

Adept Binary IO Reference Guide

By Joel Malander
Mechanical Engineering
Colorado State University

Getting Started

As the name implies the adept robot is a highly adaptive manufacturing centered robot. A significant portion of the robots flexibility comes from a large numbers of external IO channels. These channels are intended to serve in the robots integration with assembly lines and facilitate safety measures in dangers areas. Of the many types of IO controllers available to the Adept user this guide is concerned with only one. That is, this guide will try to relay all the important information given in the manuals about the robot's binary IO capability.

Hardware

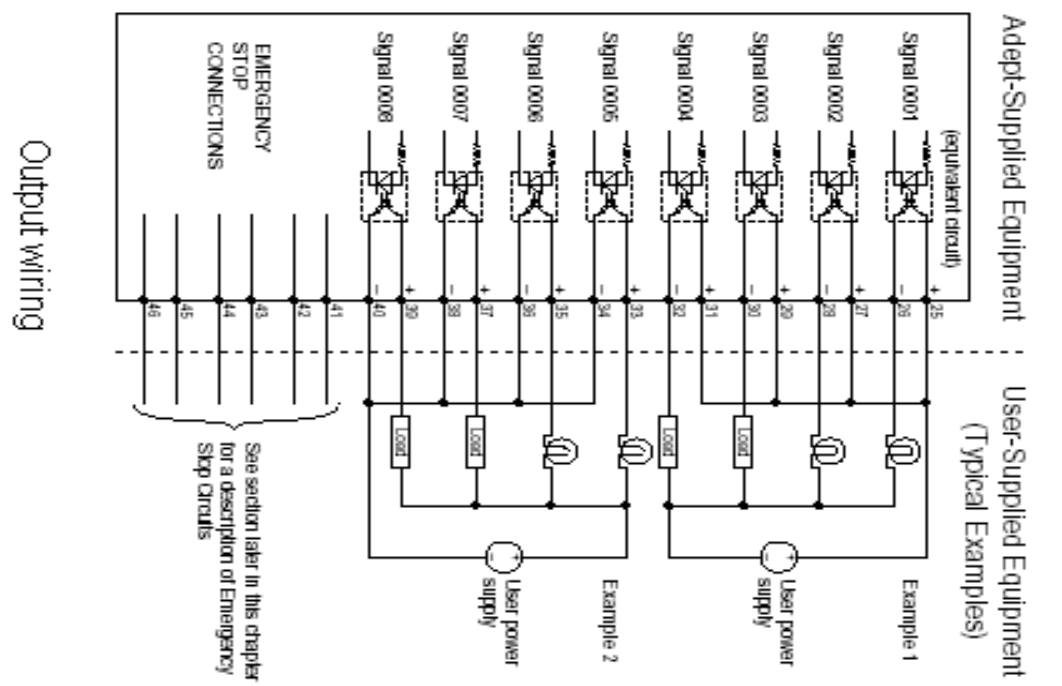
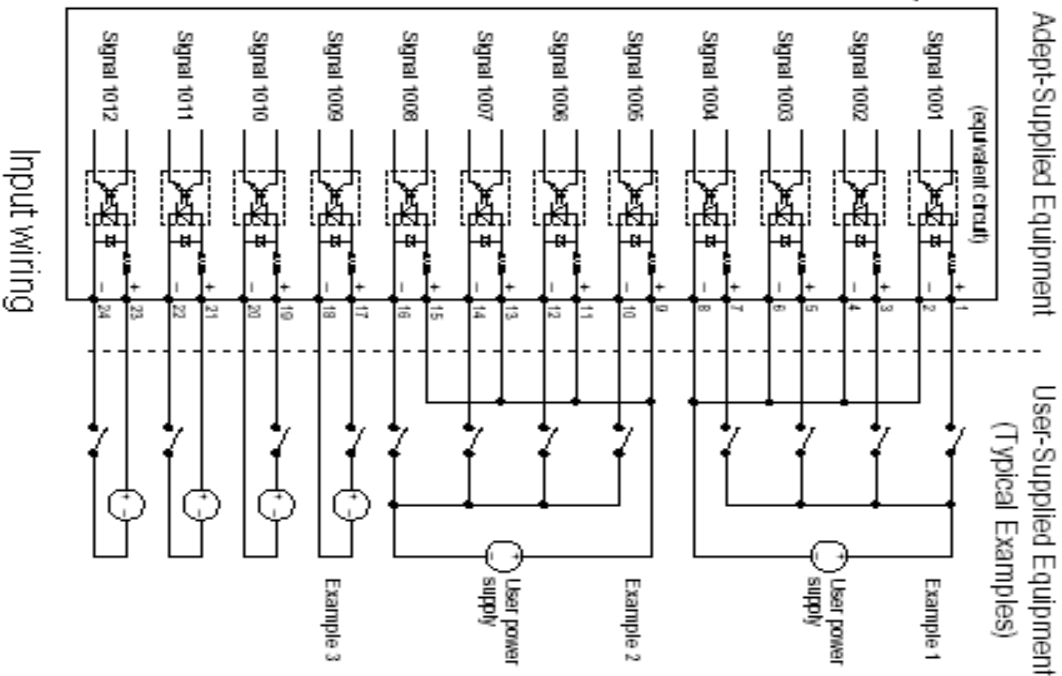
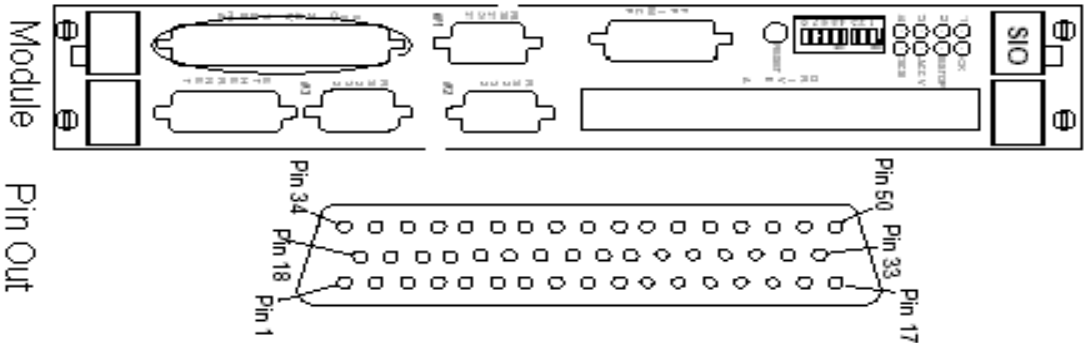
Binary IO channels can be found on two modular boards supplied by Adept. The first is the System Input/ Output module (SIO). This module acts as an interface for all IO hardware including diskettes drives, hard drives, serial ports, and binary IO ports. The second board is the Digital IO module (DIO). This module acts as an auxiliary controller in situations where all binary IO channels on the SIO are in use. When working with the adept IO channels the following list of tips are important to keep in mind.

