NAME

ato?? -- ASCII to machine format conversion

SYNOPSIS

```
ato??(s1,v1)
char *s1;
int *v1;

/*
 'v' may be int or long depending on whether the function
 converts to int or long
*/
```

DESCRIPTION

ato?? describes a family of 10 functions which convert ASCII string input to word or double word binary numeric representation. The first five functions convert the string to word or integer format. The second five functions convert the string to double word or long format. The following is a list of the subroutine names:

- atob - binary
- atod - signed decimal
- atoo - octal
- atou - unsigned decimal
- atox - hexadecimal
- atolb - long binary
- atold - long signed decimal
- atolo - long octal
- atolu - long unsigned decimal
- atolx - long hexadecimal

These functions return an integer indicating the length of the string s1 if no error occurred. If an error occurred, the value returned is zero. The value returned is the same as would be returned by the len function.

s1 is the string to be evaluated. This string contains the ASCII representation of the number for the conversion.

v1 is a pointer to an integer variable or a pointer to a long integer variable depending upon whether the function converts to integer or long. If an error occurs, the data pointed to by v1 is always set to zero. If the pointer v1 is zero, the data pointed to by v1 is not modified and the function returns a zero.

The input convention for the string s1 for the signed decimal functions is as follows:

```
[<blanks>][<sign>][<leading zeros>]<numeric characters><null>
where <sign> ::= + | -
```

The input convention for the string s1 for unsigned functions is as follows:

[<blanks>][<leading zeros>]<numeric characters><null>

There are six reasons for the error condition.

(A) s1 points to an empty string.

(B) s1 points to a string containing only blanks.

(C) s1 points to a string which contains characters out of the range required by the conversion. For example, atob will accept only zeros and ones. The legal characters for hexadecimal conversions are "012345789ABCDEF".

(D) s1 points to a string whose format is illegal. For example, the string contains trailing blanks.

(E) v1, the pointer to the word or double word, has the address zero.

(F) s1 points to a string whose numeric value exceeds that of the conversion type.

The ranges of each of the conversion types is listed

atob	-	0 to 1111111111111111
atod	-	-32768 to 32767
atoo	-	0 to 177777
atou	-	0 to 65535
atox	-	0 to FFFF
atolb	-	0 to a string of 32 one's
atold	-	-2147483648 to 2147483647
atolo	-	0 to 37777777777
atolu	-	0 to 4294967295
atolx	-	0 to FFFFFFFF

LIBRARY

/lib/lib3.a

SEE ALSO

??toa(3L) (listed alphabetically as toa(3L))