

Theta-Phi Stand and Muon Stand Control  
for Fermilab Test Fixtures, Hera-Zeus, E790

Table of Contents

|   |    |
|---|----|
| IPC-2100 Front Panel Use -----          | 2  |
| IPC-2100 Front Panel Parameters -----   | 3  |
| IPC-2100 Pin out on back of unit -----  | 4  |
| IPC-2100 Wiring Notes -----             | 5  |
| IPC-2100 RS422 Commands -----           | 6  |
| IPC-2100 General Notes -----            | 7  |
| IPC-2100 Theta-phi stand settings ----- | 8  |
| IPC-2100 Muon stand settings -----      | 9  |
|   |    |
| Software Interface -----                | 10 |
| GETDATA and SETDATA Commonality -----   | 11 |
| GETDATA Interface -----                 | 12 |
| SETDATA Interface -----                 | 13 |
| GENDATA and GENDATA DIAG Programs ----- | 14 |
| OPTO 147 and OPTO VAX Programs -----    | 14 |
| OPTO AC31 Commands -----                | 15 |
| OPTO AC31 General Notes -----           | 16 |
| OPTO AC7 General Notes -----            | 16 |
| OPTO Port General Notes -----           | 16 |
| Interface Modules -----                 | 17 |
| Diagnostic Modules -----                | 18 |
| Include files -----                     | 19 |
| Rebuilding -----                        | 20 |

Other manuals included:

*STARTUP PROCEDURE FOR MOTOR CONTROL*  
INTROL IPC-2100 POSITIONER OPERATIONS MANUAL  
OPTO-22 AC31 INTELLIGENT INTERFACE ADAPTER  
OPTO-22 AC7 RS232 TO RS422 ADAPTER CARD  
MOTOR CONTROLLER MANUAL

For more information contact:

Gary De Clute  
Lead Programmer  
UW-Physical Sciences Lab  
3725 Schnieder Drive  
Stoughton WI, 53589  
(608) 873-6651  
HEPNET Email PSL::GWD or PSLC::GWD  
Bitnet Email GWD@WISCPSL

or

Dick Loveless  
Senior Scientist  
UW-Physics Department  
University of Wisconsin  
Madison WI, 53705  
(608) 262-4767  
HEPNET Email WISHEP::LOVELESS  
Bitnet Email LOVELESS@WISCHEP

also

Theta-phi stand engineer:

Jeff Cherwinka  
Mechanical Engineer  
UW-Physical Sciences Lab  
3725 Schnieder Drive  
Stoughton WI, 53589  
(608) 873-6651  
HEPNET Email PSLC::JJC  
Bitnet Email JJC@WISCPSL

Muon stand engineer:

Ken Kriesel  
Mechanical Engineer  
UW-Physical Sciences Lab  
3725 Schnieder Drive  
Stoughton WI, 53589  
(608) 873-6651  
HEPNET Email PSLC::KRIESEL  
Bitnet Email KRIESEL@WISCPSL

Copyright (C) 1990, University of Wisconsin, Physical Sciences Lab.  
Permission is given to make copies for non-profit research purposes.

21-Feb-92 GWD

Refer to INTROL IPC-2100 POSITIONER OPERATIONS MANUAL as needed.

0) To regain control of a locked up unit, cycle power on main chasis:

push emergency stop button in  
pull emergency stop button out  
press reset button  
press run button

1) To move to a position:

press clear once (displays previous destination, if any)  
press clear again (clears display, displays 0)  
enter the sign (+ or -), this is required!  
enter the digits with optional decimal point and decimal digits  
press enter to begin motion

(velocity of motion is set as a parameter via the front panel)  
(acceleration is set as a parameter via the front panel)

2) To stop motion in progress:

press jog once (enters jog mode "J")  
wait for motion to stop  
press clear once (clears jog mode)

3) To home:

press home

You cannot home if the controller is waiting for a destination to be entered via the front panel. If home button does not work, cycle power.

(you cannot regain control until homing is complete)  
(velocity of home search is set as a parameter via front panel)  
(velocity of home limit search is set as a parameter via front panel)

4) To jog:

press jog once (enters jog mode "J")  
press up-arrow or down-arrow

Direction of arrow on arrow buttons does not always match the expected direction of motion. Each axis is different.

(hold down the arrow key to continue jog as long as desired)  
(velocity of jog is set as a parameter via the front panel)

5) To increment:

press jog once (enters jog mode "J")  
press jog again (enters increment mode "i")  
press up-arrow or down-arrow to increment in desired direction

(amount of increment is set as a parameter via the front panel)  
(velocity of increment is set as a parameter via the front panel)

## IPC-2100 Front Panel Parameters

Refer to INTROL IPC-2100 POSITIONER OPERATIONS MANUAL as needed.

\*\*\*\* Caution \*\*\*\*

Understand what these parameters do before you change them!  
 Incorrect settings can result in damage to the motor or motor  
 controller, or can lead to electrical or mechanical failure.

