NAME

`configuration` – administer hardware configuration file

DESCRIPTION

Each spectrometer has an associated file named *config* that describes the hardware interfaces and devices being used. There is also a binary-format file named *settings* that holds motor positions, user-unit offsets and software limits. The program *edconf* (invoked by the spec macro `config`) is used to maintain these files.

This note describes the ASCII format of the *config* file. Although the *config* file can be edited by hand, you will be safer using the *edconf* program to make modifications as *edconf* insures the *config* file obeys the structuring rules required by spec.

Without arguments, *edconf* will use the *config* and *settings* files in the current directory. If given a directory name as an argument, it will use the files in that directory. If invoked with the **–s** flag, *edconf* will run in "simulate" mode, allowing the user to view but not modify the files. If the user doesn't have write permission for the *config* file, *edconf* will automatically run in "simulate" mode.

Comment lines begin with a # in the *config* file. Other lines contain key words, such as `CDEV` or `MOT00`, followed by an equals sign and one or more parameters.

DEVICE KEY WORDS

The following table summarizes the key words that select particular hardware devices.

```
Keyword      Parameters
===========  ====================================
CDEV         device_name
SDEV_#       device_name  baud_rate  opt_modes
PC_PORT_#    base_address #_of_ports rw_mode
SW_SFTWARE   1

PC_AM9513    base_address
PC_DAC_B12   base_address #_of_motors
PC_DAC_B16   base_address #_of_motors
PC_DAC_T12   base_address #_of_motors
PC_DAC_T16   base_address #_of_motors
PC_DSP6001   base_address
PC_GPIB11    device_name
PC_GPIBPC    device_name
PC_GPIBPC_L  device_name
PC_GPIBPC2   device_name
PC_GPIBPC2_L device_name
PC_GPIBPC3   device_name
PC_GPIBPC3_L device_name
PC_GPIBPC4   device_name
PC_GPIBPC4_L device_name
PC_IOTECH    device_name
PC_MIZAR     vme_address  #_of_counters IRQ#_or_POLL
PC_KS2926    base_address
PC_MM2000    base_address #_of_motors
PC_NIVME     /dev/null
PC_OMS       device_name  #_of_motors INTR_or_POLL
PC_OMSP      base_address #_of_motors
PC_OMSV      vme_address  #_of_motors IRQ#_or_POLL
PC_PCA3      base_address
PC_PCAII     device_name  base_address INTR_or_POLL
PC_SICL_H    device_name
PC_SICL_HP   device_name
```

```
PC_TEC488    base_address
PC_TEC488_L  base_address

RS_18011     device_name  baud_rate #_of_motors
RS_18092     device_name  baud_rate #_of_motors
RS_CATO      device_name  baud_rate
RS_CM3000    device_name  baud_rate #_of_motors
RS_CM4000    device_name  baud_rate #_of_motors
RS_CMSX      device_name  baud_rate #_of_motors
RS_INEL      device_name  baud_rate #_of_counters
RS_IP28      device_name  baud_rate #_of_motors
RS_ITL09     device_name  baud_rate #_of_motors
RS_MC4       device_name  baud_rate #_of_motors
RS_MCB       device_name  baud_rate #_of_motors
RS_MCU       device_name  baud_rate #_of_motors
RS_MCU_E     device_name  baud_rate #_of_motors
RS_MM2000    device_name  baud_rate #_of_motors
RS_NSK       device_name  baud_rate #_of_motors
RS_OR9XB     device_name  baud_rate #_of_counters
RS_OR9XC     device_name  baud_rate #_of_counters
RS_OR9XT     device_name  baud_rate #_of_counters
RS_SIX19     device_name  base_rate #_of_motors
RS_TC100     device_name  base_rate #_of_channels
RS_XRGCI_M   device_name  baud_rate #_of_motors
RS_XRGCI_T   device_name  baud_rate #_of_counters

GP_CC488     gpib_address INTR_or_POLL
GP_CM3000    gpib_address #_of_motors
GP_CM4000    gpib_address #_of_motors
GP_HUB9000   gpib_address #_of_motors
GP_IFE2D     gpib_address
GP_IP28      gpib_address #_of_motors
GP_ITL09     gpib_address #_of_motors
GP_K2001     gpib_address
GP_KS3988    gpib_address INTR_or_POLL
GP_MC4       gpib_address #_of_motors
GP_MCB       gpib_address #_of_motors
GP_MM2000    gpib_address #_of_motors
GP_MMC32     gpib_address #_of_motors
GP_OR918A    gpib_address
GP_OR9XB     gpib_address #_of_counters
GP_OR9XT     gpib_address #_of_counters
GP_OR9XC     gpib_address #_of_counters
GP_PCA_M     gpib_address
GP_PI        gpib_address #_of_motors
GP_ST116     gpib_address
GP_STAR1     gpib_address
```

CDEV specifies the CAMAC device accessed through a spec CAMAC driver. For example, `CDEV = /dev/ca00`.