Tips for binary IO users

- All IO channels on both SIO and DIO modules are opto-isolated from the controller circuitry and their neighbors. This means if excessive current is put through a channel only that channel will fail.
- Only the input signals on the SIO module can be used with the REACT/REACTI commands.
- The first three input signals on the SIO are scanned at a faster rate and can be used with the INT.EVENT and LATCH program instructions. These instructions are intended for quick responses situations.
- No current limiting resistors are need on input signals because the internal impedance of the channels is sufficient to limit current properly.
- Care should be used in determining the load on output channel circuitry because the current is not limited on any of the output channels.
- All IO signals will require external power supplies.
- The SIO and DIO binary IO channels have different electrical specifications and should not be confused.

SIO Binary IO

The digital IO port on the SIO module consists of a 50-pin, high-density, D-sub female connector. This port contains 12 input channels, 8 output channels, 2 E-stop input channels and 1 E-stop output channel. The pin and circuit information can be seen on the next page.



The input and output wiring diagrams for the SIO module show several examples of how the user could set up their external wiring. Both common positive and negative leads might be a good choice for users that would like to limit the number of power supplies. When wiring connectors one should note that the pins are polarity sensitive. Thought miss wiring will not cause damage to the optical transistors, it will cause the signal to be off at all times. Table 1 gives the exact pin out description for the schematic depicted on the previous page.