\*\*\*\*\*

To enter parameters:

press enter (unit will display CODE. 0)  
 enter 8788  
 press enter

To skip a parameter, press enter

To change a parameter, press clear, enter value, press enter

To exit press jog (display "J"), press clear

Parameters (in order of presentation) are:

PerP Percent of full speed when moving to a position  
 PerH Percent of full speed when searching for home switch  
 PerL Percent of full speed when final homing to home switch  
 ACCE Acceleration time in seconds (.01 to 6.0) to 100% speed  
 CC Constant of Calibration (2\*encoder\_counts\_per\_unit\_of\_measurement)  
 PerJ Percent of full speed when jogging  
 inpo unknown (always 0)  
 dLy unknown (always 0)  
 in. amount of movement in one increment  
 PerI Percent of full speed when incrementing  
 Baud baud rate of RS422 line  
 Id.no. Ascii character which identifies this axis  
 D.C. display calibrator ((1/CC) \* resolution\_of\_display)  
 dp.no. number of decimal points to display

Notes:

The percentages of full speed can be 1 to 250 percent.

On Theta-Phi and Muon stands: Do not exceed 100%. Above 100%, the rated RPM capacity of motors is exceeded, and mechanical and electrical loading of other components may also exceed specifications.

Acceleration time is measured in seconds to 100% speed (.01 to 6.0). Do not go below 0.5 seconds without consulting Jeff Cherwinka, PSL Mechanical Engineer (Theta-phi stand) or Ken Kriesel, PSL Mechanical Engineer (Muon Stand). Do not go above 1.5 seconds without consulting Gary De Clute, PSL programmer.

CC., D.C., and dp.no., control the display and unit of measurement.

On Theta-Phi and Muon stands: CC will eventually be calculated for 1mm travel on the surface of the loaded modules. D.C. is determined by CC. and Dp.no., and should be set so that no more than 6 of the eight digits of the display are used over the full range of motion.

Baud, and Id.no., control RS422 access. Baud cannot be higher than 4800. Baud and Id.no. cannot be changed without changing other hardware and software.

Parameter in., is measured in units as displayed on the front panel. It can be set to any convenient value. Parameters inpo and dLy are mysterious and should always be 0.

## IPC-2100 Pin out on back of unit

Refer to INTROL IPC-2100 POSITIONER OPERATIONS MANUAL as needed.

## TB1

|   |  |             |
|---|--|-------------|
| 2 | -10 to +10 V D.C. reference (isolated) | (see below) |
| 8 | gnd for reference (isolated)           | (see below) |

## TB2

|    |   |                                       |
|----|---|---------------------------------------|
| 1  | optional input reference frequency            | (not used)                            |
| 2  | in position                                   | (see below)                           |
| 3  | RXD RS232 (not available on our units)        | (not used)                            |
| 4  | TXD RS232 (not available on our units)        | (not used)                            |
| 6  | reset button                                  | (to activate: ground to earth ground) |
| 7  | direction button                              | (to activate: ground to earth ground) |
| 8  | jog button                                    | (to activate: ground to earth ground) |
| 9  | home limit switch                             | (see below)                           |
| 10 | home button                                   | (to activate: ground to earth ground) |
| 11 | increment button                              | (to activate: ground to earth ground) |
| 12 | encoder channel A                             | (see below)                           |
| 13 | + 5 V D.C. to encoder                         | (not used)                            |
| 14 | encoder channel B                             | (see below)                           |
| 15 | common gnd all control signals (to earth GND) |                                       |
| 16 | optional index marker from encoder            | (see below)                           |
| 17 | steps output (stepper motors)                 | (not used)                            |
| 18 | direction output (stepper motors)             | (not used)                            |
| 19 | 120 V A.C. Polarity is not relevant           |                                       |
| 20 | 120 V A.C. Polarity is not relevant           |                                       |

## TB3

|   |       |       |
|---|-------|-------|
| 1 | TXD + | RS422 |
| 3 | TXD - | RS422 |
| 4 | RXD + | RS422 |
| 5 | RXD - | RS422 |
| 6 | COM   | RS422 |

## IPC-2100 Wiring Notes

Refer to INTROL IPC-2100 POSITIONER OPERATIONS MANUAL as needed.

To reverse direction of motor

1) Turn off power.

On back of Introl IPC-2100:

2) Switch polarity of outputs to motor: Swap TB1 pin 8 with TB1 pin 2. Output is isolated, so it does not matter if TB2 pin 8 is and earth ground, but do NOT wire both TB1 pin 8 and TB1 pin 2 to earth ground. Only one of TB1 pin 2 and TB1 pin 8 can be earth ground.

3) Switch encoder channel A with encoder channel B: Swap TB2 pin 12 with TB2 pin 14.

4) Restore power, motor should now run backwards from previous orientation.

## IPC-2100

For incremental encoders without marker pulse

When an incremental encoder has a marker pulse, the IPC-2100 will search for the home switch, and will then back off and continue moving until it detects the first marker pulse, which becomes the zero point. If there is no marker pulse, this results in an infinite search. To fix this, wire as follows:

- 1) Jumper TB2 pin 9 to TB2 pin 16  
(instead of hooking TB2 pin 16 to the incremental encoder)
- 2) An internal jumper inside the IPC-2100 must also be changed. It is jumper J on the bottom board. It must be moved to EXT. It is necessary to disassemble the IPC-2100 to reach this jumper.

