AVR-GDB Tutorial

# Memory mapped abstraction

Since the DDRx, PINx, and PORTx values are mapped to locations in memory, they don’t show up in the symbol table and so the display command won’t work. Instead, we have to inspect the locations in memory themselves using the x command and we need to know the address in memory.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Offset** | **Name** | **Bits** |  |  |  |  |  |  |  |  |
| 0x20 | PINA | 7:0 | PINA7 | PINA6 | PINA5 | PINA4 | PINA3 | PINA2 | PINA1 | PINA0 |
| 0x21 | DDRA | 7:0 | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| 0x22 | PORTA | 7:0 | PORTA7 | PORTA6 | PORTA5 | PORTA4 | PORTA3 | PORTA2 | PORTA1 | PORTA0 |
| ... |  |  |  |  |  |  |  |  |  |  |

Register Summary p. 446 ATmega1284 Datasheet

From this we can get the memory locations for each DDRx, PINx, and PORTx and inspect them using x/[length][format][size modifier] [address expression].

For example, to inspect PINA:

(gdb) x/1xb 0x800020

This can be tedious, so we have developed the commands.gdb script to provide a layer of abstraction to these locations in memory:

printPINx [format]

printPORTx [format]

printDDRx [format]

setPINx [value]

Where [format] can be

* x - hexadecimal
* d - decimal
* t - binary

The commands can be loaded by running

$ avr-gdb -x commands.gdb

Additionally, there is an initDebugger.gdb file that is updated everytime the program is compiled for testing or debugging using the Makefile (make test or make debug). This file (shown below) contains some information for AVR-GDB to set up the debugging environment.

|  |  |  |
| --- | --- | --- |
| Line | initDebugger.gdb | Line descriptionDescription |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | file build/objects/main.elf  target remote :1234  set logging file <filename>  set logging overwrite on  set logging on  set $passed=1  set $failed=0  set $tests=0  break main.c:<#>  commands  silent  ...  end  continue | File to debug  Connect debugger to SimAVR (must be running)  Set the log file to <filename>  Overwrite existing log file  Turn on logging  Some additional information for the test cases  Set break point at line <#> in main.c  Additional commands to execute everytime break point is hit (i.e. printPORTx, display state, etc.) |

If make test fails on a given test case, you will want to step through the execution of the program to see at which point the values error out. There are only a handful of commands that are available for doing this, listed below. To run the debugger set up to debug, you can type:

$make debug

Which will initialize the simulator and run the command

$avr-gdb -x test/commands.gdb -x test/initDebugger.gdb

And leave you in AVR-GDB with a break point at the top of the while(1) ready to debug.

(gdb)

Most relevant GDB commands:

|  |  |
| --- | --- |
| **Command** | **Description** |
| break | Set breakpoint at specified location |
| step | Step program until it reaches a different source line |
| stepi | Step one instruction exactly |
| commands | Set commands to be executed when a breakpoint is hit |
| display EXP | Print value of expressions EXP each time the program stops |
| print EXP | Print value of expression EXP |
| set VAR = EXP | Evaluate expression EXP and assign result to variable VAR |

If you need more, type help to see a list of all the commands.

We have additionally set up the following commands in commands.gdb to aid in debugging

|  |  |
| --- | --- |
| **Command** | **Description** |
| setPINx VAL | Set PINx to VAL (set the input) |
| printPINx | Print the value of PINx (Input value) |
| printDDRx | Print the value of DDRx (Data Direction Register) |
| printPORTx | Print the value of PORTx (output value) |