Avid AVS4000 Command Line Interface (CLI)

This document describes the command line interface (CLI) used with AVS4000 system.

Commands

Commands are entered one per line. Commands are alpha-numeric strings at the beginning of the line. Parameters to the commands are whitespace delimited.

Help

The CLI has a help system available. A list of all available commands can be obtained by doing:

```
AVS4000D>> help
Available Commands:
  adt               alias             cfg               cmd
  debug             help              hist              quit
  restart           su                version
```

Additional help for an additional command can be done by:

```
AVS4000D> help help
Command: help
Group: script
Method: script.CmdHelp(%A)
Description: Display help for commands
Usage:
     help               // List all available commands
     help [key]         // List all commands with 'key' in the command
```

The method shows show the command is executed in Javascript. The parameter substitutions (%0, %1, %V, etc..) follow the same usage as defined by the alias command.

Usage

Usage defines the syntax for a command:

```
Usage:
        cmd <param1> [params2]
```

Where 'cmd' is the command literal that is used. Required parameters are denoted by **<>** and optional parameters are denoted by **[]**; In the above command, param1 is required while param2 is optional.

Based on Javascript

At its heart, the CLI is based on Javascript. The QT QML Javascript Engine is used to implement the CLI. This allows Javascript functionality to be used from the CLI.

Numbers

All Javascript numeric types should be supported

```
AVS4000D> 3.14          // floating point
3.14(0x00000003)
AVS4000D> 123e6     // exponents
0x754D4C0(123000000)
AVS4000D> 123e-5
0.00123(0x00000000)
AVS4000D> 0xff          // hexadecimal
0xFF(255)
```

Variables

Variables can be assigned at the CLI and do not have to be declared. Javascript variables are not type specific.

```
AVS4000D> a=123
0x7B(123)
AVS4000D> b=a
0x7B(123)
AVS4000D> b         // typing the variable name at the CLI prompt will display its cu
rrent value
0x7B(123)
AVS4000D> print(b)  // variables can be used as parameters of functions and commands
0x7B(123)
```

Operators

All Javascript operators should be accessible

```
AVS4000D> 1+1
0x02(2)
AVS4000D> 4-2
0x02(2)
AVS4000D> 3*9
0x1B(27)
AVS4000D> 9/2
4.5(0x00000005)
AVS4000D> 9%2
0x01(1)
AVS4000D> a=1
0x01(1)
AVS4000D> a++
0x01(1)
AVS4000D> a
0x02(2)
AVS4000D> a+=1
0x03(3)
AVS4000D> a*=2
0x06(6)
AVS4000D> 1<<3
0x08(8)
AVS4000D> 8>>1
0x04(4)
```

Functions

Javascript functions can be defined by the CLI or via a script file.

```
AVS4000D> function mult(a,b) { return a*b; }
AVS4000D> mult(9,4)
0x24(36)
```

Math Object

Javascript Math object is fully accessible.

```
AVS4000D> Math.PI
3.14159(0x00000003)
AVS4000D> Math.pow(2,3)
0x08(8)
AVS4000D> Math.sqrt(64)
0x08(8)
AVS4000D> Math.sqrt(2)
1.41421(0x00000001)
AVS4000D> Math.sin(90*Math.PI/180)
0x01(1)
```

Command Substitution

Command substitution provides a method for taking the output of a command and using it as a variable. The basic syntax is:

```
$(cmd)
```

Where **cmd** is any valid CLI command or method. The following is an example of using a shell *date* command to generate a date string and assigning it to a variable:

```
AVS4000D> a=$(!date)
Fri Sep 14 18:10:12 EDT 2018
AVS4000D> a
Fri Sep 14 18:10:12 EDT 2018
```

The following writes the current date to a file *t1*:

```
AVS4000D> fwrite t1 $(!date)
true
AVS4000D> fread t1
Fri Sep 14 18:10:33 EDT 2018
AVS4000D> print($(fread t1))
Fri Sep 14 18:10:33 EDT 2018
```

Shell Commands

Any CLI command that has a **!** as the first character will be treated as a shell command.

```
AVS4000D> !date
Fri Sep 14 18:14:13 EDT 2018
AVS4000D> !df -h .
Filesystem   Size   Used  Avail Capacity iused      ifree %iused  Mounted on
/dev/disk1  930Gi  611Gi  319Gi    66% 4799773 4290167506    0%   /
AVS4000D> !ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=10b<RXCSUM,TXCSUM,VLAN_HWTAGGING,AV>
        ether 3c:07:54:58:04:a5
        inet6 fe80::143f:ec5c:a4cb:e382%en0 prefixlen 64 secured scopeid 0x6
        inet 192.168.1.15 netmask 0xffffff00 broadcast 192.168.1.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect (1000baseT <full-duplex,flow-control>)
        status: active
```

Script Files

The CLI can run script files. The script files may contain CLI commands as well as Javascript. To execute a script file, just simply type the filename at the CLI prompt:

```
AVS4000D> !cat x1
VAR1=123
VAR2=456

print("Executed X1");
AVS4000D> x1
Executed X1
true
AVS4000D> VAR1
0x7B(123)
AVS4000D> VAR2
0x1C8(456)
```

Any global objects created by the script file will remain available in the CLI until restarted. The above *VAR1* and *VAR2* show they exist in the CLI after the **x1** script has completed.