## IPC-2100

Behavior of push button hookups

|                  |  |
|------------------|--|
| Reset            | Reset IPC-2100                           |
| Home button      | Cause home, as from front panel          |
| Jog button       | Causes jog, see direction button         |
| Increment button | Causes increment, see direction button   |
| Direction button | Determines direction of jog or increment |

Direction button latches after jog or increment has begun, so it is not necessary to continue grounding it after a negative motion is started.

Reset: when normal motion in progress: stop motion  
 when stationary: clears position (position becomes 0.0)  
 when jogging: reset does not appear to affect jog  
 when incrementing: reset does not appear to affect increment

## IPC-2100

Motion in progress alarm

We are using the "in position" line, TB2 pin 2 to control an audible alarm which indicates that the motor is running. When motion is in progress, a relay is actuated which sounds the alarm.

## IPC-2100 RS422 Commands

Refer to INTROL IPC-2100 POSITIONER OPERATIONS MANUAL as needed.

The Id Number of the unit is a hexadecimal ascii value. Any ascii value can be used. We will assume 41 ("A").

<esc> is the escape character (1B hex)  
 <sign> + or - sign  
 <position> a number with an optional sign and optional decimal point  
 <cr> carriage return (1D hex)  
 <lf> line feed (1A hex)

There is some question about what exactly is coming back from the IPC-2100:

<?> an unknown character (what it might be is indicated)

0) To regain control of a locked up unit, cycle power.

1a) To move to a position with report of success:

send: <esc>A<position><cr>  
 immediate reply: A<position><cr>A<nul?> (echo)  
 at end of move: INPO.<cr><lf>

1b) To move to a position with no report of success:

send: <esc>A<position>,<cr>  
 immediate reply: A<position><cr> (echo)

1c) Multi-axis move with report of success of last axis:

send: <esc>A<position>,B<position><cr>  
 immediate reply: A<position>,B<position><cr>B<nul?> (echo)  
 at end of move: INPO.<cr><lf>

1d) Multi-axis move with no report of success:

send: <esc>A<position>,B<position>,<cr>  
 immediate reply: A<position>,B<position>,<cr> (echo)

2) To stop motion in progress:

send: <esc>AJ<cr>  
 immediate reply: AJ<cr> (echo)

3) To home: can't be done

4) To jog:

send: <esc>AJ<sign><cr>  
 immediate reply: AJ<sign><cr> (echo)  
 to stop send: S<cr>  
 immediate reply: SINPO.<cr><lf>

5) To increment: can't be done

6) To request current position:

send: <esc>A?<cr>  
 immediate reply: A?INPO.<cr>A@<position><cr>

Refer to INTROL IPC-2100 POSITIONER OPERATIONS MANUAL as needed.

1) Lock up is frequent

If the introl fails to respond to commands from the front panel or via the RS422 line, it may be "locked up". It is necessary to cycle power to regain control of the unit. This appears to occur fairly frequently when the normal operating parameters are exceeded or invalid commands are entered or sent to it. Be cautious, therefore, when entering or sending commands. It is difficult to damage the unit, but easy to lock it up.

2) Bench testing difficult

The IPC-2100 cannot be meaningfully bench tested apart from a servo system. The most you can do is to make it assert an arbitrary positive voltage (< 10 V) when told to move to a positive location, or to an arbitrary negative voltage (> -10V) when told to move to a negative location. This can be tested from the front panel and/or the via RS422. Use only very small positions (like 1.0). Measure the voltage across TB1 pin 2 and TB1 pin 8. To test any more, you must place the IPC-2100 in a working servo system.

3) Documentation is poor

Much of what is known about the detailed behavior of the IPC-2100 has been determined by calling Introl and/or through experimentation. In general, any information originating from Introl should be tested before it is accepted as correct.

4) Contact person at Introl

Ali Shams (716) 434-6919, Fax: (716) 434-1911.  
Introl is open 8:00-16:30 (Eastern time) MTWT, 8:00-11:30 Friday.  
Ali is very helpful, and easy to reach.

## IPC-2100 Theta-Phi Stand Settings

Refer to INTROL IPC-2100 POSITIONER OPERATIONS MANUAL as needed.

Internal switch settings:

Do not mess with these settings!

|     | Horizontal Axis | Vertical Axis |
|-----|-----------------|---------------|
| SW1 | 3 on            | 3 on          |
| SW2 | 4 on            | 4 on          |
| SW3 | 3,4,6,7 on      | 3,4,6,7 on    |

Front panel settings:

Understand these settings before you mess with them!

|        | Horizontal Axis | Vertical Axis |
|--------|-----------------|---------------|
| PerP   | 55%             | 95%           |
| PerH   | 55%             | 95%           |
| PerL   | 2%              | 2%            |
| ACCE   | 1.0             | 1.0           |
| CC     | 60              | 60            |
| PerJ   | 55%             | 95%           |
| inpo   | (always 0)      | (always 0)    |
| dLy    | (always 0)      | (always 0)    |
| in.    | 5.22            | 9.36          |
| PerI   | 55%             | 95%           |
| Baud   | 4800            | 4800          |
| Id.no. | 41              | 42            |
| D.C.   | 1.6667          | 1.6667        |
| dP.no. | 1               | 1             |

Notes:

PerP, PerH, PerJ, and PerI should all be the same.

