Package 20001 - Email Tools
Base 64 (*web/base64)
Details

Calling Sequence (must be made to a specific Entry Point):

Entry Points Available for converting Strings:

```
CALL "*WEB/BASE64;ENCODE_STR",instring$,outstring$
CALL "*WEB/BASE64;DECODE_STR",instring$,outstring$
```

Entry Points Available for converting entire files:

```
CALL "*WEB/BASE64;ENCODE_FILE",infile$,outfile$
CALL "*WEB/BASE64;ENCODE_FILE",infile$,outfile$,blocksize
CALL "*WEB/BASE64; ENCODE_FILE", infile$,outfile$, blocksize, EndOfLine$
CALL "*WEB/BASE64;DECODE_FILE",infile$,outfile$
```

Purpose of the Program:
The Base64 algorithm encodes 3 bytes (value 0-255 each)
to 4 bytes (value 0-63 each) using plain ASCII characters
A-Z,a-z,0-9,+/
This allows binary files to be transmitted within text only
protocols across the Internet, e.g. e-mail, authorization
field in web pages. Base64 is most commonly used for MIME
Encoding of Email Attachments.

Usage Details:
When encoding and decoding files, both the infile$ and outfile$
may contain channel numbers which are currently open to files,
or they may contain physical file names. This allows for a file
on disk, to be encoded to a *memory* file for later program handling.

The only channels closed will be the ones this program opens.
If a channel number is passed in either infile$ or outfile$
then those channels will remain open.

The encode file allows for a 2 extra parameters.
1st is a block size to output the data at.
This won't have any affect on a disk file. But does allow the data
to be blocked into a *memory* file, each write record, will be a
separate index. Use of this block size would allow you to specify
how big those encoded records will be. The primary purpose for this
was for email, since the lines need to be less than or equal to 998
bytes each. If block size is specified, then it will be automatically
segmented for you. If no block size is given, it will default to 4096
byte blocks.
The 2nd parameter EndOfLine$, allows you to pass an end-of-line
terminator. The EndOfLine$ should be either $0a$ or $0d0a$. The
EndOfLine$ is only used for encoding the file. An EndOfLine$ if
given will be appended to each "block". If no block size is passed,
then no The EndOfLine$ will be appended.

Note on File Position (affects ENCODE/DECODE FILE only.) The incoming file, will be read from its beginning.

The output file, will be created if it doesn't exist, and appended to if it does. If you use Channel Numbers, then the output channel will be appended to. This allows the output to be tacked on to the end of an ongoing encryption/decryption.

If this program encounters an error during processing then it will EXIT ERR.

Errors Returned:

Typical File open errors, for either infile$ or outfile$ -

0-Busy/perms, 12-cant find, etc

Handling Errors:

46 - string when encoded, will exceed maximum string length.

48 - Invalid Data, Unknown Character encountered during decode, indicates corrupted data, or data not in base64 format.