Script files may also include functions:

```
AVS4000D> !cat x2
function myfunc(a) {
   var i;
   for (i=0;i<a;i++)
        print("Step "+i);
}

print("Executed X2");
AVS4000D> x2
Executed X2
true
AVS4000D> myfunc(5)
Step 0
Step 1
Step 2
Step 3
Step 4
```

Script files use CLI commands as well as javascript:

```
AVS4000D> !cat x3           // Display script file
cpld wr 4 9
r4=$(cpld rd 4)
AVS4000D> cpld wr 4 0   // write register 4 with a 0
AVS4000D> cpld rd 4 0   // read register 4
0x00(0)
AVS4000D> x3                // run x3 script file
true
AVS4000D> cpld rd 4 0   // read register 4 (shows the script file set the register)
0x09(9)
AVS4000D> r4                // The script file also set the variable r4
0x09(9)
```

Command Reference

Alias

The alias sub system can provide a way of creating shortcuts for complex commands.

```
AVS4000D> help alias
Command: alias
Group: script
Method: script.CmdAlias('%0',%A)
Description: Display or add aliases
Usage:
     alias               // Display all aliases
     alias load          // Load aliases
     alias save          // Save aliases
     alias del <name>    // Delete alias
     alias <name> <cmd>  // Create/Change alias
```

Example of defining and using an alias:

```
AVS4000D> alias df !df -h .
Alias added.
AVS4000D> df
Filesystem   Size   Used  Avail Capacity iused      ifree %iused  Mounted on
/dev/disk1  930Gi  611Gi  319Gi    66% 4802697 4290164582    0%   /
```

Operator	Substitution
%0,%1,%2...	Positional parameters
%V	All remaining positional parameters separated by commas
%S	All remaining positional parameters as strings separated by commas
%A	All remaining positional parameters combined as a single string

Aliases may use positional parameters: %0, %1, %2, %3, etc...

```
AVS4000D> alias add %0+%1
Alias added.
AVS4000D> add 3 5
0x08(8)
```

The %V substitution allows calling of methods with variable number of parameters:

```
AVS4000D> alias add script.CmdAdd(%V)
Alias added.
AVS4000D> add 1 1
0x02(2)
AVS4000D> add 1 1 1
0x03(3)
AVS4000D> add 2 2 2 2 2
0x0A(10)
```

Cmd

Dump command definitions.

```
AVS4000D> help cmd
Command: cmd
Group: script
Method: script.CmdDump('%1')
Description: Dump details of commands
Usage:
     cmd                 // Dump all available commands
     cmd [obj]           // Dump all commands from obj
Example:
     cmd sys             // Dump all commands from SYS object
```

Debug

Enable/Disable debug messages for the CLI

```
AVS4000D>> help debug
Command: debug
Group: script
Method: debug.Cmd('%0')
Description: View & Adjust debug settings

AVS4000D>> debug
Debug Settings:
  ad9364          : disabled
  config          : disabled
  fx3             : disabled
  global          : enabled
  hwlib           : enabled
  script          : disabled
  usb             : disabled

AVS4000D> debug script
    script enabled.
```

Analog Devices 9364

ADT Command Help

Help for the ADT command can be obtained by doing **adt** with no parameters:

```
AVS4000D>> adt
ADT Commands:
    adt dump           Dump Registers                  ad9364.Dump('%0')
    adt get            AD9364 View settings commands
    adt groups         List Register Groups            ad9364.Groups()
    adt info           Display info about Registers    ad9364.Info('%0')
    adt rd             Read from a register            ad9364.ReadCmd('%0')
    adt rdb            Read bits from a register       ad9364.ReadBCmd('%0',%V)
    adt set            AD9364 Change settings commands
    adt tables         List AD9364 Tables              ad9364.Tables()
    adt wr             Write to a register             ad9364.WriteCmd('%0',%1)
    adt wrb            Write bits to a register        ad9364.WriteBCmd('%0',%V)
```

More specific help can be obtained for individual commands by doing:

```
AVS4000D> help adt <cmd>
AVS4000D> help adt rd
```

ADT Command Reference

adt dump

Display/dump AD9364 registers.

```
AVS4000D> help adt dump
Command: dump
Object: adt
Method: ad9364.Dump('%0')
Description: Dump Registers
Usage:
          adt dump [group|register|addr]
Examples:
          adt dump              // Dump all registers
          adt dump 0x03         // Dump register by hex address
          adt dump 3            // Dump register by decimal address
          adt dump gen          // Dump group of registers
          adt dump gen.rxc      // Dump register by group.name
```

A list of valid register groups can be obtained using the adt groups command. A list of valid register names can be obtained using the adt info command.

