ; MC 7 ... copy a block(start, end, bytes to new start)
137                                     ;
138 000516  022765  000003  000004  MC7:     CMP     #3,4(R5)
139 000524  001402                          BEQ     1$
140 000526  000167  002134                  JMP     MC98
141 000532  004767  000000G         1$:     JSR     PC,TOPTOI
142 000536  010065  177766                  MOV     R0,-12(R5)
143 000542  004767  000000G                 JSR     PC,TOPTOI
144 000546  010065  177764                  MOV     R0,-14(R5)
145 000552  004767  000000G                 JSR     PC,TOPTOI
146 000556  010065  177762                  MOV     R0,-16(R5)
147 000562  005765  177766                  TST     -12(R5)
148 000566  003415                          BLE     3$
149 000570  026565  177764  177762  2$:     CMP     -14(R5),-16(R5)
150 000576  103411                          BLO     3$
151 000600  016500  177764                  MOV     -14(R5),R0
152 000604  066500  177766                  ADD     -12(R5),R0
153 000610  117510  177764                  MOVB    @-14(R5),(R0)
154 000614  005365  177764                  DEC     -14(R5)
155 000620  000763                          BR      2$
156 000622  005765  177766          3$:     TST     -12(R5)
157 000626  002015                          BGE     5$
158 000630  026565  177764  177762  4$:     CMP     -14(R5),-16(R5)
159 000636  103411                          BLO     5$
160 000640  016500  177762                  MOV     -16(R5),R0
161 000644  066500  177766                  ADD     -12(R5),R0
162 000650  117510  177762                  MOVB    @-16(R5),(R0)
163 000654  005265  177762                  INC     -16(R5)
164 000660  000763                          BR      4$
165 000662  005016                  5$:     CLR     (SP)
166 000664  004737  000000G                 JSR     PC,@#PUSHINT
167 000670  000167  000000G                 JMP     RET
```

```
168                                     ;
169                                     ; MC 8 ... count character instances(from, to, character)
170                                     ;
171 000674  022765  000003  000004  MC8:     CMP     #3,4(R5)
172 000702  001402                          BEQ     1$
173 000704  000167  001756                  JMP     MC98
174 000710  004767  000000G         1$:     JSR     PC,TOPTOI
175 000714  110065  177766                  MOVB    R0,-12(R5)
176 000720  004767  000000G                 JSR     PC,TOPTOI
177 000724  010065  177764                  MOV     R0,-14(R5)
178 000730  004767  000000G                 JSR     PC,TOPTOI
179 000734  010065  177762                  MOV     R0,-16(R5)
180 000740  005065  177760                  CLR     -20(R5)
181 000744  026565  177764  177762  2$:     CMP     -14(R5),-16(R5)
182 000752  103412                        BLO     3$
183 000754  016500  177762                  MOV     -16(R5),R0
184 000760  005265  177762                  INC     -16(R5)
185 000764  126510  177766                  CMPB    -12(R5),(R0)
186 000770  001365                        BNE     2$
187 000772  005265  177760                  INC     -20(R5)
188 000776  000762                        BR      2$
189 001000  016516  177760         3$:     MOV     -20(R5),(SP)
190 001004  004737  000000G                 JSR     PC,@#PUSHINT
191 001010  000167  000000G                 JMP     RET
192                                     ;
193                                     ; MC 9 ... find nth character instance(from, to, character, n)
194                                     ;
195 001014  022765  000004  000004  MC9:     CMP     #4,4(R5)
196 001022  001402                          BEQ     1$
197 001024  000167  001636                  JMP     MC98
198 001030  004767  000000G         1$:     JSR     PC,TOPTOI
199 001034  010065  177764                  MOV     R0,-14(R5)
200 001040  111065  177754                  MOVB    (R0),-24(R5)
201 001044  116065  000001  177755          MOVB    1(R0),-23(R5)
202 001052  004767  000000G                 JSR     PC,TOPTOI
203 001056  110065  177766                  MOVB    R0,-12(R5)
204 001062  004767  000000G                 JSR     PC,TOPTOI
205 001066  010065  177762                  MOV     R0,-16(R5)
206 001072  004767  000000G                 JSR     PC,TOPTOI
207 001076  010065  177760                  MOV     R0,-20(R5)
208 001102  012765  177777  177756          MOV     #-1,-22(R5)
209 001110  005765  177754          2$:     TST     -24(R5)
210 001114  003420                        BLE     4$
211 001116  026565  177762  177760          CMP     -16(R5),-20(R5)
212 001124  103414                        BLO     4$
213 001126  016500  177760                  MOV     -20(R5),R0
214 001132  005265  177760                  INC     -20(R5)
215 001136  126510  177766                  CMPB    -12(R5),(R0)
216 001142  001002                        BNE     3$
```