ACCE should not be less than 0.5 or greater than 1.5. Consult Jeff Cherwinka before lowering this value. Too small a value can result in excessive loading when accelerating and deaccelerating. Consult Gary De Clute before raising this value. Too large a value can result in overrunning a switch and encountering the next switch (or perhaps the hard stop.)

CC., D.C., and dP.no. depend upon the units chosen to measure position. They are currently set for motor revolutions. Eventually these will be changed to represent 1 mm of travel on the surface of the installed modules.

Parameter in. represents an increment of motion. It should eventually be set to 1mm, or 5 cm, or 20 cm of travel on the surface of the installed modules, depending upon the axis and desired use of the increment function.

Baud must be 4800. This is highest speed permitted. It must be coordinated with the OPTO-22 AC31 board jumper settings.

Parameter Id.no. can be changed, but this must be coordinated with the software interface. No two IPC-2100's at Lab E should have the same Id.no.

## IPC-2100 Muon Stand Settings

Refer to INTROL IPC-2100 POSITIONER OPERATIONS MANUAL as needed.

Internal switch settings:

Do not mess with these settings!

|     | Horizontal Axis | Vertical Axis |
|-----|-----------------|---------------|
| SW1 | 3 on            | 3 on          |
| SW2 | 4 on            | 4 on          |
| SW3 | 3,4,6,7 on      | 3,4,5,6,7 on  |

Front panel settings:

Understand these settings before you mess with them!

|        | Horizontal Axis | Vertical Axis |
|--------|-----------------|---------------|
| PerP   | 90%             | 90%           |
| PerH   | 90%             | 90%           |
| PerL   | 30%             | 30%           |
| ACCE   | 1.0             | 1.0           |
| CC     | 60              | 60            |
| PerJ   | 90%             | 90%           |
| inpo   | (always 0)      | (always 0)    |
| dLy    | (always 0)      | (always 0)    |
| in.    | 100             | 100           |
| PerI   | 90%             | 90%           |
| Baud   | 4800            | 4800          |
| Id.no. | 43              | 44            |
| D.C.   | 1.6667          | 1.6667        |
| dP.no. | 2               | 2             |

Notes:

PerP, PerH, PerJ, and PerI should all be the same.

ACCE should not be less than 0.1 or greater than 1.5. Consult Gary De Clute before raising this value. Too large a value can result in overrunning a switch and encountering the next switch (or perhaps the hard stop.)

CC., D.C., and dP.no. depend upon the units chosen to measure position. They are currently set for motor revolutions. Eventually these will be changed to represent 1 mm of travel on the surface of the installed modules.

Parameter in. represents an increment of motion. It should eventually be set to 1mm, or 5 cm, or 20 cm of travel on the surface of the installed modules, depending upon the axis and desired use of the increment function.

Baud must be 4800. This is highest speed permitted. It must be coordinated with the OPTO-22 AC31 board jumper settings.

Parameter Id.no. can be changed, but this must be coordinated with the software interface. No two IPC-2100's at Lab E should have the same Id.no.

## Software Interface

Refer to comments located in the various C modules as needed.

The theta-phi and muon stands have a software interface on the Motorola 147 computer. The higher levels of this interface are beyond the scope of this manual, as is the connection of the Motorola 147 and the VAX/VMS computers.

This is how it works:

- 1) A "setdata" or "getdata" request is generated in some manner on some computer. The request makes its way (if necessary) to the Motorola 147, and a process is forked (if necessary) to service the request. Some requests are generated directly on the 147. When using the diagnostic programs, the request are generated on the 147 and no forking is done.
- 2) The process which will service the request will call one of four functions which will perform the operation:

```
getdata_thph  getdata_muon  setdata_thph  setdata_muon
```

- 3) A status value is returned by these functions, which makes its way back to wherever the request originated, along with any "getdata" parameters that were requested ("setdata" does not return parameter values).

The "getdata" and "setdata" parameter lists are very general in nature, and all have the same form:

```
(systype, cardtype, address, func, numvals, data).
```

The values of interest here are:

```
char systype[5];          "muon" or "thph"
char cardtype[9];        "IPC-2100"
int address;             (specially encoded)
int func;                (different for setdata and getdata)
int numvals;             (depends upon func)
float data[numvals];
```

The interface consists of two libraries: introl.l and opto.l:

Introl.l consists of all routines needed to perform setdata and getdata for the muon or thph stands, using the opto.l library routines. Most details of the RS422 chain and the opto boards are hidden from the introl.l routines. The exceptions are obvious and have been noted elsewhere.

Opto.l consists of all routines needed to communicate with an OPTO AC31 board on an RS422 chain leading from a serial port on a Motorola 147 computer running the OS/9 operating system. Opto.l could be expanded to allow communications with other opto board types. Nothing in opto.l is specific to the introl controllers.