Pin	Signal Name	Pin	Signal	Pin	Signal	Pin	Signal
1	Input 1001	2	1001 return	27	Output 0002+	28	Output 0002-
3	Input 1002	4	1002 return	29	Output 0003+	30	Output 0003-
5	Input 1003	6	1003 return	31	Output 0004+	32	Output 0004-
7	Input 1004	8	1004 return	33	Output 0005+	34	Output 0005-
9	Input 1005	10	1005 return	35	Output 0006+	36	Output 0006-
11	Input 1006	12	1006 return	37	Output 0007+	38	Output 0007-
13	Input 1007	14	1007 return	39	Output 0008+	40	Output 0008-
15	Input 1008	16	1008 return	41 ^a	Auxiliary E-Stop input+	42 ^a	External E-Stop input-
17	Input 1009	18	1009 return	43 ^a	Auxiliary E-Stop input -	44 ^a	External E-Stop input +
19	Input 1010	20	1010 return	45	Passive E-Stop output+	46	Passive E-Stop output-
21	Input 1011	22	1011 return	47	Not used	48	Not used
23	Input 1012	24	1012 return	49	Not used	50	Not used
25	Output 0001+	26	Output 0001-				
See next section for information on ordering a compatible third-party connector.							

Table 1

Pins 41- 46 are E-stop channels. The two E-stop input signal are monitored for an “on” to “off” transition at all times. If such a transition occurs the robot will shutdown in the normal “emergency stop” fashion. Also, when the controller encounters an E-stop command it will transmit an “on” signal through the E-stop output channel. The E-stop output signal is controlled by relays and is rated at 10 VA so caution should be taken when wiring it.

The Binary IO channels on the SIO have the following electrical specifications:

Operating voltage range	0 to 24 VDC
Operational current range, per channel	$I_{out} \leq 100 \text{ mA}$
V_{drop} across output in on condition	$V_{drop} \leq 0.85 \text{ V}$ at 100 mA $V_{drop} \leq 0.80 \text{ V}$ at 10 mA
Output off leakage current	$I_{out} \leq 600 \mu\text{A}$
Turn on response time (hardware)	3 μsec maximum
Software scan rate/response time	16 ms scan cycle/ 32 ms max response time
Turn off response time (hardware)	200 μsec maximum
Software scan rate/response time	16 ms scan cycle/ 32 ms max response time

SIO Output

Operational voltage range	0 to 24 VDC
"Off" state voltage range	0 to 3 VDC
"On" state voltage range	10 to 24 VDC
Typical threshold voltage	$V_{in} = 8$ VDC
Operational current range ^a	0 to 20 mA
"Off" state current range ^a	0 to 1.2 mA
"On" state current range ^a	7 to 20 mA
Typical threshold current, per channel ^a	10 mA
Impedance (V_{in}/I_{in})	1.3 K Ω minimum
Current at $V_{in} = +24$ VDC	$I_{in} \leq 20$ mA
Turn on response time (hardware)	5 μ sec maximum
Software scan rate/response time	16 ms scan cycle/ 32 ms max response time ^b
Turn off response time (hardware)	5 μ sec maximum
Software scan rate/response time	16 ms scan cycle/ 32 ms max response time ^b

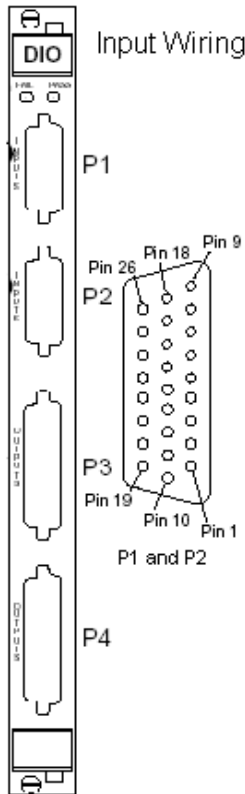
SIO Input

Table 2

There are a few things worth pointing out in the above specifications. First, the normal scan rate of signals by the software is 16 ms. This means if you are testing a object by moving the robot until a switch is activated, you should move the robot slowly to avoid smashing it into the object. In a case where one is trying to optimize this type of robot control, the first three signals should be used with the INT.EVENT command. This method results in a reduction in the response time to 2ms. A second point of interest is that there is a region of voltage on the input channels from 3V to 10V that is neither in the "on" or the "off" state. To avoid having a signal act erratically due to voltage fluctuations near this region, one should isolate the external hardware signals from module by some method (e.g. transistor, relay, ext.). The example at the end of the guide uses this technique.