Below is an example of dumping a group of registers:

```
AVS4000D> adt dump gen
                  Register Name   Addr       Value
-------------------------------------------------------------
                         gen.sc  [0x000] --> 0x00 (0)
                        gen.mcs  [0x001] --> 0x00 (0)
                        gen.txc  [0x002] --> 0x5F (95)
                        gen.rxc  [0x003] --> 0x5F (95)
                        gen.inp  [0x004] --> 0x00 (0)
                      gen.rfpll  [0x005] --> 0x00 (0)
                      gen.rxclk  [0x006] --> 0x00 (0)
                      gen.txclk  [0x007] --> 0x00 (0)
```

adt get

Commands used to view AD9364 Settings:

```
AVS4000D>> adt get
ADT GET Commands:
    adt get rxbw       Get RX RF Bandwidth             ad9364.GetRxRfBw()
    adt get rxfir      Get Rx FIR Enable               ad9364.GetRxFirEn()
    adt get rxgc       Get RX Gain Control             ad9364.GetRxGC(%0)
    adt get rxlo       Get RX LO Frequency             ad9364.GetRxLOFreq()
    adt get rxsamp     Get RX Sampling Frequency       ad9364.GetRxSamp(%0)
```

adt get rxbw

Get RX Bandwidth:

```
AVS4000D>> help adt get rxbw
Command: rxbw
Group: adt get
Method: ad9364.GetRxRfBw()
Description: Get RX RF Bandwidth
Usage:
        adt get rxbw
```

adt get rxfir

Get RX FIR Enable:

```
AVS4000D>> help adt get rxfir
Command: rxfir
Group: adt get
Method: ad9364.GetRxFirEn()
Description: Get Rx FIR Enable
Usage:
        adt get rxfir
```

adt get rxgc

Get RX Gain Control:

```
AVS4000D>> help adt get rxgc
Command: rxgc
Group: adt get
Method: ad9364.GetRxGC(%0)
Description: Get RX Gain Control
Usage:
        adt get rxgc <ch>
Example:
        adt get rxgc 0
        adt get rxgc 1
```

adt get rxlo

Get RX LO Frequency:

```
AVS4000D>> help adt get rxlo
Command: rxlo
Group: adt get
Method: ad9364.GetRxLOFreq()
Description: Get RX LO Frequency
Usage:
        adt get rxlo
```

adt get rxsamp

Get RX Sampling Frequency:

```
AVS4000D>> help adt get rxsamp
Command: rxsamp
Group: adt get
Method: ad9364.GetRxSamp(%0)
Description: Get RX Sampling Frequency
Usage:
        adt get rxsamp
```

adt groups

List the groups of registers defined for the AD9364.

```
AVS4000D> help adt groups
Command: groups
Object: adt
Method: ad9364.Groups()
Description: List Register Groups
Usage:
          adt groups
AVS4000D>> adt groups
aadc,adac,agcgt,bbpll,cc,cgt,co,control,dcxo,dio,dtest,elnag,ensm,faagc,
gc,gcgs,gen,gpo,id,mbbg,mst,ov,pdo,ppc,pwr,rdiv,ref,rssimc,rssirb,rxbbf,
rxbboff,rxfir,rxfl,rxgrb,rxlo,rxoff,rxpgc,rxqc,rxsynth,rxtia,saagc,ts,
txbbf,txbbft,txfir,txfl,txlo,txmon,txpc,txqc,txsfilt,txsynth
```

adt info

Display information about AD9364 registers.

```
AVS4000D> help adt info
Command: info
Object: adt
Method: ad9364.Info('%0')
Description: Display info about Registers
Usage:
          adt info [group|register|addr]
Examples:
          adt info              // Display info on all registers
          adt info 0x03         // Display info on hex address
          adt info 3            // Display info on decimal address
          adt info gen          // Display info on a group
          adt info gen.rxc      // Display info on a group.name
```