```
217 001144  005365  177754                      DEC     -24(R5)
218 001150  005265  177756          3$:         INC     -22(R5)
219 001154  000755                              BR      2$
220 001156  016500  177764          4$:         MOV     -14(R5),R0
221 001162  116510  177754                      MOVB    -24(R5),(R0)
222 001166  116560  177755  000001              MOVB    -23(R5),1(R0)
223 001174  016516  177756                      MOV     -22(R5),(SP)
224 001200  004737  000000G                     JSR     PC,@#PUSHINT
225 001204  000167  000000G                     JMP     RET
226                                     ;
227                                     ; MC 10 ... execute a processor halt
228                                     ;
229 001210  000000                      MC10:   HALT
230 001212  000167  000000G                     JMP     RET
231                                     ;
232                                     ; MC 11 ... enter an application program
233                                     ;
234 001216  022765  000004  000004      MC11:   CMP     #4,4(R5)
235 001224  001402                              BEQ     1$
236 001226  000167  001434                      JMP     MC98
237 001232  016765  000000G 177764      1$:     MOV     CURSOR,-14(R5)
238 001240  016765  000000G 177756              MOV     PROGEND,-22(R5)
239 001246  016765  000000G 177752              MOV     PRUSED,-26(R5)
240 001254  016765  000000G 177766              MOV     CURGLOBA,-12(R5)
241 001262  004767  000000G                     JSR     PC,TOPTOI
242 001266  010067  000000G                     MOV     R0,CURSOR
243 001272  010065  177762                      MOV     R0,-16(R5)
244 001276  004767  000000G                     JSR     PC,TOPTOI
245 001302  010067  000000G                     MOV     R0,PROGEND
246 001306  010067  000000G                     MOV     R0,PRUSED
247 001312  004767  000000G                     JSR     PC,LINK
248 001316  016767  000000G 000000G             MOV     CURFUN,CURGLOBA
249 001324  004767  000000G                     JSR     PC,TOPTOI
250 001330  010067  000000G                     MOV     R0,CURSOR
251 001334  004767  000000G                     JSR     PC,NEWFUN
252 001340  004767  000000G                     JSR     PC,TOPTOI
253 001344  010065  177760                      MOV     R0,-20(R5)
254 001350  016765  000000G 177750             MOV     TOP,-30(R5)
255 001356  005767  000000G                     TST     ERR
256 001362  001012                              BNE     2$
257 001364  005267  000000G                     INC     APPLVL
258 001370  004767  000000G                     JSR     PC,PRBEGN
259 001374  004767  000000G                     JSR     PC,ST
260 001400  004767  000000G                     JSR     PC,PRDONE
261 001404  005367  000000G                     DEC     APPLVL
262 001410  004767  000000G          2$:        JSR     PC,FUNDONE
263 001414  004767  000000G                     JSR     PC,FUNDONE
264 001420  016700  000000G                     MOV     CURSOR,R0
265 001424  166500  177762                      SUB     -16(R5),R0
266 001430  010065  177754                      MOV     R0,-24(R5)
267 001434  005767  000000G                     TST     ERR
268 001440  001406                              BEQ     3$
```

```
269 001442  016700  000000G                 MOV     ERRAT,R0
270 001446  166500  177762                  SUB     -16(R5),R0
271 001452  010065  177754                  MOV     R0,-24(R5)
272 001456  012716  000002          3$:     MOV     #2,(SP)
273 001462  010546                          MOV     R5,-(SP)
274 001464  062716  177754                  ADD     #-24,(SP)
275 001470  016546  177760                  MOV     -20(R5),-(SP)
276 001474  062716  000002                  ADD     #2,(SP)
277 001500  004737  000000G                 JSR     PC,@#MOVN
278 001504  022626                          CMP     (SP)+,(SP)+
279 001506  012716  000002                  MOV     #2,(SP)
280 001512  012746  000000G                 MOV     #ERR,-(SP)
281 001516  016546  177760                  MOV     -20(R5),-(SP)
282 001522  004737  000000G                 JSR     PC,@#MOVN
283 001526  016567  177764  000000G         MOV     -14(R5),CURSOR
284 001534  016567  177750  000000G         MOV     -30(R5),TOP
285 001542  016567  177756  000000G         MOV     -22(R5),PROGEND
286 001550  016567  177752  000000G         MOV     -26(R5),PRUSED
287 001556  016567  177766  000000G         MOV     -12(R5),CURGLOBA
288 001564  005067  000000G                 CLR     ERR
289 001570  005016                          CLR     (SP)
290 001572  004737  000000G                 JSR     PC,@#PUSHINT
291 001576  000167  000000G                 JMP     RET
292                                 ;
293                                 ; MC 12 ... character ready
294                                 ;
295 001602  005765  000004          MC12:   TST     4(R5)
296 001606  001402                          BEQ     1$
297 001610  000167  001052                  JMP     MC98
298 001614  004767  000000G         1$:     JSR     PC,CHRDY
299 001620  010016                          MOV     R0,(SP)
300 001622  004737  000000G                 JSR     PC,@#PUSHINT
301 001626  000167  000000G                 JMP     RET
302                                 ;
303                                 ; MC 13 ... print character block(from, to)
304                                 ;
305 001632  022765  000002  000004  MC13:   CMP     #2,4(R5)
306 001640  001402                          BEQ     1$
307 001642  000167  001020                  JMP     MC98
308 001646  004767  000000G         1$:     JSR     PC,TOPTOI
309 001652  010065  177766                  MOV     R0,-12(R5)
310 001656  004767  000000G                 JSR     PC,TOPTOI
311 001662  010001                          MOV     R0,R1
312 001664  020165  177766          2$:     CMP     R1,-12(R5)
313 001670  101010                          BHI     4$
314 001672  111116                          MOVB    @R1,(SP)
315 001674  001002                          BNE     3$
316 001676  112716  000042                  MOVB    #'",(SP)
317 001702  004737  000000G         3$:     JSR     PC,@#OUTCH
318 001706  005201                          INC     R1
319 001710  000765                          BR      2$
320 001712  005016                  4$:     CLR     (SP)
321 001714  004737  000000G                 JSR     PC,@#PUSHINT
322 001720  000167  000000G                 JMP     RET
```