Using the interface:

An application program required to service getdata and/or setdata calls for the muon and/or thph stands, should make the desired calls noted above, and link to the two libraries noted above. Then simply fork or run the program.

The only caveat to this is that the program HWC\_OS9 must have been run at least once since booting the 147, in order that certain required "events" be created. If this requirement cannot be met, see link\_gendata\_diag.sh for an example of how to link in such a way that the "events" can be circumvented. If you do circumvent the "events", one and only one program can be running that communicates in any way with the OPTO RS422 chain!

## GETDATA and SETDATA Commonality

Refer to comments located in the various C modules as needed.

The following are commonalities in getdata and setdata calls to control the muon or thph stands.

Systype: either "muon" or "thph".

The systype names can be changed by editing introl sanity.h and rebuilding. Software elsewhere in the system may also need to be changed.

Cardtype: always "IPC-2100".

The cardtype name can be changed by editing introl sanity.h and rebuilding. Software elsewhere in the system may also need to be changed.

Address: a longword divided into 4 bytes (from lowest to highest):

|                  |   |
|------------------|---|
| <u>INTROL_1</u>  | Introl Id.no. of primary axis             |
| <u>INTROL_2</u>  | Introl Id.no. of secondary axis           |
| <u>OPTO_UNIT</u> | Opto address of AC31 board                |
| <u>OPTO_PORT</u> | Serial port number of line to opto boards |

INTROL\_1 (primary axis) must be non-zero.

If systype is "thph", a non-zero INTROL\_1 or INTROL\_2 must be 41 or 42 (hex).  
If systype is "muon", a non-zero INTROL\_1 or INTROL\_2 must be 43 or 44 (hex).  
In any case, INTROL\_1 and INTROL\_2 cannot be the same.

The required values of INTROL\_1 and INTROL\_2 can be changed by editing introl sanity.h and rebuilding. Note that if these required values are to be changed, the programmed Id.no. of the introl controllers must be changed to match the new values. Software elsewhere in the system may also need to be changed.

OPTO\_UNIT must fall between 0 and 8.

OPTO\_PORT must fall between 1 and 3.

The limits of OPTO\_PORT and OPTO\_UNIT can be changed by editing opto sanity.h and rebuilding. Note that even though these define the acceptable ranges, only one OPTO\_PORT can be functional, and only one OPTO\_UNIT works for the muon stand, and only one different OPTO\_UNIT works for the thph stand. To change any of these values requires changes in hardware and software.

Example: to control both "muon" axes:

Opto port of 01 (hex)  
Opto unit of 02 (hex)  
Introl "D" is secondary axis, 44 (hex)  
Introl "C" is primary axis, 43 (hex)

Address is: 01024443 (hex)

Example: to control a single "muon" axis:

Opto port of 01 (hex)  
Opto unit of 02 (hex)  
Introl "C" is axis of interest, 43 (hex)

Address is: 01020043

## Data:

All setdata and getdata data items related to the thph or muon stands are of the following two types:

A position (an incremental encoder reading or an introl target position):

Positions are signed floating point numbers. Accurate to within about .1. Units are in shaft revolutions of the motor. The zero point is located with the "home" switch. Note that units may change depending upon the constant of calibration and display calibration setting of the IPC-2100.

An accuracy (measured in units of introl target position counts):

Desired accuracies are positive floating point numbers in the range 1.0 and greater. Accuracies of less than 1.0 are impractical. Accuracies greater than 10.0 are risky in that they could allow a significant error to escape detection. Default accuracy is 3.0. Note that units may change depending upon the constant of calibration and display calibration setting of the IPC-2100. The default value is adequate for most purposes.

## GETDATA Interface

Refer to comments located in the various C modules as needed.

```
int getdata_thph(systype, cardtype, address, func, numvals, data);
  or
int getdata_muon(systype, cardtype, address, func, numvals, data);

char systype[5];          "muon" or "thph"
char cardtype[9];        "IPC-2100"
int address;
int func;
int numvals;
float data[];
```

Function codes and return values:

| Func | Operation |
|------|-----------|
|------|-----------|

|    |  |
|----|--|
| 1  | Return incremental encoder position of axis      |
| 11 | Return incremental encoder positions of two axes |

| Func | Numvals | Data |
|------|---------|------|
|------|---------|------|

|    |   |  |
|----|---|--|
| 1  | 1 | [1] Incremental encoder reading of primary axis  |
| 11 | 2 | [1] Incremental encoder reading of primary axis<br>[2] Incremental encoder reading of secondary axis |

GETDATA timing:

| Func |  |
|------|--|
| 1    | 2.3 seconds to read position of one axis |
| 11   | 4.6 seconds to read position of two axes |