Below is a list of all registers defined for the AD9364

```
AVS4000D>> adt info
  Addr   Group  Register Name   Register Description
------------------------------------------------------------------------
  0x000   gen      sc           SPI Configuration
  0x001   gen      mcs          Multichip Sync & Tx Mon Control
  0x002   gen      txc          Tx Enable & Filter Control
  0x003   gen      rxc          Rx Enable & Filter Control
  0x004   gen      inp          Input Select
  0x005   gen      rfpll        RFPLL Dividers
  0x006   gen      rxclk        Rx Clock & Data Control
```

```
0x007   gen     txclk       Tx Clock & Data Control
0x009   cc      ce          Clock Enable
0x00A   cc      bbpll       BBPLL
0x00B   ts      offset      Offset
0x00C   ts      start       Start Temp Reading
0x00D   ts      sense2      Temp Sense2
0x00E   ts      temp        Temperature
0x00F   ts      config      Temp Sensor Config
0x010   ppc     config1     Parallel Port Configuration 1
0x011   ppc     config2     Parallel Port Configuration 2
0x012   ppc     config3     Parallel Port Configuration 3
0x013   ensm    mode        ENSM Mode
0x014   ensm    config1     ENSM Config 1
0x015   ensm    config2     ENSM Config 2
0x016   ensm    cal         Calibration Control
0x017   ensm    state       State
0x018   adac    ad1w        Aux DAC 1 Word
0x019   adac    ad2w        Aux DAC 2 Word
0x01A   adac    ad1c        Aux DAC 1 Config
0x01B   adac    ad2c        Aux DAC 2 Config
0x01C   aadc    clk         Aux ADC Clock Divider
0x01D   aadc    config      Aux ADC Config
0x01E   aadc    msb         Aux ADC Word MSB
0x01F   aadc    lsb         Aux ADC Word LSB
0x020   gpo     auto        Auto GPO
0x021   gpo     agcgld      AGC Gain Lock Delay
0x022   gpo     agcad       AGC Attack Delay
0x023   gpo     control     AuxDAC Enable Control
0x024   gpo     rxlsd       Rx Load Synth Delay
0x025   gpo     txlsd       Tx Load Synth Delay
0x026   gpo     elna        External LNA Control
0x027   gpo     force       GPO Force & Init
0x028   gpo     grxd0       GPO 0 Rx Delay
0x029   gpo     grxd1       GPO 1 Rx Delay
0x02A   gpo     grxd2       GPO 2 Rx Delay
0x02B   gpo     grxd3       GPO 3 Rx Delay
0x02C   gpo     gtxd0       GPO 0 Tx Delay
0x02D   gpo     gtxd1       GPO 1 Tx Delay
0x02E   gpo     gtxd2       GPO 2 Tx Delay
0x02F   gpo     gtxd3       GPO 3 Tx Delay
0x030   gpo     arxd1       AuxDAC 1 Rx Delay
0x031   gpo     atxd1       AuxDAC 1 Tx Delay
0x032   gpo     arxd2       AuxDAC 2 Rx Delay
0x033   gpo     atxd2       AuxDAC 2 Rx Delay
0x035   co      ptr         Control Output Pointer
```

```
0x036   co       enable       Control Output Enable
0x037   id       product      Product ID
0x03A   ref      clock        Reference Clock Cycles
0x03B   dio      control      Digital IO Control
0x03C   dio      bias         LVDS Bias Control
0x03D   dio      ic1          Digital IO Control 1
0x03E   dio      ic2          Digital IO Control 2
0x03F   bbpll    control1     BBPL Control 1
0x041   bbpll    ffreq1       Fractional BB Freq Word 1
0x042   bbpll    ffreq2       Fractional BB Freq Word 2
0x043   bbpll    ffreq3       Fractional BB Freq Word 3
0x044   bbpll    ifreq        Integer BB Freq Word
0x045   bbpll    rcscale      Ref Clock Scaler
0x046   bbpll    cpc          CP Current
0x047   bbpll    mcs          MCS Scale
0x048   bbpll    lf1          Loop Filter 1
0x049   bbpll    lf2          Loop Filter 2
0x04A   bbpll    lf3          Loop Filter 3
0x04B   bbpll    vco          VCO Control
0x04C   bbpll    mb86         Must be set to 0x86
0x04D   bbpll    control2     BBPL Control 2
0x04E   bbpll    control3     BBPL Control 3
0x050   pdo      rxs          Rx Synth Power Down Override
0x051   pdo      txs          Tx Synth Power Down Override
0x052   pdo      control0     Rx Synth Power Down Override
0x054   pdo      rxadc        Rx ADC Power Down Override
0x056   pdo      txadc        Tx ADC Power Down Override
0x057   pdo      analog       Analog Power Down Override
0x058   pdo      misc         Misc Power Down Override
0x05E   ov       overflow     CH Overflow
0x060   txfir    addr         Tx Filter Coef Address
0x061   txfir    wrdata1      Tx Filter Coef Write Data 1
0x062   txfir    wrdata2      Tx Filter Coef Write Data 2
0x063   txfir    rddata1      Tx Filter Coef Read Data 1
0x064   txfir    rddata2      Tx Filter Coef Read Data 2
0x065   txfir    config       Tx Filter Configuration
0x067   txmon    txlg         Tx Mon Low Gain
0x068   txmon    txhg         Tx Mon High Gain
0x069   txmon    txdc         Tx Mon Delay Counter
0x06A   txmon    txlt         Tx Mon Level Threshold
0x06B   txmon    txrssi1      Tx RSSI 1
0x06C   txmon    txrssi2      Tx RSSI 2
0x06D   txmon    txrssilsb    Tx RSSI LSB
0x06E   txmon    tpm          TPM Mode Enable
0x06F   txmon    tgc          Temp Gain Coefficient
```

```
0x070   txmon    config       Tx Mon Config
0x073   txpc     atten0       Tx Atten 0
0x074   txpc     atten1       Tx Atten 1
0x077   txpc     aoffset      Tx Atten Offset
0x078   txpc     athresh      Tx Atten Threshold
0x07C   txpc     update       Immediate Update
0x08E   txqc     phase1       Tx Out 1 Phase Corr
0x08F   txqc     gain1        Tx Out 1 Gain Corr
0x092   txqc     ioffset1     Tx Out 1 Offset I
0x093   txqc     qoffset1     Tx Out 1 Offset Q
0x096   txqc     phase2       Tx Out 2 Phase Corr
0x097   txqc     gain2        Tx Out 2 Gain Corr
0x09A   txqc     ioffset2     Tx Out 2 Offset I
0x09B   txqc     qoffset2     Tx Out 2 Offset Q
0x09F   txqc     force        Force Bits
0x0A0   txqc     rxnco        Quad Cal NCO Freq & Phase Offset
0x0A1   txqc     control      Quad Cal Control
0x0A2   txqc     st7f         Set to 0x7F