```
                                        ;
                                        ; MC 14 ... print signed integer(n)
                                        ;
001724  022765  000001  000004  MC14:   CMP     #1,4(R5)
001732  001402                         BEQ     2$
001734  000167  000726                  JMP     MC98
001740  004767  000000G         2$:     JSR     PC,TOPTOI
001744  010016                         MOV     R0,(SP)
001746  012746  000000                  MOV     #0,-(SP)
001752  005766  000002                  TST     2(SP)
001756  100014                         BPL     4$
001760  005466  000002                  NEG     2(SP)
001764  102004                         BVC     3$
001766  012716  044000                  MOV     #44000,(SP)
001772  005066  000002                  CLR     2(SP)
001776  012746  000055          3$:     MOV     #'-,-(SP)
002002  004737  000000G                 JSR     PC,@#OUTCH
002006  005726                         TST     (SP)+
002010  016604  000002          4$:     MOV     2(SP),R4
002014  005002                         CLR     R2
002016  011603                         MOV     (SP),R3
002020  005746                         TST     -(SP)
002022  012700  000037                  MOV     #37,R0
002026  005704                         TST     R4
002030  001006                         BNE     5$
002032  012716  000060                  MOV     #'0,(SP)
002036  004737  000000G                 JSR     PC,@#OUTCH
002042  000167  000460                  JMP     EX14
002046  005300                 5$:     DEC     R0
002050  006304                         ASL     R4
002052  006103                         ROL     R3
002054  030327  040000          6$:     BIT     R3,#40000
002060  001772                         BEQ     5$
002062  005700                         TST     R0
002064  003016                         BGT     7$
002066  006203                         ASR     R3
002070  006004                         ROR     R4
002072  006203                         ASR     R3
002074  006004                         ROR     R4
002076  006203                         ASR     R3
002100  006004                         ROR     R4
002102  006203                         ASR     R3
002104  006004                         ROR     R4
002106  062700  000004                  ADD     #4,R0
002112  004767  000422                  JSR     PC,MUL
002116  005302                         DEC     R2
002120  000755                         BR      6$
002122  020027  000004          7$:     CMP     R0,#4
002126  003407                         BLE     10$
002130  004767  000434          8$:     JSR     PC,DIV
002134  005202                         INC     R2
002136  000746                         BR      6$
002140  005200                 9$:     INC     R0
002142  006203                         ASR     R3
```

```
377 002144  006004                          ROR     R4
378 002146  020027  000004          10$:    CMP     R0,#4
379 002152  002772                          BLT     9$
380 002154  020327  050000                  CMP     R3,#50000
381 002160  103363                          BHIS    8$
382 002162  062704  001250                  ADD     #1250,R4
383 002166  103010                  RESET1: BCC     11$
384 002170  005203                          INC     R3
385 002172  020327  050000                  CMP     R3,#50000
386 002176  103404                          BLO     11$
387 002200  012703  040000                  MOV     #40000,R3
388 002204  005004                          CLR     R4
389 002206  005202                          INC     R2
390 002210  012700  000006          11$:    MOV     #6,R0
391 002214  010346                  12$:    MOV     R3,-(SP)
392 002216  000316                          SWAB    (SP)
393 002220  006016                          ROR     (SP)
394 002222  006016                          ROR     (SP)
395 002224  006016                          ROR     (SP)
396 002226  042716  177760                  BIC     #177760,(SP)
397 002232  062716  000060                  ADD     #'0,(SP)
398 002236  042703  174000                  BIC     #174000,R3
399 002242  004767  000272                  JSR     PC,MUL
400 002246  005300                          DEC     R0
401 002250  003361                          BGT     12$
402 002252  010603                          MOV     SP,R3
403 002254  062703  000014                  ADD     #14,R3
404 002260  020227  177776                  CMP     R2,#-2
405 002264  002404                          BLT     13$
406 002266  001456                          BEQ     18$
407 002270  020227  000006                  CMP     R2,#6
408 002274  002463                          BLT     19$
409 002276  014346                  13$:    MOV     -(R3),-(SP)
410 002300  004737  000000G                 JSR     PC,@#OUTCH
411 002304  012716  000056                  MOV     #'.,(SP)
412 002310  004737  000000G                 JSR     PC,@#OUTCH
413 002314  012704  000005                  MOV     #5,R4
414 002320  014316                  14$:    MOV     -(R3),(SP)
415 002322  004737  000000G                 JSR     PC,@#OUTCH
416 002326  005304                          DEC     R4
417 002330  003373                          BGT     14$
418 002332  012716  000105                  MOV     #'E,(SP)
419 002336  004737  000000G                 JSR     PC,@#OUTCH
420 002342  012716  000053                  MOV     #'+,(SP)
421 002346  005702                          TST     R2
422 002350  100003                          BPL     15$
423 002352  012716  000055                  MOV     #'-,(SP)
424 002356  005402                          NEG     R2
425 002360  004737  000000G          15$:    JSR     PC,@#OUTCH
426 002364  012716  000060                  MOV     #'0,(SP)
427 002370  162702  000012          16$:    SUB     #12,R2
428 002374  100402                          BMI     17$
429 002376  005200                          INC     R0
430 002400  000773                          BR      16$
431 002402  004737  000000G          17$:    JSR     PC,@#OUTCH
```