## SETDATA Interface

Refer to comments located in the various C modules as needed.

```
int setdata_thph(systype, cardtype, address, func, numvals, data);
  or
int setdata_muon(systype, cardtype, address, func, numvals, data);

char systype[5];          "muon" or "thph"
char cardtype[9];        "IPC-2100"
int address;
int func;
int numvals;
float data[];
```

| Func | Operation |
|------|-----------|
|------|-----------|

|    |  |
|----|--|
| 2  | Move a single axis to an incremental encoder position, wait for move |
| 3  | Move a single axis to an incremental encoder position, do not wait   |
| 4  | Halt motion on a single axis   |
| 12 | Move two axes to incremental encoder coordinates, wait for move      |
| 13 | Move two axes to incremental encoder coordinates, do not wait        |
| 14 | Halt motion on two axes.   |

## Function codes and set values:

| Func | Numvals | Data   |
|------|---------|--|
| 2    | 1 or 2  | [1] Incremental encoder target for primary axis<br>[2] Primary axis positioning accuracy (optional)  |
| 3    | 1       | [1] Incremental encoder target for primary axis  |
| 4    | 0       | none   |
| 12   | 2 or 4  | [1] Incremental encoder target for primary axis<br>[2] Incremental encoder target for secondary axis<br>[3] Primary axis positioning accuracy (optional)<br>[4] Secondary axis positioning accuracy (optional) |
| 13   | 2       | [1] Incremental encoder target for primary axis<br>[2] Incremental encoder target for secondary axis   |
| 14   | 0       | none   |

## SETDATA timing:

## Func

|    |  |
|----|--|
| 2  | 5.5 seconds to start motion on one axis<br>6.3 seconds polling interval<br>6.3 to approximately 18 seconds to detect end of motion |
| 12 | 5.5 seconds to start motion on two axes<br>8.6 seconds polling interval<br>8.6 to approximately 30 seconds to detect end of motion |
| 3  | 5.5 seconds to start motion on one axis  |
| 13 | 5.5 seconds to start motion on two axes  |
| 4  | 4.2 seconds to halt one axis   |
| 14 | 4.2 seconds to halt two axes   |

## GENDATA and GENDATA\_DIAG Programs

Refer to comments located in the various C modules as needed.

These two programs are available to "generate" getdata and setdata calls for the muon and thph stands. To use:

```
chd /h0/hwc/introl
run gendata           or
run gendata_diag
```

The program prompts for getdata or setdata.

The program then prompts for each parameter of the call to be generated, in the order the parameters occur in the parameter list.

After all parameters have been entered, the parameters are displayed, and the call is made.

Status is displayed as well as any return parameters from a getdata (setdata has no return parameters.)

## OPTO\_147 and OPTO\_VAX Programs

These two programs are available for directly communicating with the Introl controllers via an OPTO-MUX AC31 board.

The RS232 port of the OPTO AC7 board must be attached to a serial port of the machine in question, and the serial port must be properly configured. The AC31 must be attached to the RS422 chain leading from the AC7 board.

On the 147, use OPTO\_147.C (The opto port is /T1, recompile to change this.)  
On the VAX, use OPTO\_VAX.C (The opto port is RRB4:, recompile to change this.)

Examples of using OPTO\_147 or OPTO\_VAX programs:

Assume the OPTO AC31 is at opto address 03 and introl "C" is of interest.

To tell introl to move to 567.3

```
>03B           !Reset opto AC31
A             !Reply is A
>03S\x1BC567.3\r !Send <esc>C567.3<cr> to introls
A             !Reply is A (chars sent)
>03R           !Get num chars waiting
A0007xx       !7 characters are waiting
>03N7         !Read 7 characters
AC567.3xx     !Reads echo from introls
etc...
```

To read position from introl

```
>03B           !Reset opto AC31
A             !Reply is A
>03S\x1BC?\r  !Send <esc>C?<cr> to introls
A             !Reply is A (chars sent)
>03L           !Read string from introls
AC?INPO.xx    !Reads echo and INPO.
>03L           !Read string from introls
A@ 567.34xx   !Introl reports position
```

You have to play around with this for awhile to get used to it. See the sections titled OPTO AC31 Commands and IPC-2100 RS422 Commands for more details. Keep in mind that all commands and replies to/from the Introls are imbedded inside commands and replies to/from the AC31 board.

## OPTO AC31 Commands

Refer to the OPTO-22 AC31 INTELLIGENT INTERFACE ADAPTER manual as needed.

This is VERY brief summary of the AC31 command language and how to use it to communicate with the introls. See the OPTO AC31 manual for details.

All opto commands must be preceded by >xx where xx is the hex opto address of the opto board. Opto commands must be terminated with a checksum and <cr>. The checksum and <cr> are provided by the software. It is good idea to use the AC31 reset command prior to sending each command to the introls. This will clear the AC31 buffers.

Useful OPTO AC31 board commands are:

|         |   |
|---------|---|
| >xxA    | Power up clear (must be first command after power up) |
| >xxB    | Reset (also clears buffer from introls)               |
| >xxL    | Read a string (terminated by <cr>)                    |
| >xxR    | Return number of chars waiting                        |
| >xxNxx  | Read number of chars waiting (or until <cr>)          |
| >xxS... | Send to introls (... is any chars)                    |

To send special characters to introls use the following:

|         |      |
|---------|------|
| To send | Use  |
| <cr>    | \r   |
| <esc>   | \x1B |

Replies from the OPTO AC31 are of three forms:

- 1) A                    Indicates success, no further data.
- 2) Adata??           Indicates success. Data is what is returned. ?? is checksum.
- 3) Nxx                Indicates failure. Reason is coded into xx (hex).