SDEV_0 specifies the name and baud rate of the serial device with spec device-address 0 to be used with the `ser_get()` and `ser_put()` functions. The *opt_modes* are optional arguments that set the line modes. For example, `SDEV_0 = /dev/com2 9600 raw`. Choices for *opt_modes* are `raw`, `cooked`, `evenp`, `oddp`, `noflow` and `igncr`. Several of these can be combined. See the *serial* help file for more information. The default mode is `cooked`. SDEV_1, SDEV_2, ... specify additional serial devices.

PC_PORT_0 identifies a range of PC IO port addresses for use with the port_get() and port_put() functions. The board's hexadecimal base address is the first argument. The number of contiguous ports (maximum of 16) that can be accessed is next. If *rw_mode* is 0, the ports are read only, if 1, the ports are both readable and writable. For example, PC_PORT_0 = 0x300 3 1.

SW_SFTWARE selects the software timer. The parameter is unused.

PC BOARD DEVICES

Key words with the PC_ prefix select devices that generally are on adapter boards that fit into a PC computer E/ISA slot.

PC_AM9513 selects either of two boards that use the Am9513 counter chip. Both the Metrabyte Model CTM-05 interface board and the Scientific Solutions (formerly Tecmar) Labmaster board are supported. The hexadecimal base address of the counter-chip registers on the board must be given as a parameter, as in PC_AM9513 = 0x348.

PC_GPIB11 selects the National Instruments GPIB11V board for GPIB control on a MicroVax computer. Only a *device_name* parameter is needed.

PC_GPIBPC selects the National Instruments PCII board for GPIB control. Only a *device_name* parameter is needed.

PC_PCAII selects the Tennelec/Nucleus PCA II MCA board. The board's base address is given as an argument.

PC_OMS gives information about an Oregon Micro Systems PCX/PC38 board for motor control. The number of motors and a key word, either INTR or POLL must also be given, as in PC_OMS = /dev/oms 4 INTR.

PC_TEC488 selects the TECMAR (or Scientific Solutions) IEEE-488 board for GPIB control. The board's hexadecimal base address must be given as a parameter.

RS-232C DEVICES

Key words that begin with the RS_ prefix describe devices that reside on a serial (RS-232C) interface. All serial devices require device name and baud rate parameters.

RS_CATO selects the Silena MCA.

RS_CM3000 selects the Compumotor 3000 motor controller. A number-of-motors parameter is required.

RS_CM4000 selects the Compumotor 4000 motor controller. A number-of-motors parameter is required.

RS_INEL selects the Inel 715 dual counter. A number-of-counters parameter is expected.

RS_IP28 chooses the Microcontrole IP28 stepper motor controller. This device also requires a number-of-motors parameter, as in RS_IP28 = /dev/com2 9600 4.

RS_MCB chooses the Advanced Control Systems MCB stepper motor controller. This device also requires a number-of-motors parameter.

RS_MCU chooses the Advanced Control Systems MCU stepper motor controller. This device also requires a number-of-motors parameter.

RS_MCU_E is the same as above, except that the motor controllers use encoders. The only difference from the standard behavior is that during the synchronization of software motors positions with the hardware registers, if the discrepancy is less than some fixed number of steps, the hardware register are automatically assumed to contain the correct position. The number of steps is taken from field eight of the motor parameter configuration (see below).

RS_MC4 chooses the Klinger MC-4 stepper motor controller. This device also requires a number-of-motors parameter.

RS_OR9XT or RS_OR9XC select Ortec 900 series counters or counter/timers. A number-of-counters parameter is required.

RS_SIX19 selects the Microcontrole SIX19 motor controller.

TC100 selects the Nicomp TC-100 autocorrelator. The number of channels is given as a parameter.

RS_XRGCI_M selects the Inel XRGCI motor controller/timer instrument and specifies the number of motors that are used with it.

RS_XRGCI_T selects the Inel XRGCI motor controller/timer instrument and specifies the number of counters that are used with it.

GPIB DEVICES

Key words that begin with the GP_ prefix describe devices that use the GPIB interface. All such devices require a GPIB address parameter.

GP_CM3000 selects the Compumotor 3000 motor controller.

GP_CM4000 selects the Compumotor 4000 motor controller.

GP_IP28 chooses the Microcontrole IP28 stepper motor controller on a GPIB interface This device requires a GPIB address and a number-of-motors parameter, as in `GP_IP28 = 12 4`.

GP_KS3988 selects the Kinetic Systems Model 3988-G2A GPIB CAMAC crate controller. The National Instruments GPIB controller (PC_GPIBPC) must also be selected. The GPIB address is specified as an argument, along with either of the key words, INTR or POLL. (Interrupt-driven mode is recommended, but switch to polled mode if there are problems and contact CSS.)