0x0A3   txqc     txnco        Tx NCO Frequency
0x0A4   txqc     stf0         Set to 0xF0
0x0A5   txqc     mft          Mag Ftest Thresh
0x0A7   txqc     status       Tx Quad Cal Status
0x0A9   txqc     stff         Set to 0xFF
0x0AA   txqc     lmtg         Tx Quad Full/LMT Gain
0x0AE   txqc     lmfg         Tx Quad LMT Gain
0x0C2   txbbf    r1           Tx BBF R1
0x0C3   txbbf    r2           Tx BBF R2
0x0C4   txbbf    r3           Tx BBF R3
0x0C5   txbbf    r4           Tx BBF R4
0x0C6   txbbf    rp           Tx BBF RP
0x0C7   txbbf    c1           Tx BBF C1
0x0C8   txbbf    c2           Tx BBF C2
0x0C9   txbbf    cp           Tx BBF CP
0x0CA   txbbf    pd           Tuner PD
0x0CB   txbbf    r2b          Tx BBF R2b
0x0D0   txsfilt  config       Config0
0x0D1   txsfilt  resist       Resistor
0x0D2   txsfilt  cap          Capacitor
0x0D3   txsfilt  mb60         Must be 0x60
0x0D6   txbbft   divide       Tx BBF Tune Divider
0x0D7   txbbft   mode         Tx BBF Tune Mode
0x0F0   rxfir    addr         Rx Filter Coef Address
0x0F1   rxfir    wrdata1      Rx Filter Coef Write Data 1
0x0F2   rxfir    wrdata2      Rx Filter Coef Write Data 2
0x0F3   rxfir    rddata1      Rx Filter Coef Read Data 1
```

```
0x0F4   rxfir    rddata2      Rx Filter Coef Read Data 2
0x0F5   rxfir    config       Rx Filter Configuration
0x0F6   rxfir    gain         Rx Filter Gain
0x0FA   gcgs     config1      AGC Config1
0x0FB   gcgs     config2      AGC Config2
0x0FC   gcgs     config3      AGC Config3
0x0FD   gcgs     maxlmt       Max LMT/Full Gain
0x0FE   gcgs     pwt          Peak Wait Time
0x100   gcgs     dg           Digital Gain
0x101   gcgs     agcll        AGC Lock Level
0x103   gcgs     gsconfig1    Gain Step Config 1
0x104   gcgs     adcsot       ADC Small Overload Threshold
0x105   gcgs     adclot       ADC Large Overload Threshold
0x106   gcgs     gsconfig2    Gain Step Config 2
0x107   gcgs     lmtsot       LMT Small Overload Threshold
0x108   gcgs     lmtlot       LMT Large Overload Threshold
0x109   gcgs     manlmt       Manual LMT/Full Gain
0x10A   gcgs     mlpfg        Manual LPF Gain
0x10B   gcgs     mdfg         Manual Digital/Forced Gain
0x110   faagc    config1      Fast Attack AGC Config 1
0x111   faagc    config2      Fast Attack AGC Config 2
0x112   faagc    elt          Energy Lost Threshold
0x113   faagc    sst          Stronger Signal Threshold
0x114   faagc    lpt          Low Power Threshold
0x115   faagc    ssf          Stronger Signal Freeze
0x116   faagc    foog         Final Overrange and Opt Gain
0x117   faagc    edc          Energy Detect Count
0x118   faagc    agcllul      AGCLL Upper Limit
0x119   faagc    glec         Gain Lock Exit Count
0x11A   faagc    ilmtgl       Initial LMT Gain Limit
0x11B   faagc    itime        Increment Time
0x120   saagc    agcilt       AGC Inner Low Threshold
0x121   saagc    lmtoc        LMT Overload Counters
0x122   saagc    agcoc        AGC Overload Counters
0x123   saagc    gstep1       Gain Step 1
0x124   saagc    guc1         Gain Update Counter 1
0x125   saagc    guc2         Gain Update Counter 2
0x128   saagc    dsc          Digital Sat Counter
0x129   saagc    opt          Outer Power Thresholds
0x12A   saagc    gstep2       Gain Step 2
0x12C   elnag    high         Ext LNA High Gain
0x12D   elnag    low          Ext LNA Low Gain
0x130   agcgt    addr         Gain Table Address
0x131   agcgt    wrdata1      Gain Table Write Data 1
0x132   agcgt    wrdata2      Gain Table Write Data 2
```

```
0x133   agcgt    wrdata3      Gain Table Write Data 3
0x134   agcgt    rddata1      Gain Table Read Data 1
0x135   agcgt    rddata2      Gain Table Read Data 2
0x136   agcgt    rddata3      Gain Table Read Data 3
0x137   agcgt    config       Gain Table Configuration
0x138   mst      addr         Mixer Subtable Address
0x139   mst      wrgain       Mixer Subtable Gain Word Write
0x13A   mst      wrbias       Mixer Subtable Bias Word Write
0x13B   mst      wrcontrol    Mixer Subtable Control Word Write
0x13C   mst      rdgain       Mixer Subtable Gain Word Write
0x13D   mst      rdbias       Mixer Subtable Bias Word Write
0x13E   mst      rdcontrol    Mixer Subtable Control Word Write
0x13F   mst      config       Mixer Subtable Config
0x140   cgt      addr         Word Address
0x141   cgt      wrdiff       Gain Diff Word/Error Write
0x142   cgt      rderror      Gain Error Read
0x143   cgt      config       Calibration Gain Table Config
0x144   cgt      rdlna        LNA Gain Diff Read Back
0x145   gc       mmcgi        Max Mixer Calibration Gain Index
0x146   gc       tgc          Temp Gain Coefficient
0x147   gc       settle       Settle Time
0x148   gc       duration     Measure Duration
0x149   gc       caltemp      Cal Temp Sensor Word
0x150   rssimc   dur01        Duration 0,1
0x151   rssimc   dur23        Duration 2,3
0x152   rssimc   weight0      Weight 0
0x153   rssimc   weight1      Weight 1
0x154   rssimc   weight2      Weight 2
0x155   rssimc   weight3      Weight 3
0x156   rssimc   delay        RSSI Delay
0x157   rssimc   wait         RSSI Wait Time
0x158   rssimc   config       RSSI Config
0x15C   rssimc   dpd          Dec Power Duration
0x15D   rssimc   lnagain      LNA Gain
0x161   pwr      rxfilter     CH1 Rx Filter Power
0x169   rxqc     config1      Calibration Config 1
0x16A   rxqc     mb75         Must be 0x75
0x16B   rxqc     mb95         Must be 0x95
0x170   rxpgc    rxaphase     RxA Phase Corr
0x171   rxpgc    rxagain      RxA Gain Corr
0x174   rxpgc    rxaoff1      RxA Q Offset
0x175   rxpgc    rxaoff2      RxA I/Q Offset
0x176   rxpgc    rxaoff3      RxA I Offset
0x179   rxpgc    rxbphase     RxB Phase Corr
0x17A   rxpgc    rxbgain      RxB Gain Corr
```