Don't forget, you must tell the OPTO AC31 board to send and read strings. You cannot communicate directly with the introls.

Don't forget to send an <esc> as the first character of each command to the introls. Use the sequence \x1B to represent <esc>.

Don't forget to terminate each command to the introls with <cr>. Use the sequence \r to represent <cr>.

The software provides checksum and trailing <cr> for each OPTO AC31 command, and strips off the final <cr> returned by the AC31 board. Program displays the strings transmitted and received (with checksums). Checksum is checked on all strings received.

## OPTO AC31 General Notes

Refer to the OPTO-22 AC31 INTELLIGENT INTERFACE ADAPTER manual as needed.

## 1) Inconsistencies in the documentation

a) The documentation claims that all replies from the OPTO AC31 will be of two forms:

A<data><checksum><cr> where data is optional

or Nxx<cr> where xx is a two digit hex error code

In fact, there are three forms. The two above plus:

A<cr>

Which appears to occur whenever there is no data in the reply. The checksum is dispensed with in this case.

b) The instructions for setting group A jumpers depending upon whether the AC31 is in the middle or end of the RS422 link are specific to the AC31. Similar documentation for some other OPTO boards is not very clear. Trust only the AC31 documentation for the AC31.

## OPTO AC7 General Notes

Refer to the OPTO-22 AC7 RS232 TO RS422 ADAPTER CARD manual for details.

The correct jumper settings on the AC7 were determined by trial and error (mostly error.)

The AC7 board is completely transparent to the software.

## OPTO Port Setup

Refer to the file /h0/startup for an example of this.

In the startup shell of OS/9, do not `iniz` and `tsmon` the opto port. If you do, the port will not be available for the application to use. Apparently all default values are adequate.

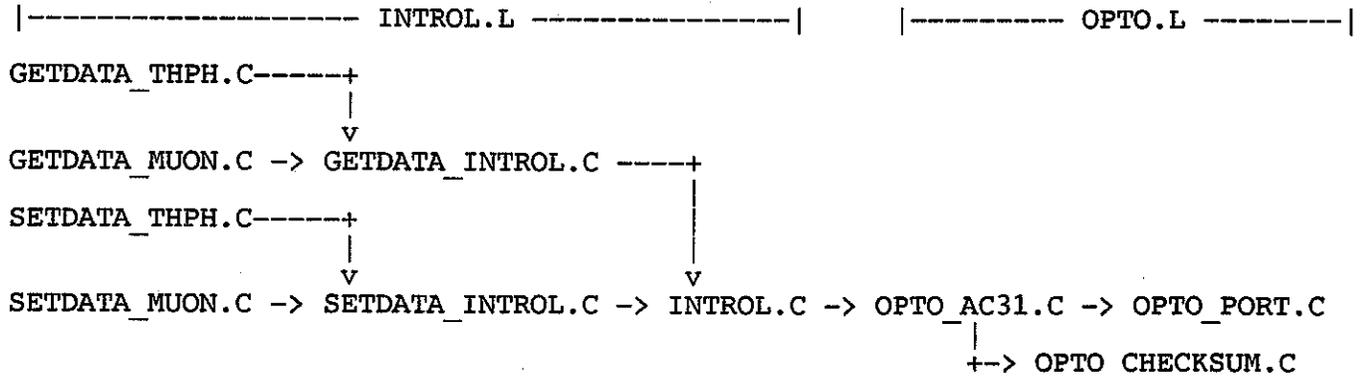
## Interface Modules

Refer to the files located on /h0/hwc/introl as needed.

The software is located on /h0/hwc/introl and all files there are directly related to the interface. For details please examine the code. Fairly extensive comments have been provided. Information in this document may become out of date. The code on /h0/hwc/introl should be considered correct if inconsistencies between the code and the documentation are discovered.

GETDATA\_THPH.C and SETDATA\_THPH.C control the thph hardware system.  
GETDATA\_MUON.C and SETDATA\_MUON.C control the muon hardware system.

Procedure call hierarchy:



Exception: GETDATA\_INTROL.C and SETDATA\_INTROL.C call the routine `opto_port_init()` in the module OPTO\_PORT.C. This is necessary to assure that the `opto_port_init` call is made only once for each getdata or setdata process. Also, the events manipulated by OPTO\_PORT.C must have been previously created before any calls to these routines are made.

## Diagnostic modules

Refer to the files located on /h0/hwc/introl as needed.

GENDATA "generates" a getdata or setdata call. These in turn generate the appropriate getdata or setdata for the specified hardware system.

Use GENDATA for testing the thph or muon hardware systems while other optomux activity is expected. The "events" created by HWC\_OS9 must exist, which means that HWC\_OS9 must have been run at least once since booting the 147.