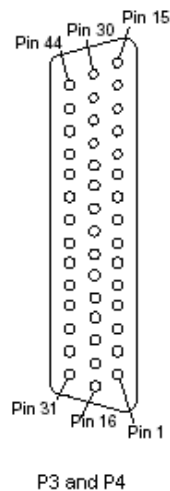
DIO Binary IO

The DIO module is designed exclusively for binary IO operation. It contains 32 input channels on two 26-pin, D-sub, female connectors and 32 output channels on two 44-pin, D-sub, female connectors. The Pin out and the wiring diagram can be found on the following two pages.



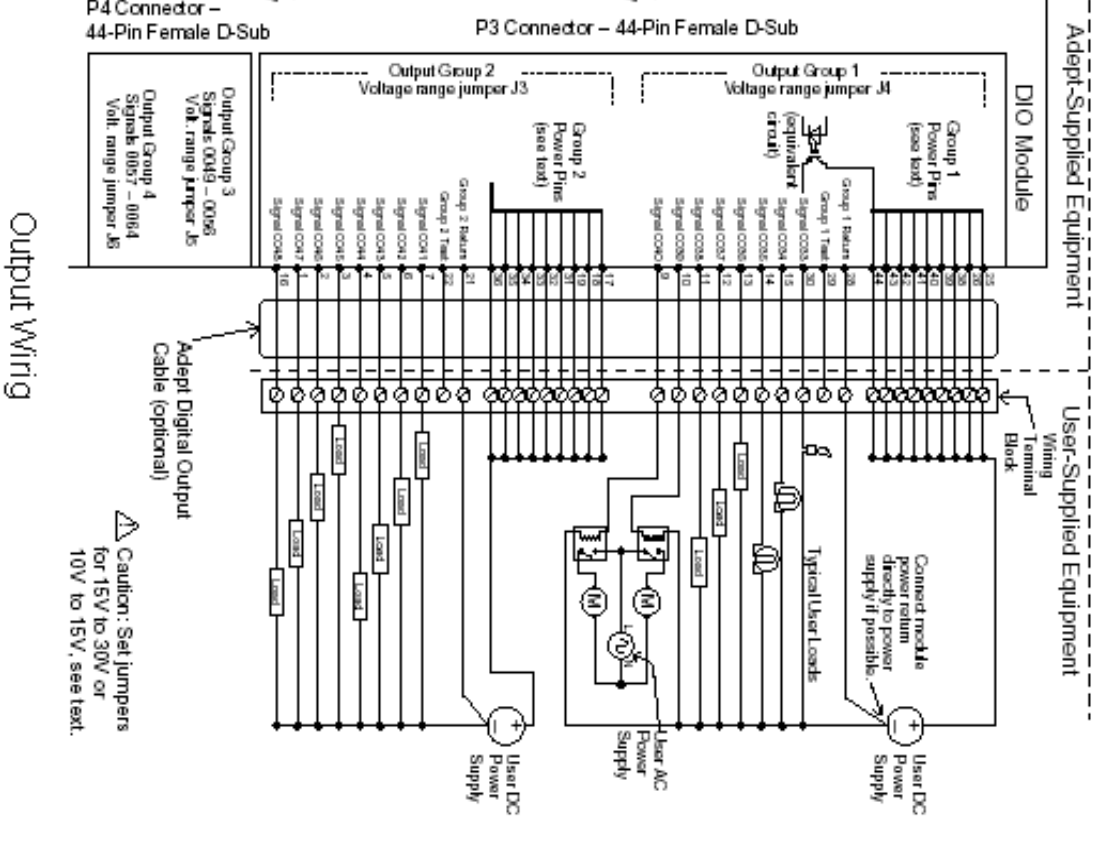
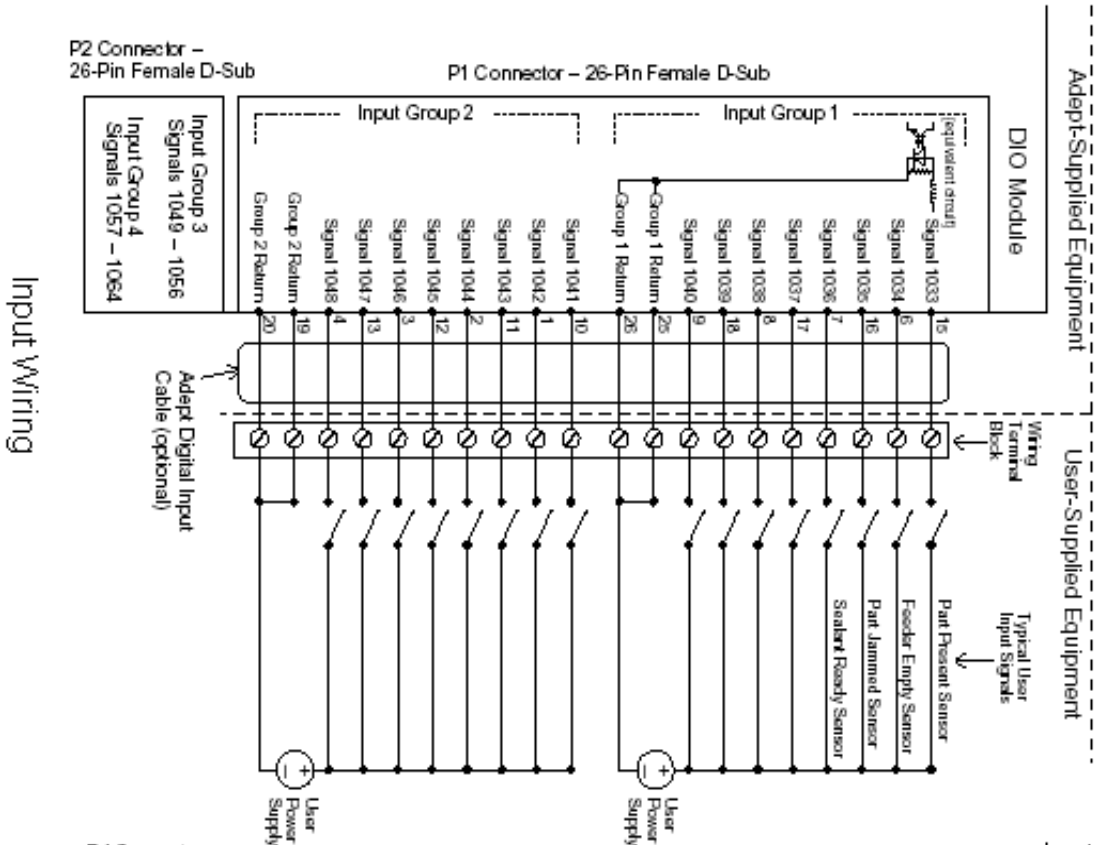
Pin Number	Signal Group	Module #1 Signal ^a	Pin Number	Signal Group	Module #1 Signal ^a
P1-15	1	1033	P2-15	3	1049
P1-6	1	1034	P2-6	3	1050
P1-16	1	1035	P2-16	3	1051
P1-7	1	1036	P2-7	3	1052
P1-17	1	1037	P2-17	3	1053
P1-8	1	1038	P2-8	3	1054
P1-18	1	1039	P2-18	3	1055
P1-9	1	1040	P2-9	3	1056
P1-25	1	group 1 return	P2-25	3	group 3 return
P1-26	1	group 1 return	P2-26	3	group 3 return
P1-10	2	1041	P2-10	4	1057
P1-1	2	1042	P2-1	4	1058
P1-11	2	1043	P2-11	4	1059
P1-2	2	1044	P2-2	4	1060
P1-12	2	1045	P2-12	4	1061
P1-3	2	1046	P2-3	4	1062
P1-13	2	1047	P2-13	4	1063
P1-4	2	1048	P2-4	4	1064
P1-19	2	group 2 return	P2-19	4	group 4 return
P1-20	2	group 2 return	P2-20	4	group 4 return