```
 432 002406  010216                        MOV     R2,(SP)
 433 002410  062716  000072                ADD     #'0+12,(SP)
 434 002414  004737  000000G               JSR     PC,@#OUTCH
 435 002420  000167  000102                JMP     EX14
 436 002424  012716  000056        18$:    MOV     #'.,(SP)
 437 002430  004737  000000G               JSR     PC,@#OUTCH
 438 002434  012716  000060                MOV     #'0,(SP)
 439 002440  004737  000000G               JSR     PC,@#OUTCH
 440 002444  012704  000006        19$:    MOV     #6,R4
 441 002450  010600                        MOV     SP,R0
 442 002452  022027  000060        20$:    CMP     (R0)+,#'0
 443 002456  001002                        BNE     21$
 444 002460  005304                        DEC     R4
 445 002462  000773                        BR      20$
 446 002464  020402                21$:    CMP     R4,R2
 447 002466  003002                        BGT     22$
 448 002470  010204                        MOV     R2,R4
 449 002472  005204                        INC     R4
 450 002474  005202                22$:    INC     R2
 451 002476  005202                        INC     R2
 452 002500  005302                23$:    DEC     R2
 453 002502  001004                        BNE     24$
 454 002504  012716  000056                MOV     #'.,(SP)
 455 002510  004737  000000G               JSR     PC,@#OUTCH
 456 002514  014316                24$:    MOV     -(R3),(SP)
 457 002516  004737  000000G               JSR     PC,@#OUTCH
 458 002522  005304                        DEC     R4
 459 002524  003365                        BGT     23$
 460 002526  005016                EX14:   CLR     (SP)
 461 002530  004737  000000G               JSR     PC,@#PUSHINT
 462 002534  000167  000000G               JMP     RET
 463 002540  010346                MUL:    MOV     R3,-(SP)
 464 002542  010446                        MOV     R4,-(SP)
 465 002544  006304                        ASL     R4
 466 002546  006103                        ROL     R3
 467 002550  006304                        ASL     R4
 468 002552  006103                        ROL     R3
 469 002554  062604                        ADD     (SP)+,R4
 470 002556  005503                        ADC     R3
 471 002560  062603                        ADD     (SP)+,R3
 472 002562  006304                        ASL     R4
 473 002564  006103                        ROL     R3
 474 002566  000207                        RTS     PC
 475 002570  012746  000034        DIV:    MOV     #34,-(SP)
 476 002574  022703  050000        25$:    CMP     #50000,R3
 477 002600  101002                        BHI     26$
 478 002602  062703  130000                ADD     #130000,R3
 479 002606  006104                26$:    ROL     R4
 480 002610  006103                        ROL     R3
 481 002612  005316                        DEC     (SP)
 482 002614  003367                        BGT     25$
 483 002616  042703  170000                BIC     #170000,R3
 484 002622  005726                        TST     (SP)+
 485 002624  000207                        RTS     PC
```

```
                                        ;
                                        ; MC 15 ... get a string
                                        ;
002626  022765  000001  000004  MC15:   CMP     #1,4(R5)
002634  001402                          BEQ     1$
002636  000167  000024                   JMP     MC98
002642  004767  000000G         1$:     JSR     PC,TOPTOI
002646  010016                          MOV     R0,(SP)
002650  004737  000000G                 JSR     PC,@#INST
002654  010016                          MOV     R0,(SP)
002656  004737  000000G                 JSR     PC,@#PUSHINT
002662  000167  000000G                 JMP     RET
                                        ;
                                        ; wrong number of arguments ... pop 'em off the stack
                                        ;
002666  005765  000004          MC98:   TST     4(R5)
002672  003405                          BLE     MC99
002674  004767  000000G                 JSR     PC,POP
002700  005365  000004                  DEC     4(R5)
002704  000770                          BR      MC98
                                        ;
                                        ; MC error
                                        ;
002706  012716  000030          MC99:   MOV     #MCERR,(SP)
002712  004737  000000G                 JSR     PC,@#ERROR
002716  005016                          CLR     (SP)
002720  004737  000000G                 JSR     PC,@#PUSHINT
002724  000167  000000G                 JMP     RET
        000001                          .END
```

