QNICE monitor function documentation

Auto generated

December 29, 2015

Chapter 1

CHR

1.1 CHR$TO_UPPER

CHR$TO_UPPER expects a character to be converted to upper case in R8

Chapter 2

DBG

2.1 DBG$DISASM

DBG$DISASM disassembles a single instruction pointed to by R8. R8 will be incremented according to the addressing modes used in the instruction.
R8: Contains the address of the instruction to be disassembled
R8 WILL BE CHANGED BY THIS FUNCTION!

Chapter 3

IO

3.1 IO$DUMP_MEMORY

IO$DUMP_MEMORY prints a hexadecimal memory dump of a specified memory region.
R8: Contains the start address
R9: Contains the end address (inclusive)
The contents of R8 and R9 are preserved during the run of this function.

3.2 IO$GETCHAR

IO$GETCHAR reads a character either from the first UART in the system or from an
attached USB keyboard. This depends on the setting of bit 0 of the switch register.
If SW[0] == 0, then the character is read from the UART, otherwise it is read from
the keyboard data register.
R8 will contain the character read in its lower eight bits.

3.3 IO$GETS

IO$GETS reads a CR/LF terminated string from the serial line
R8 has to point to a preallocated memory area to store the input line

3.4 IO$GET_W_HEX

IO$GET_W_HEX reads four hex nibbles from stdin and returns the corresponding
value in R8
Illegal characters (not 1-9A-F or a-f) will generate a bell signal. The only exception to this behaviour is the character 'x' which will erase any input up to this point. This has the positive effect that a hexadecimal value can be entered as 0x.... or just as ....

3.5 IO$PUTCHAR

IO$PUTCHAR prints a single character.
R8: Contains the character to be printed
The contents of R8 are being preserved during the run of this function.

3.6 IO$PUTS

IO$PUTS prints a null terminated string.
R8: Pointer to the string to be printed. Of each word only the lower eight bits will be printed. The terminating word has to be zero.
The contents of R8 are being preserved during the run of this function.

3.7 IO$PUT_CRLF

IO$PUT_CRLF prints actually a LF/CR (the reason for this is that curses on the
MAC, where the emulation currently runs, has problems with CR/LF, but not with LF/CR)

3.8 IO$PUT_W_HEX

IO$PUT_W_HEX prints a machine word in hexadecimal notation.
R8: Contains the machine word to be printed in hex notation.

The contents of R8 are being preserved during the run of this function.

Chapter 4

KBD

4.1 KBD$GETCHAR

KBD$GETCHAR reads a character from the USB-keyboard.
R8 will contain the character read in its lower eight bits.

Chapter 5

MEM

5.1 MEM$FILL

MEM$FILL fills a block of memory running from the address stored in R8. R9 contains the number of words to be written. R10 contains the value to be stored in the memory area.

5.2 MEM$MOVE

MEM$MOVE moves the memory area starting at the address contained in R8 to the area starting at the address contained in R9. R10 contains the number of words to be moved.

Chapter 6

MISC

6.1 MISC$WAIT

MISC$WAIT
Waits for several milliseconds, controlled by R8. This is just a stupid wait loop as we as of now do not have hardware timers and interrupts.
R8: Contains delay value

Chapter 7

MTH

7.1 MTH$MUL

MTH$MUL performs a highly unelegant signed 16 x 16 multiplication of the form R11(H)/R10(L) = R8 * R9.
(R2 = XL, R3 = XH) = (R8 = —A—), initally
R4 = —B—
R10 = RL, R11 = RH (result)
R5 = counter

Chapter 8

STR

8.1 STR$CHOMP

STR$CHOMP removes a trailing LF/CR from a string pointed to by R8

8.2 STR$CMP

STR$CMP compares two strings
R8: Pointer to the first string (S0),
R9: Pointer to the second string (S1),
R10: negative if (S0 ¡ S1), zero if (S0 == S1), positive if (S0 ¿ S1)
The contents of R8 and R9 are being preserved during the run of this function

8.3 STR$LEN

STR$LEN expects the address of a string in R8 and returns its length in R9

8.4 STR$STRCHR

STR$STRCHR seaches for the first occurrence of the character stored in R8 in
a
string pointed to by R9.
R8: Pointer to the string

R9: Character to be searched
R10: Zero if the character has not been found, otherwise it contains a pointer to the first occurrence of the character in the string
The contents of R8 and R9 are being preserved during the run of this function

8.5 STR$TO_UPPER

STR$TO_UPPER expects the address of a string to be converted to upper case in R8

Chapter 9

UART

9.1 UART$GETCHAR

UART$GETCHAR reads a character from the first UART in the system.
R8 will contain the character read in its lower eight bits.

9.2 UART$PUTCHAR

UART$PUTCHAR writes a single character to the serial line.
R8: Contains the character to be printed
The contents of R8 are being preserved during the run of this function.

Chapter 10

VGA

10.1 VGA$CHAR_AT_XY

VGA$CHAR_AT_XY
R8: Contains character to be printed
R9: X-coordinate (0 .. 79)
R10: Y-coordinate (0 .. 39)
Output a single char at a given coordinate pair.

10.2 VGA$INIT

VGA$INIT
VGA on, hardware cursor on, large, blinking, reset current character coordinates

10.3 VGA$PUTCHAR

VGA$PUTCHAR
Print a character to the VGA display. This routine automatically increments the
X- and, if necessary, the Y-coordinate. No scrolling is currently implemented - if the end of the screen is reached, the next character will be printed at location (0, 0) again. I.e. this functions implements a rather crude type-writer.
This routine relies on the stored coordinates VGA$X and VGA$Y which always contain
the coordinate of the next (!) character to be displayed and will be updated

accordingly. This implies that it is possible to perform other character output and
cursor coordinate manipulation between two calls to VGA$PUTC without disturbing
the position of the next character to be printed.
R8: Contains the character to be printed.