uMON Command List and Basic Usage Guide

UMON is a simple monitor that runs w/o the need of any ram. It uses three registers for a stack space for commands.

In order to push items onto the 'stack' you simply enter a number at the | prompt.

0ee00000

This will push the value 0x0ee00000 onto the top of the stack and push any other values currently on the stack down the stack.

If this was the stack prior to the above:

AA55AA55 <- Top

00001000

00000000 <- Bottom

After entering the number above the stack would be:

0EE00000 <- Top

AA55AA55

00001000 <- Bottom

NOTE: Entering any number that begins with a Hex digit (A-F) must be preceded by a 0.

Example: AA55AA55 would have to be entered as 0AA55AA55

At any time you can enter a . and it will dump out the current stack top->middle->bottom

Most commands can be stopped by pressing s on the console.
This will bring up a prompt:
Type '^q' or 'q' to continue, '.' to terminate.

UMON Command List

Compare

Compare data command. This command compares data to memory
locations pointed to by address range contained on the top of stack.

Usage:

xxxxx - Compare start address
yyyyy - Compare last address
zzzzz - Compare pattern
c

If a non-match is found it will print to the console the address and the mismatched data.

If all data matches it will return to the prompt.

Examine

Examine data command. This command fetches and prints the data
from the memory location pointed to by address contained on the
top of stack. It will then let you modify the data at the
address or step to the next data in memory. The address on the stack
is incremented by the size of the item fetched.

It has three usages one for a byte, half word, word and doublw word

Usage:

xxxx – Address to examine
eb - fetches a byte

xxxx – Address to examine
eh - fetches a half word

xxxx – Address to examine
ew - fetches a word

xxxx – Address to examine
ed - fetches a double word

Increment

Increment fill data command. This command stores data to memory locations pointed to by address range contained on the top of stack. It will increment the pattern by 1 for each location stored too

Usage:
xxxxx - fill start address
yyyy - fill last address
zzzzz - Fill pattern

ib - fill with bytes

ih - fill with shorts

iw - fill with words

Icompare

increment compare data command. This command compares data to memory locations pointed to by address range contained on the top of stack. It will increment the pattern by 1 for each location compared

Usage:
xxxxx - Fill start address
yyyyy - fill last address
zzzzz - Fill pattern

nb - fill with bytes

nh - fill with shorts

nw - fill with words

Fill

Fill data command. This command stores data to memory locations pointed to by address range contained on the top of stack. It fills on a word boundary

Usage:
xxxxx - Fill start address
yyyyy - fill last address
zzzzz - Fill pattern
f

Jump

This code segment jump to the location pointed to the top of the stack

Usage:

xxxx – Jump Address
j

Load

Load data command. This command fetches and prints the data from the memory location pointed to by address contained in the top of stack. The address on the stack is incremented by the size of the item fetched.

Usage:

Xxxx – Address to start loading from

lb - fetches a byte

lh - fetches a half word

lw - fetches a word

Memrchk

Memory test loop. The range of the loop is specified by TOS and NOS. This test first writes a pattern through the range of address and then reads and compares to see if the pattern is correct. This will test address lines and if refreash is happening often enough. A good test is to pick a large range (1 meg+) and test it for bytes, half words and words. Let run over night each time. Pattern starts at 0x00 and increments by 1 for each address written.

Usage:

xxxxx - start address
yyyyy - end address

mb - writes a byte

mh - writes a half word

mw - writes a word

Read

This code segment enters into an endless loop to read the
location pointed to by the TOS.
NOTE: There is no way to break out of this cmd other than a power cycle.

Usage:

xxxx – Address to read forever and ever..

rb - fetches a byte

rh - fetches a half word

rw - fetches a word

rd – fetches a double word

Store

Store data command. This command stores the data on the top of the stack into memory by the location point to by address contained in the next to the top of stack. The stack is popped so that only the address remains on the stack. The address on the stack is incremented by the size of the item stored.
NOTE: This can be used in a script to program a series of registers with values

Usage:

xxxx – Address to store to
yyyy – Data to store

sb - stores a byte

sh - stores a half word

sw - stores a word

Trans

Transfer data command. This command transfers data between
locations pointed to by address range contained on the top of stack.

Usage:

xxxxx - block source start address
yyyyy - block source last address
zzzzz - block destination address
t

Write

Endless loop to write the value in TOS to the location pointed to by the NOS.
NOTE: There is no way to break out of this command other than a power cycle

Usage:

xxxx - Write Address
yyyy – Data to Write

wb - writes a byte

wh - writes a half word

ww - writes a word

Memrx

This code segment enters into an memory test loop. The range of the loop
is specified by TOS (S2) and NOS (S1). This test writes a word and immediately
reads it to test for refreash arbitration problems. To catch an arbitration
problem set the logic analyzer to trigger on the branch to memerror.
The write or read that caused the problem will be close by.

Usage:

xxxx – Start Address
yyyy – End Address
x

Printstk

This code segment prints whats in the stack and leaves the stack alone.

Usage:

Dump

Dump data command. This command fetches and prints the data from the memory locations pointed to by address range contained on the top of stack.

Usage:

xxxxx - Dump start address
yyyyy - Dump last address
d