GP_MCB chooses the Advanced Control Systems MCB stepper motor controller. This device also requires a number-of-motors parameter.

GP_MC4 chooses the Klinger MC-4 stepper motor controller. This device also requires a number-of-motors parameter.

GP_MMC32 chooses the NSLS-made stepper motor controller. This device also requires a number-of-motors parameter.

GP_OR9XT or GP_OR9XC select Ortec 900 series counters or counter/timers. A number-of-counters is required.

GP_OR974T or GP_OR974C select the Ortec 974 device to be used as a counter/timer or just a counter. A number-of-counters is required.

CAMAC MODULES

CAMAC slot assignments consist of a module code on the left and a slot number on the right. For example,

```
CA_KS3610 = 2
```

tells the program a Kinetic Systems 3610 hex scaler is in slot 2.

The following modules names are recognized by spec (KS is Kinetic Systems, DSP is DSP Technology):

CA_DSP2190	DSP 2190 MCS Averager
CA_E250*	DSP E250A 12-bit D/A (as motor controller)
CA_E500*	DSP E500A Stepper Motor Controller
CA_IO*	Any module using F codes of 0 or 16
CA_IOM1	BiRa 2601 I/O For E500 Multiplexing

CA_IOM2	F16,A0 I/O For E500 Multiplexing
CA_IOM3	F16,A1 I/O For E500 Multiplexing
CA_KS3112*	KS 3112 D/A (as motor controller)
CA_KS3116*	KS 3116 D/A (as motor controller)
CA_KS3195*	KS 3195 D/A (as motor controller)
CA_KS3388	KS 3388 GPIB interface
CA_KS3512*	KS 3512 ADC (as counters)
CA_KS3610*	KS 3610 6-Channel, 50 MHz Counter
CA_KS3640C*	KS 3640 Up/Down Counter as Counter
CA_KS3640M*	KS 3640 Up/Down Counter (for SMC's)
CA_KS3640T	KS 3640 Up/Down Counter as Timer
CA_KS3655	KS 3655 8-Channel Timing Generator
CA_KS3929	KS 3929 SCSI Crate Controller on Sun
CA_KS3929_HP	KS 3929 SCSI Crate Controller on HP
CA_KSC	KS Crate Controller with KS Software
CA_LC2301	LeCroy 2301 interface for QVT MCA
CA_LC3512	LeCroy 3512 Spectroscopy ADC
CA_LC3521	LeCroy 3521A Multichannel Scaling
CA_LC3588	LeCroy 3588 Multi Channel Scaler
CA_LC8206	LeCroy MM8206A Histogramming Memory
CA_QS450	DSP QS-450 4-Channel Counter
CA_RTC018	DSP RTC-018 Real Time Clock
CA_SMC*	Joerger Motor Controller SMC-L or SMC-24
CA_TS201	DSP TS-201 Dual Timer/Scaler

* More than one of these modules are allowed. Append _# to number modules consecutively, where # is 0, 1, 2, etc.

MOTOR PARAMETERS

Motor parameter assignment consists of key words of the form MOT00, MOT01, ... followed by 11 values. The MOT key words must be numbered consecutively starting at zero. The values are

1 Controller type (E500, SMC, OMS, ...)
2 Steps per unit (sign changes direction of motion)
3 Sign between user and dial units (+1 or −1)
4 Steady state rate (Hz) (must be positive)
5 Base rate (Hz) (must be positive) (also is backlash rate)
6 Steps for backlash (sign changes direction of motion)
7 Acceleration time (msec)
8 Not used
9 Motor flags (protection, units, etc.)
10 Motor mnemonic (th, phi, sl1, ...)
11 Motor name (Theta, Phi, Slit 1, ...)

An example is

```
#Fields   1     2 3    4   5  6   7 8 9 10 11
MOT00 = OMS −2000 1 2000 200 50 125 0 3 th Theta
```

Valid controller types are currently:

18011	18092	CM3000	CM4000	CMSX	CMSX_E	DAC_B12
DAC_B16	DAC_T12	DAC_T16	E250	E500	E500_M	EP_OMS
ES_OMS	ES_PIE	ES_VPAP	HUB9000	IP28	ITL09	ITL09_E
KS3112	KS3116	KS3195	MAXE	MAXE_E	MAXE_S	MC4
MCB	MCU	MCU_E	MM2000	MMC32	NONE	NSK
OMS	OMS_E	PI	SIX19	SMC	XRGCI_M	

Field 2, the steps per unit, may be non-integral, and the units can be in degrees, millimeters or whatever. The rest of the numeric fields must be integral. The motor names should be kept to nine characters or less, as the standard macros truncate them to fit a nine-character field when printing them out.