```
                                        .TITLE  tiny-c/ll-01-03/VECTOR
000000                                  .PSECT  VECTOR
                                ;
                                ; tiny-c/ll installation vector
                                ;
                                        .ENABL  GBL
                                        .GLOBL  INCH,INST,CHRDY,OUTCH
                                        .GLOBL  FLOAD,FOPEN,FREAD,FWRITE,FCLOSE
                                        .GLOBL  USERMC,PRBEGN,PRDONE,STBEGIN
                                ;
000000  000167  000000G       INCH:   JMP     GETCHR
000004  000167  000000G       INST:   JMP     GETSTR
000010  000167  000000G       CHRDY:  JMP     TSTCHR
000014  000167  000000G       OUTCH:  JMP     PUTCHR
000020  000167  000000G       FLOAD:  JMP     LOAD
000024  000167  000000G       FOPEN:  JMP     OPEN
000030  000167  000000G       FREAD:  JMP     GETBL
000034  000167  000000G       FWRITE: JMP     PUTBL
000040  000167  000000G       FCLOSE: JMP     CLOSE
000044  000207                USERMC: RTS     PC
000046  000000                        .WORD   0
000050  000207                PRBEGN: RTS     PC
000052  000000                        .WORD   0
000054  000207                PRDONE: RTS     PC
000056  000000                        .WORD   0
000060  000167  000000G       STBEGIN:JMP     MYBEGN
        000001                          .END
```

```
                                        .TITLE  tiny-c/ll-01-03/TCIO
000000                                  .PSECT  TCIO
                                        .ENABL  GBL
                                ;
                                ; tiny-c/ll input/output error codes
                                ;
                                        LOADERR = 25.
                                        KILL    = 99.
                                ;
                                ; tiny-c/ll input/output programmed requests
                                ;
                                        .MCALL  .TTYIN,.TTINR,.TTYOUT
                                        .MCALL  .LOOKUP,.ENTER,.WRITW,.READW,.CLOSE
                                ;
                                ; tiny-c/ll input/output routines
                                ;
                                        .GLOBL  GETCHR,GETSTR,TSTCHR,PUTCHR,MYBEGN
                                        .GLOBL  LOAD,OPEN,GETBL,PUTBL,CLOSE
                                ;
                                ; return a character from the console terminal in r0
                                ;
000000  005000                  GETCHR: CLR     R0
000002  005767  001630                  TST     CHRBUF
000006  001405                          BEQ     1$
000010  016700  001622                  MOV     CHRBUF,R0
000014  005067  001616                  CLR     CHRBUF
000020  000405                          BR      2$
000022  052737  050000  000044  1$:     BIS     #50000,@#44
000030                                  .TTYIN
000030  104340                          EMT     ^O340
000032  103776                          BCS     .-2.
000034  122700  000004          2$:     CMPB    #4,R0
000040  001002                          BNE     3$
000042  005000                          CLR     R0
000044  000403                          BR      4$
000046  122700  000015          3$:     CMPB    #15,R0
000052  001763                          BEQ     1$
000054  042737  050000  000044  4$:     BIC     #50000,@#44
000062  000207                          RTS     PC
```

```
                                        ;
                                        ; get a string from the console terminal
                                        ;
000064  004567  000000G         GETSTR: JSR     R5,SAVE
000070  016501  000004                  MOV     4(R5),R1
000074  012702  177777                  MOV     #-1,R2
000100  052737  040000  000044          BIS     #40000,@#44
000106                          1$:     .TTYIN
000106  104340                          EMT     ^O340
000110  103776                          BCS     .-2.
000112  122700  000004                  CMPB    #4,R0
000116  001002                          BNE     2$
000120  005000                          CLR     R0
000122  000403                          BR      3$
000124  022700  000015          2$:     CMP     #15,R0
000130  001766                          BEQ     1$
000132  110011                  3$:     MOVB    R0,(R1)
000134  005201                          INC     R1
000136  005202                          INC     R2
000140  022700  000012                  CMP     #12,R0
000144  001360                          BNE     1$
000146  042737  040000  000044          BIC     #40000,@#44
000154  005301                          DEC     R1
000156  105011                          CLRB    (R1)
000160  010200                          MOV     R2,R0
000162  000167  000000G                 JMP     RET
                                        ;
                                        ; return a copy of a character from the terminal in r0
                                        ;
000166  005000                  TSTCHR: CLR     R0
000170  052737  050000  000044          BIS     #50000,@#44
000176                                  .TTINR
000176  104340                          EMT     ^O340
000200  042737  050000  000044          BIC     #50000,@#44
000206  005700                          TST     R0
000210  001403                          BEQ     1$
000212  010067  001420                  MOV     R0,CHRBUF
000216  000207                          RTS     PC
000220  016700  001412          1$:     MOV     CHRBUF,R0
000224  000207                          RTS     PC
                                        ;
                                        ; send the character in 2(sp) to the console terminal
                                        ;
000226  052737  040000  000044  PUTCHR: BIS     #40000,@#44
000234  122766  000012  000002          CMPB    #12,2(SP)
000242  001004                          BNE     4$
000244  012700  000015                  MOV     #15,R0
000250                                  .TTYOUT
000250  104341                          EMT     ^O341
000252  103776                          BCS     .-2.
```