Example:

```
chd /h0/hwc/introl
run gendata
```

Use GENDATA\_DIAG for testing thph or muon hardware systems while other optomux activity is not expected. The "events" created by HWC\_OS9 are not needed.

Example:

```
chd /h0/hwc/introl
run gendata_diag
```

Procedure call hierarchy:

```
GENDATA.C --+--> SETDATA.C --+--> SETDATA_THPH.C --> etc.
          |                               |
          |                               +--> SETDATA_MUON.C --> etc.
          +--> GETDATA.C --+--> GETDATA_THPH.C --> etc.
                               |
                               +--> GETDATA_MUON.C --> etc.
```

GENDATA\_DIAG.C --> same as above but linked to opto\_port\_diag.r instead of opto\_port.r

Low level diagnostics:

OPTO 147.C can be used to directly communicate with any optomux board. (From the VAX use OPTO VAX.C). The Opto AC7 must be attached to a serial port on the computer from which the program is to be run, and the port must be properly configured. The source of OPTO\_147 (or OPTO\_VAX) may have to be modified depending upon which port is used. OPTO\_147.C should generally be set up and ready to go. OPTO\_VAX will seldom (if ever) be used. These are primitive routines designed to diagnose hardware problems.

Example (147):

```
chd /h0/hwc/introl
run opto_147
```

Example (VAX):

```
set def [declute.pro.daq]
run opto_vax
```

## Include files

Refer to the files located on /h0/hwc/introl as needed.

```

GETDATA_MUON.C, GETDATA_THPH.C, SETDATA_MUON.C SETDATA_THPH.C
  fortran.h          !Fortran-like C code
  introl_sanity.h    !Introl sanity checks

GETDATA_INTROL.C
  fortran.h          !Fortran-like C code
  introl_sanity.h    !Introl sanity checks
  opto_sanity.h      !Opto sanity checks
  getdata_codes.h   !Function codes for getdata call

SETDATA_INTROL.C
  fortran.h          !Fortran-like C code
  introl_sanity.h    !Introl sanity checks
  opto_sanity.h      !Opto sanity checks
  setdata_codes.h   !Function codes for setdata call

INTROL.C
  fortran.h          !Fortran-like C code
  introl_timers.h    !Timers for introl routines

OPTO_AC31.C
  fortran.h          !Fortran-like C code
  opto_timers.h      !Timers for opto routines

OPTO_CHECKSUM.C, OPTO_PORTS.C
  fortran.h          !Fortran-like C code

```

## Rebuilding

Refer to the files located on /h0/hwc/introl as needed.

The build shells are organized as follows:

```

build_opto_147.sh
  (creates_opto_147_executable)

build_all.sh
  build_opto.sh
    cc opto_ac31.c
    cc opto_checksum.c
    cc opto_port.c
    cc opto_port_diag.c
    merge_opto.sh
      (creates_opto.l)
  build_introl.sh
    cc setdata_thph.c
    cc getdata_thph.c
    cc setdata_muon.c
    cc getdata_muon.c
    cc setdata_introl.c
    cc getdata_introl.c
    merge_intr0l.sh
      (creates_introl.l)
  build_gendata.sh
    cc gendata.c
    cc setdata.c
    cc getdata.c
    link_gendata.sh
      (creates_gendata_executable)
  link_gendata_diag.sh
    (creates_gendata_diag_executable)

```

To rebuild everything:

```

  build_all.sh          and
  build_opto_147.sh

```

To entirely rebuild specific librarys:

```

  build_introl.sh      or
  build_opto.sh

```

To change only one module in a library:

```

  cc 'module'
  link_introl.sh       or
  link_opto.sh

```

To link higher levels of software, link against:

```

  introl.l            and
  opto.l

```

To rebuild diagnostic programs (gendata, gendata\_diag):

```

  build_gendata.sh
  (note: this links against introl.l and opto.l)

```

To rebuild low level diagnostic program (opto\_147):

```

  build_opto_147.sh
  (note: this does not link against the libraries)

```

```
$ set verify
$!mail kotanski.txt zeus02::kotanski /subj="Opto AC31 files. Start of
$ mail build_opto_147.sh zeus02::kotanski /subj="build_opto_147.sh, 1 of 10
$ mail opto_I47.c_zeus02::kotanski /subj="opto_I47.c, 2 of 10
$ mail build_opto.sh zeus02::kotanski /subj="build_opto.sh, 3 of 10
$ mail merge_opto.sh zeus02::kotanski /subj="merge_opto.sh, 4 of 10
$ mail fortran.h zeus02::kotanski /subj="fortran.h, 5 of 10
$ mail opto_timers.h zeus02::kotanski /subj="opto_timers.h, 6 of 10
$ mail opto_ac31.c zeus02::kotanski /subj="opto_ac31.c, 7 of 10
$ mail opto_port.c zeus02::kotanski /subj="opto_port.c, 8 of 10
$ mail opto_port diag.c zeus02::kotanski /subj="opto_port diag.c, 9 of 10
$ mail opto_checksum.c zeus02::kotanski /subj="opto_checksum.c, 10 of 10
$ mail opto.doc zeus02::kotanski /subj="Description of Opto AC31 f
$ set noverify
```