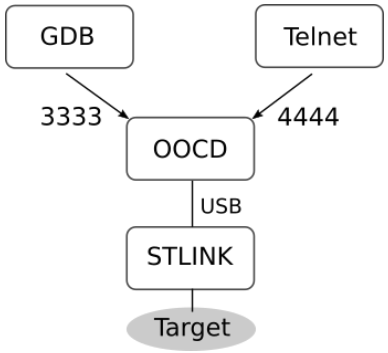
GDB für ES2 Quickref

Rev. 2023-10-17

Hubert.Hoegl@hs-augsburg.de



Programmname
`arm-none-eabi-gdb` oder `gdb-multiarch`

GDB Home
`https://www.gnu.org/software/gdb`

GDB Online Doku
`https://www.sourceware.org/gdb/documentation`

GDB Übung
`https://gitlab.informatik.hs-augsburg.de/es2/es2-nucf446/gdb-uebung`
`https://gitlab.informatik.hs-augsburg.de/es2/es2-nucl476/gdb-uebung`

Lokale GDB Doku
`apt install gdb-doc`, `info gdb`

Optionen: `gdb --help`
Starten: `--tui` Text User Interface
Hilfe:
`(gdb) info`
`(gdb) help`

Beenden: `(gdb) quit`

Startup File:
`<projekt>/.estool/gdbinit`
`~/.gdbinit`

```
estool --start-oocd
estool --stop-oocd
estool --ask-oocd
```

C- Präfix bedeutet **Strg** Taste!

`C-L` GUI Refresh

C-c Laufendes Programm auf dem STM32 abbrechen, so dass GDB die Steuerung übernehmen kann. Nun sieht man den GDB Prompt: `(arm-gdb)`

`(arm-gdb) continue` Programm fortsetzen (auch: `cont`, `c`)

`C-o` Focus auf anderes Fenster setzen (mehrfach anwenden)

↑, ↓ Fenster im Fokus scrollen

`C-x 2` nächstes Layout wählen (mehrfach anwenden)

`C-x 1` nur ein Fenster (`src` oder `asm`)

`C-x o` Focus

Vier Fenster: GDB Kommandos: `cmd`, C Quelltext: `src`, Register: `reg`, Assembler: `asm`

`set pagination [on|off]` Ausgabe im Kommandofenster seitenweise betrachten. Siehe auch das Pipe Kommando weiter unten.

Die Befehlszeile im Kommandofenster kann mit den *GNU readline* Tasten gesteuert werden: `C-p`, `C-n`, `C-b`, `C-f`, `C-a`, `C-e`, ... (wie bei `bash` und anderen Shells).
Lit.: `https://en.wikipedia.org/wiki/GNU_Readline`

Viele Kommandos, Dateinamen, Labels u.s.w. können über die **Tabulator-Taste** vervollständigt werden ("tab completion", in GNU readline eingebaut).

RETURN-Taste Wiederhole vorheriges Kommando

`n` Aktuelle Zeile im `src` Fenster ausführen ("next").

`ni` Aktuelle Zeile im `asm` Fenster ausführen ("next instruction")

`s` Aktuelle Zeile im `src` Fenster ausführen ("step"). Geht in Funktionen.

`si` Aktuelle Zeile im `asm` Fenster ausführen ("step instruction"). Geht in Funktionen.

Breakpoints

`b main` Breakpoint auf `main()` setzen.
`b 26` Breakpoint auf Zeile 26 setzen.
`b startup_stm321476xx.s:86` Breakpoint auf Datei:Zeile setzen
`b *<addr>` Breakpoint auf Adresse setzen
`b startup.c:label`
`info b` Breakpoints listen
`del <nr>` Breakpoint löschen
`clear` Delete all breakpoints
`dis <nr>` Disable
`en <nr>` Enable
`until <nr>` Run until breakpoint `<nr>`
`br 25 if var >= 10` condition
`cond <bkpt_nr> ...` Set condition for breakpoint nr
Kommandosequenz bei Breakpoint definieren:

```
commands <bkpt_nr>
> ...
end
```

`tbreak` Temporary breakpoint

Quelltext listen

`list <label>` Tab vervollständigt Namen!
`list osc.c:10`
`list osc.c:<label>`

Print

Schreibweise:

```
file::var
file:fct
file:line

fct::var
fct:label
fct:line
```

```
p/x gdbu
p/x &_sidata
p/x $lr
p {int[6]}@array
p/x *(int *) $sp
p/x array@5
p/x fkt::lok_var
p/d
p/s
p/c
p/a
p/t    print binary (t = two)
```

eXamine Memory (dump memory)

```
x/s
x/11b
x/8xw
x/8xw &buffer
x/16xb
x/i
x/i
```

Display

```
display gcount
info display
undisplay
```

Watch

```
rwatch gcount   R
watch gcount    W
awatch gcount   RW
info watch
watch -l <expr>   Adresse im Speicher
del ..., ena ..., dis ...
```

Werte setzen

```
set $r0 = 0x...
set var1 = 42
set {int}...  = 42
set {int}0x2000 = 0xff
set {int}(&buffer + 3) = 42
set *(int *)...  = 42
```

Verschiedenes

```
where
finish    aktuelle Funktion beenden
until <pos>    ausführen bis
call <address>
return <expr>    return immediately
up    One stack frame up
load    Flashen
disas    Disassemblieren
whatis
ptype
alias -a di = disas
source <file>    GDB Kommandos aus Datei lesen
| help p | less    Pipe (ab V9.1)
```

.estool/gdbinit

Funktion `debug-program`:

```
monitor reset halt
continue
```

OpenOCD Telnet

telnet localhost 4444

```
telnet> reset halt
telnet> reset run
telnet> init
telnet> help
```

In gdb mit "monitor"

```
gdb> monitor <oocd-cmd>
```

Macros

GCC: `-g3`, `-ggdb3`

```
macro expand <macro>
info macro <macro>
```

Screen Layout

```
layout split
layout next entspricht C-x 2
layout regs
layout src entspricht C-x 1
layout asm
update
winheight
set tui border-kind  siehe help set tui
```

Einstellungen

```
set pagination on|off
set history save on|off    # .gdb_history
set logging file ...
set logging on|off
set substitute-path <path1> <path2>
```

`C-x s` single key mode

```
c  count      n  next       s  step
d  down       q  quit       u  up
f  fini       r  run        v  info locals
w  where
```

Python Support

```
(gdb) python
(gdb) import sys
(gdb) print(sys.version)
(gdb) end
3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0]

(gdb) source script.py
```