```
  83 000254  016600  000002          4$:     MOV     2(SP),R0
  84 000260                                  .TTYOUT
     000260  104341                          EMT     ^O341
     000262  103776                          BCS     .-2.
  85 000264  042737  040000  000044        BIC     #40000,@#44
  86 000272  000207                          RTS     PC
  87                                 ;
  88                                 ; load tcload.tcf into the program space
  89                                 ;
  90 000274  004567  000000G         LOAD:   JSR     R5,SAVE
  91 000300                                  .LOOKUP #CSW,#1,#TCLOAD
     000300  012700  001660'                 MOV     #CSW,%0
     000304  012710  000401                  MOV     #1+<1*^O400>,(0)
     000310  012760  001650' 000002         MOV     #TCLOAD,2.(0)
     000316  104375                          EMT     ^O375
  92 000320  103005                          BCC     5$
  93 000322  012716  000031                  MOV     #LOADERR,(SP)
  94 000326  004737  000000G                 JSR     PC,@#ERROR
  95 000332  000460                          BR      11$
  96 000334  005001                  5$:     CLR     R1
  97 000336                          6$:     .READW  #CSW,#1,#BUFFER,#256.,R1
     000336  012700  001660'                 MOV     #CSW,%0
     000342  012710  004001                  MOV     #1+<8.*^O400>,(0)
     000346  010160  000002                  MOV     R1,2.(0)
     000352  012760  001740' 000004         MOV     #BUFFER,4.(0)
     000360  012760  000400  000006         MOV     #256.,6.(0)
     000366  005060  000010                  CLR     8.(0)
     000372  104375                          EMT     ^O375
  98 000374  103422                          BCS     9$
  99 000376  005201                          INC     R1
 100 000400  012702  001740'                 MOV     #BUFFER,R2
 101 000404  006300                          ASL     R0
 102 000406  122712  000015          7$:     CMPB    #15,@R2
 103 000412  001407                          BEQ     8$
 104 000414  122712  000000                  CMPB    #00,@R2
 105 000420  001404                          BEQ     8$
 106 000422  111277  000000G                 MOVB    @R2,@CURSOR
 107 000426  005267  000000G                 INC     CURSOR
 108 000432  005300                  8$:     DEC     R0
 109 000434  003740                          BLE     6$
 110 000436  005202                          INC     R2
 111 000440  000762                          BR      7$
 112 000442  105737  000052          9$:     TSTB    @#52
 113 000446  001405                          BEQ     10$
 114 000450  012716  000031                  MOV     #LOADERR,(SP)
 115 000454  004737  000000G                 JSR     PC,@#ERROR
 116 000460  000405                          BR      11$
 117 000462  016767  000000G 000000G 10$:     MOV     CURSOR,PROGEND
 118 000470  005367  000000G                 DEC     PROGEND
 119 000474                          11$:    .CLOSE  #1
     000474  012700  003001                  MOV     #1+<6.*^O400>,%0
     000500  104374                          EMT     ^O374
 120 000502  000167  000000G                 JMP     RET
```