0x17D	rxpgc	rxboff1	RxB Q Offset
0x17E	rxpgc	rxboff2	RxB I/Q Offset
0x17F	rxpgc	rxboff3	RxB I Offset
0x182	rxpgc	force	Force Bits
0x185	rxoff	wait	Wait Count
0x186	rxoff	count	RF DC Offset Count
0x187	rxoff	config1	RF DC Offset Config1
0x188	rxoff	atten	RF DC Offset Attenuation
0x189	rxoff	mb30	Must be 0x30
0x18B	rxoff	config2	RF DC Offset Config2
0x18C	rxoff	calgain	RF Cal Gain Index
0x18D	rxoff	soithresh	SOI Threshold
0x190	rxoff	bbshift	BB DC Offset Shift
0x191	rxoff	bbfsshift	BB DC Offset Fast Settle Shift
0x192	rxoff	bbfsdur	BB Fast Settle Duration
0x193	rxoff	bbcount	BB DC Offset (Must be 0x3F)
0x194	rxoff	bbatten	BB DC Offset Attenuation
0x19A	rxbboff	cimsb	BB DC Correction Word I MSB
0x19B	rxbboff	cilsb	BB DC Correction Word I LSB
0x19C	rxbboff	cqmsb	BB DC Correction Word Q MSB
0x19D	rxbboff	cqlsb	BB DC Correction Word Q LSB
0x1A2	rxbboff	tcimsb	BB DC Tracking Correction Word I MSB
0x1A3	rxbboff	tcilsb	BB DC Tracking Correction Word I LSB
0x1A4	rxbboff	tcqmsb	BB DC Tracking Correction Word Q MSB
0x1A5	rxbboff	tcqlsb	BB DC Tracking Correction Word Q LSB
0x1A7	rssirb	symmsb	RSSI Symbols MSB
0x1A8	rssirb	premsb	RSSI Preamble MSB
0x1AB	rssirb	symlsb	RSSI Symbols LSB
0x1AC	rssirb	prelsb	RSSI Preamble LSB
0x1DB	rxtia	config	Rx TIA Config
0x1DC	rxtia	lsb	TIA C LSB
0x1DD	rxtia	msb	TIA C MSB
0x1E0	rxbbf	r1a	BBF R1A
0x1E2	rxbbf	tunectl	Tune Control
0x1E3	rxbbf	tunectl2	Tune Control 2
0x1E6	rxbbf	r2346	Rx BBF R2346
0x1E7	rxbbf	c1msb	Rx BBF C1 MSB
0x1E8	rxbbf	c1lsb	Rx BBF C1 LSB
0x1E9	rxbbf	c2msb	Rx BBF C2 MSB
0x1EA	rxbbf	c2lsb	Rx BBF C2 LSB
0x1EB	rxbbf	c3msb	Rx BBF C3 MSB
0x1EC	rxbbf	c3lsb	Rx BBF C3 LSB
0x1ED	rxbbf	cc1	Rx BBF CC1 Ctr
0x1EE	rxbbf	mb60	Must be 0x60
0x1EF	rxbbf	cc2	Rx BBF CC2 Ctr

```
0x1F0   rxbbf    pow          Rx BBF Pow Rz Byte1
0x1F1   rxbbf    cc3          Rx BBF CC3 Ctr
0x1F2   rxbbf    r5tune       Rx BBF R5 Tune
0x1F3   rxbbf    tune         Rx BBF Tune
0x1F4   rxbbf    mangain      Rx BBF Man Gain
0x1F8   rxbbf    tunediv      Rx BBF Tune Divide
0x1F9   rxbbf    tunecfg      Rx BBF Tune Config
0x1FA   rxbbf    mb01         Must be 0x01
0x1FB   rxbbf    bbbwmhz      Rx BBBW MHz
0x1FC   rxbbf    bbbwkhz      Rx BBBW kHz
0x230   rxsynth  dvcocal      Disable VCO Cal
0x231   rxsynth  intb0        Integer Byte 0
0x232   rxsynth  intb1        Integer Byte 1
0x233   rxsynth  fracb0       Fractional Byte 0
0x234   rxsynth  fracb1       Fractional Byte 1
0x235   rxsynth  fracb2       Fractional Byte 2
0x236   rxsynth  falc         Force ALC
0x237   rxsynth  fvcot0       Force VCO Tune 0
0x238   rxsynth  fvcot1       Force VCO Tune 1
0x239   rxsynth  alcvar       ALC/Varactor
0x23A   rxsynth  vcoout       VCO Output
0x23B   rxsynth  cpcur        CP Current
0x23C   rxsynth  cpoff        CP Offset
0x23D   rxsynth  cpcfg        CP Config
0x23E   rxsynth  loopf1       Loop Filter 1
0x23F   rxsynth  loopf2       Loop Filter 2
0x240   rxsynth  loopf3       Loop Filter 3
0x241   rxsynth  dither       Dither/CP Cal
0x242   rxsynth  vcobias      VCO Bias 1
0x244   rxsynth  cstatus      Cal Status
0x246   rxsynth  st02         Set To 0x02
