## GDB Primer

### Displaying Registers

info reg *or* info reg name  
*Example:* info reg pc

### Changing Registers

set $name = value  
*Example:* set $pc=0

### Displaying Memory

x/*len*Sx {address}, where *len* is the length and S is the size (w=32-bit, h=16-bit, b=8-bit)  
*Examples:* x/32wx  
 x/32wx 0x50112000

### Setting Breakpoints

hb \*address *or* function *or* file:line *or* line (sets a hardware breakpoint)  
b \*address *or* function *or* file:line *or* line (sets a software breakpoint)  
*Examples:* hb \*0x30008000  
 hb init\_irq  
 hb drivers/char/misc/maximasp-gadget.c:122  
 b pass

### Displaying Breakpoints

info break

### Deleting Breakpoints

del break #   
*Examples:* del break 5

### Changing Memory

set {size}address=value, where size is long/short/char  
*Example:* set {long}0xffe54000=0x80000001

#### Memory Access Width

Please note that GDB will default to reading and writing 32-bit quantities. This can be a problem for NAND and other hardware that require a specific sequence of 8-bit or 16-bit accesses. Use *{size}* specifiers:

set {short}0x26000055 = 0x0098  
x/32wx 0x26000000  
set {char}0x2a400000=0x90  
set {char}0x2a800000=0  
x/32wx 0x2a000000

### Stack Frame

Display stack frame: bt  
Select frame: frame #

### Running, Single Stepping

Show current location: l  
Single step, step over: s, n (C instructions); si, ni (assembly)  
Run until stack frame: fin  
Continue running: c  
Disassemble at pc: disass $pc,+32  
Disassembly on step: display/i $pc

### GDB Scripts

Retyping the same commands time and again is error prone. GDB supports scripts, and the following scripts might be useful. Place these scripts into a *.gdbinit* file in your home directory, or the directory you run *arm-none-eabi-gdb* from. gdb will then load the scripts automatically.

define olimex

target remote 127.0.0.1:3333

end

define mrh  
monitor reset halt  
end

define go  
target remote 127.0.0.1:3333  
monitor reset halt  
load  
b pass  
b fail  
c

end