```
121                                     ;
122                                     ; open file 4(sp) of size 6(sp) as unit 8(sp) in mode 2(sp)
123                                     ;
124 000506  004567  000000G         OPEN:   JSR     R5,SAVE
125 000512  016503  000012                  MOV     12(R5),R3
126 000516  002004                          BGE     11$
127 000520  012700  177777          10$:    MOV     #-1,R0
128 000524  000167  000000G                 JMP     RET
129 000530  022703  000020          11$:    CMP     #20,R3
130 000534  003771                          BLE     10$
131 000536  062765  000777  000010         ADD     #511.,10(R5)
132 000544  116504  000011                  MOVB    11(R5),R4
133 000550  006204                          ASR     R4
134 000552  016516  000006                  MOV     6(R5),(SP)
135 000556  004737  001422'                JSR     PC,@#CRAD50
136 000562  010067  001052                  MOV     R0,FILEN0
137 000566  016516  000006                  MOV     6(R5),(SP)
138 000572  062716  000004                  ADD     #4,(SP)
139 000576  004737  001422'                JSR     PC,@#CRAD50
140 000602  010067  001034                  MOV     R0,FILEN1
141 000606  016516  000006                  MOV     6(R5),(SP)
142 000612  062716  000007                  ADD     #7,(SP)
143 000616  004737  001422'                JSR     PC,@#CRAD50
144 000622  010067  001016                  MOV     R0,FILEN2
145 000626  016516  000006                  MOV     6(R5),(SP)
146 000632  062716  000013                  ADD     #11.,(SP)
147 000636  004737  001422'                JSR     PC,@#CRAD50
148 000642  010067  001000                  MOV     R0,FILEN3
149 000646  022765  000001  000004         CMP     #1,4(R5)
150 000654  001022                        BNE     12$
151 000656                                  .LOOKUP #CSW,R3,#FILEN0
    000656  012700  001660'                MOV     #CSW,%0
    000662  012710  000400                  MOV     #1*^O400,(0)
    000666  110310                          MOVB    R3,(0)
    000670  012760  001640' 000002         MOV     #FILEN0,2.(0)
    000676  104375                          EMT     ^O375
152 000700  103436                          BCS     13$
153 000702  006303                          ASL     R3
154 000704  012763  177777  001700'        MOV     #-1,BLOCK(R3)
155 000712  070027  001000                  MUL     #512.,R0
156 000716  000167  000000G                 JMP     RET
157 000722  022765  000002  000004  12$:    CMP     #2,4(R5)
158 000730  001022                        BNE     13$
```

```
    159 000732                                      .ENTER  #CSW,R3,#FILEN0,R4
        000732  012700  001660'                    MOV     #CSW,%0
        000736  012710  001000                      MOV     #2.*^O400,(0)
        000742  110310                              MOVB    R3,(0)
        000744  012760  001640' 000002             MOV     #FILEN0,2.(0)
        000752  010460  000004                      MOV     R4,4.(0)
        000756  104375                              EMT     ^O375
    160 000760  103406                              BCS     13$
    161 000762  006303                              ASL     R3
    162 000764  012763  177777  001700'             MOV     #-1,BLOCK(R3)
    163 000772  000167  000000G                     JMP     RET
    164 000776  012700  177777          13$:        MOV     #-1,R0
    165 001002  105737  000052                      TSTB    @#52
    166 001006  001402                              BEQ     14$
    167 001010  012700  177776                      MOV     #-2,R0
    168 001014  000167  000000G         14$:        JMP     RET
    169                                 ;
    170                                 ; read a 512 byte data block from unit 4(sp) into 2(sp)
    171                                 ;
    172 001020  004567  000000G         GETBL:      JSR     R5,SAVE
    173 001024  016501  000004                      MOV     4(R5),R1
    174 001030  016503  000006                      MOV     6(R5),R3
    175 001034  006303                              ASL     R3
    176 001036  005263  001700'                     INC     BLOCK(R3)
    177 001042  016304  001700'                     MOV     BLOCK(R3),R4
    178 001046  006203                              ASR     R3
    179 001050                                      .READW  #CSW,R3,#BUFFER,#256.,R4
        001050  012700  001660'                    MOV     #CSW,%0
        001054  012710  004000                      MOV     #8.*^O400,(0)
        001060  110310                              MOVB    R3,(0)
        001062  010460  000002                      MOV     R4,2.(0)
        001066  012760  001740' 000004             MOV     #BUFFER,4.(0)
        001074  012760  000400  000006             MOV     #256.,6.(0)
        001102  005060  000010                      CLR     8.(0)
        001106  104375                              EMT     ^O375
    180 001110  103417                              BCS     17$
    181 001112  005000                              CLR     R0
    182 001114  012702  001740'                     MOV     #BUFFER,R2
    183 001120  122712  000004          15$:        CMPB    #4,(R2)
    184 001124  001407                              BEQ     16$
    185 001126  111211                              MOVB    (R2),(R1)
    186 001130  005200                              INC     R0
    187 001132  005201                              INC     R1
    188 001134  005202                              INC     R2
    189 001136  022700  001000                      CMP     #512.,R0
    190 001142  003366                              BGT     15$
    191 001144  000167  000000G         16$:        JMP     RET
    192 001150  012700  177777          17$:        MOV     #-1,R0
    193 001154  105737  000052                      TSTB    @#52
    194 001160  001402                              BEQ     18$
    195 001162  012700  177776                      MOV     #-2,R0
    196 001166  000167  000000G         18$:        JMP     RET
```