0x247   rxsynth  cpovrg       CP Ovrg/VCO Lock
0x248   rxsynth  st0b         Set To 0x0B
0x249   rxsynth  vcocal       VCO Cal
0x24A   rxsynth  ldconfig     Lock Detect Config
0x250   rxsynth  st70         Set To 0x70
0x251   rxsynth  vcovc1       VCO Varactor Control 1
0x25A   rxfl     setup        Rx Fast Lock Setup
0x25B   rxfl     sidelay      Rx Fast Lock Setup Init Delay
0x25C   rxfl     paddr        Rx Fast Lock Program Address
0x25D   rxfl     pdata        Rx Fast Lock Program Data
0x25E   rxfl     pread        Rx Fast Lock Program Read
0x25F   rxfl     pcontrol     Rx Fast Lock Program Control
0x261   rxlo     gpmode       Rx LO Generation Power Mode
0x270   txsynth  dvcocal      Disable VCO Cal
```

```
0x271   txsynth  intb0        Integer Byte 0
0x272   txsynth  intb1        Integer Byte 1
0x273   txsynth  fracb0       Fractional Byte 0
0x274   txsynth  fracb1       Fractional Byte 1
0x275   txsynth  fracb2       Fractional Byte 2
0x276   txsynth  falc         Force ALC
0x277   txsynth  fvcot0       Force VCO Tune 0
0x278   txsynth  fvcot1       Force VCO Tune 1
0x279   txsynth  alcvar       ALC/Varactor
0x27A   txsynth  vcoout       VCO Output
0x27B   txsynth  cpcur        CP Current
0x27C   txsynth  cpoff        CP Offset
0x27D   txsynth  cpcfg        CP Config
0x27E   txsynth  loopf1       Loop Filter 1
0x27F   txsynth  loopf2       Loop Filter 2
0x280   txsynth  loopf3       Loop Filter 3
0x281   txsynth  dither       Dither/CP Cal
0x282   txsynth  vcobias      VCO Bias 1
0x284   txsynth  cstatus      Cal Status
0x286   txsynth  st02         Set To 0x02
0x287   txsynth  cpovrg       CP Ovrg/VCO Lock
0x288   txsynth  st0b         Set To 0x0B
0x289   txsynth  vcocal       VCO Cal
0x28A   txsynth  ldconfig     Lock Detect Config
0x290   txsynth  st70         Set To 0x70
0x291   txsynth  vcovc1       VCO Varactor Control 1
0x292   dcxo     coarse       DXCO Coarse Tune
0x293   dcxo     fine2        DXCO Fine Tune 2
0x294   dcxo     fine1        DXCO Fine Tune 1
0x29A   txfl     setup        Tx Fast Lock Setup
0x29B   txfl     sidelay      Tx Fast Lock Setup Init Delay
0x29C   txfl     paddr        Tx Fast Lock Program Address
0x29D   txfl     pdata        Tx Fast Lock Program Data
0x29E   txfl     pread        Tx Fast Lock Program Read
0x29F   txfl     pcontrol     Tx Fast Lock Program Control
0x2A1   txlo     gpmode       Rx LO Generation Power Mode
0x2A6   mbbg     st0e1        Set to 0x0E
0x2A8   mbbg     st0e2        Set to 0x0E
0x2AB   rdiv     config1      Ref Divide Config 1
0x2AC   rdiv     config2      Ref Divide Config 2
0x2B0   rxgrb    gain         Gain
0x2B1   rxgrb    lpfgain      LPF Gain
0x2B2   rxgrb    diggain      Digital Gain
0x2B3   rxgrb    fast         Fast Attack State
0x2B4   rxgrb    sloop        Slow Loop State
```

```
  0x2B8   rxgrb    ovrg         Ovrg Sigs
  0x3DF   control  control      Control Register
  0x3F4   dtest    config       BIST Config
  0x3F5   dtest    config2      BIST Config 2
  0x3F6   dtest    data         BIST and Data Port Test Config
```