Field 8 is reserved.

Field 9, the flags field, contains several kinds of information. The lowest order two bits are used to enable particular operations on the selected motor. If bit 0 is set, the user can move the motor. If bit 1 is set, the user can change the software limits of the motor. Bits 2 and 3 are used by the *edconf* program to prevent users from changing certain configuration information. Bits 8 through 12 are used with the shared *config* file feature described below.

COUNTER PARAMETERS

Counter parameter assignment consists of key words of the form `CNT00`, `CNT01`, ... followed by 6 values. The `CNT` key words must be numbered consecutively starting at zero. The values are

1 Controller type (`KS3610`, `KS3640M`, `TS201`, ...)
2 Controller unit number (0, 1, ...)
3 Channel number in unit (0, 1, ...)
4 Counter function (`T`, `M` or `C` – Timer, Monitor or Counter)
5 Counter mnemonic (`sec`, `mon`, `det`, ...)
6 Counter name (`Seconds`, `Monitor`, `Detector`, ...)

Examples are:

```
#Fields        1  2  3  4        5  6
CNT00 =   KS3610  0  0  T      sec  Seconds
CNT01 =   KS3610  0  1  M      mon  Monitor
CNT02 =   KS3610  0  2  C      det  Detector
```

Valid controller types are currently:

```
AM9513    CAEN      INEL      KS3512    KS3610    KS3640C   KS3640T
LC1151    MIZAR     NONE      OR9XB     OR9XC     OR9XT     OR9XC
OR9XT     QS450     SFTWARE   TS201     VCT6      XRGCI_T
```

In field 4 (counter function), only one `T` and one `M` is allowed.

LINKED CONFIGURATIONS

An installation such as a synchrotron beamline uses many motors with most associated with beamline control. Spectrometers used for particular experiments have motors that aren't used in other experiments. To avoid having to merge the motor configurations and settings from one set of files to another when the spectrometer is changed, you can set things up so that a single version of the *config* and *settings* files will describe a number of different spectrometers. Here is how to set up the files:

1) If you already have several geometry configurations installed, you should make backup copies of the *config* and *settings* files from the current geometries.

2) If you already have several geometry configurations installed, remove the *config* and *settings* files from all but one of the geometry directories. Save the *config* from the other geometries to the remaining *config* file.

3) Set up hard links in all the geometry directories so that the *config* and *settings* in all the geometry directories refer to the same file. For example, if the files already exist in the *fourc* directory, use the commands

```
ln fourc/config surf/config
ln fourc/settings surf/settings
```

to create hard links in the *surf* directory. Don't use symbolic links.

4) Edit the *config* file by hand to add new control lines that assign numbers to the different geometries. These control lines must be before the lines that assign motor information. The format of the geometry control lines is as follows:

```
GEO0 = common
GEO1 = fourc
GEO2 = surf
GEO3 = fivec
etc.
```

The parameter `GEO0` always refers to the motors that are common to all the geometries. Subsequent lines assign consecutive numbers to the other geometries.

5) Now run *edconf*. The motor screen will have a new field that lets you assign a spectrometer geometry to each motor or to make the motor in common with all the spectrometers.

The hard links must be maintained for the shared *config* and *settings* file scheme to work. You can safely use *vi* and *cp* to manipulate the files. However, using *mv* will destroy the links. Also, the editor *ned* will destroy the links.

When running *edconf* with a geometry directory as an argument or when invoking the *config* macro from spec, use the new `G` command to toggle between displaying all the motors in the *config* file and just those motors used by the given geometry.

EXTRA PROTECTION

At some spec installations, the administrators need to prevent users from accessing or modifying the configuration of certain motors. The *edconf* program supports a wizard mode that allows such protection. If you type `^W` while running *edconf* you will be prompted for the wizard's password. If you enter it properly, you will be able to select additional levels of configuration protection.

While running spec, it is possible to enter a command that prompts for the wizard password, giving the current user access to the protected motors until the wizard mode is disabled. (See the *spec_par* help file for details on the `"specwiz"` option.)

Use the *wiz_passwd* utility provided with the spec package to set or change the wizard password, stored in a file named *passwd* in the spec auxiliary file directory (normally */usr/local/lib/spec.d*).

To prevent users from disabling the wizard protections by editing the *config* file by hand, you can use file protection features built into UNIX. One possibility is to make the *edconf* program set-user id *specwiz* or *specadm*, change the ownership of the *config* files to *specwiz* or *specadm*, and change the modes of the *config* files to `rw-r--r--`. Do that using commands (as super user) like

```
chown specadm edconf fourc/config surf/config ...
chmod u+s edconf
chmod 644 fourc/config surf/config ...
```

At present, reinstalling spec will undo the above mode changes, so that they will have to be repeated when spec is updated.