```
                                        ;
                                        ; write data from 2(sp) to 4(sp) onto unit 6(sp)
                                        ;
001172  004567  000000G         PUTBL:  JSR     R5,SAVE
001176  162706  000004                 SUB     #4,SP
001202  012765  000004  177766         MOV     #4,-12(R5)
001210  016501  000004                 MOV     4(R5),R1
001214  016502  000006                 MOV     6(R5),R2
001220  016503  000010                 MOV     10(R5),R3
001224  160102                         SUB     R1,R2
001226  062702  000001                 ADD     #1,R2
001232  022702  001000                 CMP     #512.,R2
001236  003412                         BLE     19$
001240  010104                         MOV     R1,R4
001242  060204                         ADD     R2,R4
001244  010465  177770                 MOV     R4,-10(R5)
001250  111465  177766                 MOVB    (R4),-12(R5)
001254  112714  000004                 MOVB    #4,(R4)
001260  062702  000002                 ADD     #2,R2
001264  006202                 19$:    ASR     R2
001266  006303                         ASL     R3
001270  005263  001700'                INC     BLOCK(R3)
001274  016304  001700'                MOV     BLOCK(R3),R4
001300  006203                         ASR     R3
001302                                 .WRITW  #CSW,R3,R1,R2,R4
001302  012700  001660'                MOV     #CSW,%0
001306  012710  004400                 MOV     #9.*^O400,(0)
001312  110310                         MOVB    R3,(0)
001314  010460  000002                 MOV     R4,2.(0)
001320  010160  000004                 MOV     R1,4.(0)
001324  010260  000006                 MOV     R2,6.(0)
001330  005060  000010                 CLR     8.(0)
001334  104375                         EMT     ^O375
001336  103413                         BCS     22$
001340  006300                         ASL     R0
001342  122765  000004  177766 20$:    CMPB    #4,-12(R5)
001350  001404                         BEQ     21$
001352  016504  177770                 MOV     -10(R5),R4
001356  116514  177766                 MOVB    -12(R5),(R4)
001362  000167  000000G         21$:    JMP     RET
001366  012700  177777          22$:    MOV     #-1,R0
001372  000763                         BR      20$
                                        ;
                                        ; close unit 2(sp)
                                        ;
001374  004567  000000G         CLOSE:  JSR     R5,SAVE
001400  016503  000004                 MOV     4(R5),R3
001404                                 .CLOSE  R3
001404  012700  003000                 MOV     #6.*^O400,%0
001410  150300                         BISB    R3,%0
001412  104374                         EMT     ^O374
001414  005000                         CLR     R0
001416  000167  000000G                JMP     RET
```

```
                                        ;
                                        ; convert 3 ascii characters at 2(sp) to RAD50 representation in R0
                                        ;
001422  005000                  CRAD50: CLR     R0
001424  122776  000040  000002         CMPB    #40,@2(SP)
001432  001414                         BEQ     24$
001434  117601  000002                 MOVB    @2(SP),R1
001440  162701  000022                 SUB     #22,R1
001444  022701  000050                 CMP     #50,R1
001450  003002                         BGT     23$
001452  162701  000116                 SUB     #116,R1
001456  070127  003100         23$:    MUL     #3100,R1
001462  060100                         ADD     R1,R0
001464  005266  000002         24$:    INC     2(SP)
001470  122776  000040  000002         CMPB    #40,@2(SP)
001476  001414                         BEQ     26$
001500  117601  000002                 MOVB    @2(SP),R1
001504  162701  000022                 SUB     #22,R1
001510  022701  000050                 CMP     #50,R1
001514  003002                         BGT     25$
001516  162701  000116                 SUB     #116,R1
001522  070127  000050         25$:    MUL     #50,R1
001526  060100                         ADD     R1,R0
001530  005266  000002         26$:    INC     2(SP)
001534  122776  000040  000002         CMPB    #40,@2(SP)
001542  001412                         BEQ     28$
001544  117601  000002                 MOVB    @2(SP),R1
001550  162701  000022                 SUB     #22,R1
001554  022701  000050                 CMP     #50,R1
001560  003002                         BGT     27$
001562  162701  000116                 SUB     #116,R1
001566  060100                 27$:    ADD     R1,R0
001570  000207                 28$:    RTS     PC
```

```
                                        ;
                                        ; check for application program interrupt
                                        ;
001572  005767  000000G         MYBEGN: TST     APPLVL
001576  001001                         BNE     1$
001600  000207                         RTS     PC
001602  004767  000000G         1$:     JSR     PC,CHRDY
001606  122700  000033                 CMPB    #33,R0
001612  001401                         BEQ     2$
001614  000207                         RTS     PC
001616  012746  000143          2$:     MOV     #KILL,-(SP)
001622  004737  000000G                 JSR     PC,@#ERROR
001626  005726                         TST     (SP)+
001630  004767  000000G                 JSR     PC,INCH
001634  000207                         RTS     PC
001636  000000                 CHRBUF: .WORD   0
001640  000000                 FILEN0: .RAD50  /   /
001642  000000                 FILEN1: .RAD50  /   /
001644  000000                 FILEN2: .RAD50  /   /
001646  000000                 FILEN3: .RAD50  /   /
001650  075250  076604  056754 TCLOAD: .RAD50  /SY TCLOADTCF/
001656  076576
001660                         CSW:    .BLKW   10
001700                         BLOCK:  .BLKW   20
001740                         BUFFER: .BLKW   256.
        000001                         .END
```