adt rd

Read an AD9364 register.

```
AVS4000D> help adt rd
Command: rd
Object: adt
Method: ad9364.ReadCmd('%0')
Description: Read from a register
Usage:
          adt rd <register|addr>
Examples:
          adt rd 0x03       // Address as hex address
          adt rd 3          // Address as decimal address
          adt rd gen.rxc    // Address as group.name

AVS4000D> adt rd 3
0x5F(95)
```

adt rdb

Read bits from an AD9364 register.

```
AVS4000D> help adt rdb
Command: rdb
Object: adt
Method: ad9364.ReadBitsCmd('%0',%V)
Description: Read bits from a register
Usage:
          adt rdb <register|addr> <bit> [num]
Examples:
          adt rdb 0x03 6         // Read bit 6 from register 0x03
          adt rdb 3 0 2          // Read bits 0-1 from register 0x03
          adt rdb gen.rxc 0 2    // Read bits 0-1 from register gen.rxc
```

If the last parameter (num) is omitte it will default to 1.

```
AVS4000D> adt rdb gen.rxc 0 2
0x03(3)
AVS4000D> adt rdb gen.rxc 6
0x01(1)
AVS4000D> adt rd gen.rxc
0x5F(95)
```

adt set

Commands to change AD9364 settings:

```
AVS4000D>> adt set
ADT SET Commands:
    adt set rxbw       Set RX RF Bandwidth             ad9364.SetRxRfBw(%0)
    adt set rxfir      Set Rx FIR Enable               ad9364.SetRxFirEn(%0)
    adt set rxgc       Set RX Gain Control             ad9364.SetRxGC(%0,%1)
    adt set rxlo       Set RX LO Frequency             ad9364.SetRxLOFreq(%0)
    adt set rxsamp     Set RX Sampling Frequency       ad9364.SetRxSamp(%0)
```

adt set rxbw

Set RX Bandwidth:

```
AVS4000D>> help adt set rxbw
Command: rxbw
Group: adt set
Method: ad9364.SetRxRfBw(%0)
Description: Set RX RF Bandwidth
Usage:
        adt set rxbw <bw>
Example:
        adt set rxbw 1e6
```

adt set rxfir

Set RX FIR Enable:

```
AVS4000D>> help adt set rxfir
Command: rxfir
Group: adt set
Method: ad9364.SetRxFirEn(%0)
Description: Set Rx FIR Enable
Usage:
        adt set rxfir <enable>
Example:
        adt set rxfir true      # Enable
        adt set rxfir false     # Disable
```

adt set rxgc

Set RX Gain Control:

```
AVS4000D>> help adt set rxgc
Command: rxgc
Group: adt set
Method: ad9364.SetRxGC(%0,%1)
Description: Set RX Gain Control
Usage:
        adt set rxgc <ch> <val>
Example:
        adt set rxgc 0 0x00
```

adt set rxlo

Set RX LO Frequency:

```
AVS4000D>> help adt set rxlo
Command: rxlo
Group: adt set
Method: ad9364.SetRxLOFreq(%0)
Description: Set RX LO Frequency
Usage:
        adt set rxlo <freq>
Example:
        adt set rxlo 2.4e9
```

adt set rxsamp

Set RX Sampling Frequency:

```
AVS4000D>> help adt set rxsamp
Command: rxsamp
Group: adt set
Method: ad9364.SetRxSamp(%0)
Description: Set RX Sampling Frequency
Usage:
        adt set rxsamp <freq>
Example:
        adt set rxsamp 50e6
```

adt wr

Write to an AD9364 register.

```
AVS4000D> help adt wr
Command: wr
Object: adt
Method: ad9364.WriteCmd('%0',%1)
Description: Write to a register
Usage:
          adt wr <register|addr> <value>
Examples:
          adt wr 0x04 9         // Write to a hex address
          adt wr 4 0x0A         // Write to a decimal address
          adt wr gen.inp 11 // Write to a group.address
```

adt wrb

Write to selected bits in an AD9364 register. This operation will actually perform a read-modify-write of the register.

```
AVS4000D> help adt wrb
Command: wrb
Object: adt
Method: ad9364.WriteBitsCmd('%0',%V)
Description: Write to a register
Usage:
          adt wrb <register|addr> <value> <bits> [num]
Examples:
          adt wrb 0x03 1 6      // Write a value of 1 to bit 6 of addr 0x03
          adt wrb 0x03 2 0 2    // Write a value of 2 to bits 0-1 of addr 0x03
          adt wrb gen.rxc 3 4 2 // Write value of 3 to bits 4-5 of gen.rxc
```

If the last parameter (num) is omitte it will default to 1.