Output Wiring



Pin Number	Group Number	Module #1 Signal name ^a
P3-30	1	0033
P3-15	1	0034
P3-14	1	0035
P3-13	1	0036
P3-12	1	0037
P3-11	1	0038
P3-10	1	0039
P3-9	1	0040
P3-25	1	power
P3-26	1	power
P3-38	1	power
P3-39	1	power
P3-40	1	power
P3-41	1	power
P3-42	1	power
P3-43	1	power
P3-44	1	power
P3-28	1	group 1 return
P3-29	1	group 1 test
P3-7	2	0041
P3-6	2	0042
P3-5	2	0043
P3-4	2	0044
P3-3	2	0045
P3-2	2	0046
P3-1	2	0047
P3-16	2	0048
P3-17	2	power
P3-18	2	power
P3-19	2	power
P3-31	2	power
P3-32	2	power
P3-33	2	power
P3-34	2	power
P3-35	2	power
P3-36	2	power
P3-21	2	group 2 return
P3-22	2	group 2 test

Pin Number	Group Number	Module #1 Signal name ^a
P4-30	3	0049
P4-15	3	0050
P4-14	3	0051
P4-13	3	0052
P4-12	3	0053
P4-11	3	0054
P4-10	3	0055
P4-9	3	0056
P4-25	3	power
P4-26	3	power
P4-38	3	power
P4-39	3	power
P4-40	3	power
P4-41	3	power
P4-42	3	power
P4-43	3	power
P4-44	3	power
P4-28	3	group 3 return
P4-29	3	group 3 test
P4-7	4	0057
P4-6	4	0058
P4-5	4	0059
P4-4	4	0060
P4-3	4	0061
P4-2	4	0062
P4-1	4	0063
P4-16	4	0064
P4-17	4	power
P4-18	4	power
P4-19	4	power
P4-31	4	power
P4-32	4	power
P4-33	4	power
P4-34	4	power
P4-35	4	power
P4-36	4	power
P4-21	4	group 4 return
P4-22	4	group 4 test



It can be seen in the DIO wiring schematic that a couple of changes have been made in comparison to the SIO circuits. First, there are return groups instead of individual returns for each channel. In each case there are two return groups for each connector and two return channels for each group. The return channels are connected internally, so soldering to one is sufficient to ground any channel in the signal group. When creating connects it is probably a good idea to solder the return channels together with only one wire extending from them. Second, there are no special features for any specific channel. All channel scan at the same rate and none can be use with the INT.EVENT, REACT, or LATCH commands.

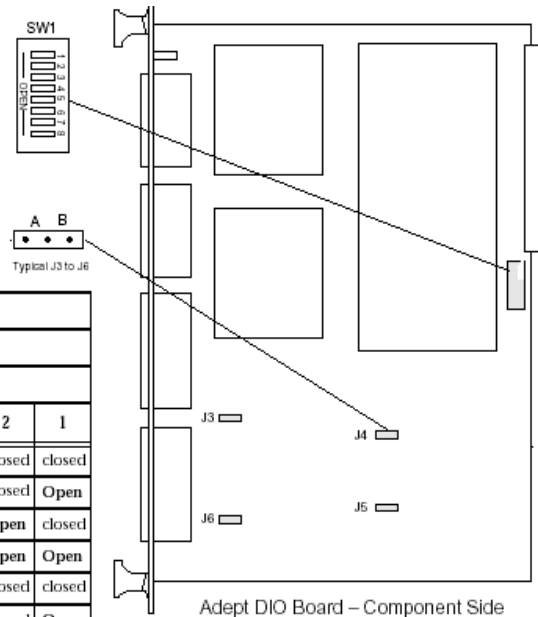
The DIO module allows the configuration of the voltage capacity on output channels. This means you must check the jumpers on the module to find what voltage is need across channel terminals before using the output channels. If the voltage is either higher or lower then the prescribed range the optical transistor will operate in a non-linear state and will dissipate excessive power. As a side note, the DIO module also allows the configuration of signal numbers via dipswitches. Information on how both of these features work can be found in Table 4.

Signal Group Number	Jumper Number	Voltage output in Position A	Voltage output in Position B
1	J4	15 to 30 V	10 to 15 V
2	J3	15 to 30 V	10 to 15 V
3	J5	15 to 30 V	10 to 15 V
4	J6	15 to 30 V	10 to 15 V

8-Position DIP Switch S1 on DIO PC board								
Switch positions 1 to 6 – to select bus address for DIO modules								
DIO Module Number	Input Signals	Output Signals	Switch position					
			6	5	4	3	2	1
1	1033 – 1064	33 – 64	closed	closed	closed	closed	closed	closed
2	1065 – 1096	65 – 96	closed	closed	closed	closed	closed	Open
3	1097 – 1128	97 – 128	closed	closed	closed	closed	Open	closed
4	1129 – 1160	129 – 160	closed	closed	closed	closed	Open	Open
5	1161 – 1192	161 – 192	closed	closed	closed	Open	closed	closed
6	1193 – 1224	193 – 224	closed	closed	closed	Open	closed	Open
7	1225 – 1256	225 – 256	closed	closed	closed	Open	Open	closed
8	1257 – 1288	257 – 288	closed	closed	closed	Open	Open	Open
Switch position 7 – required setting is closed								
Switch position 8 – required setting is open								

Table 4

The current configuration is set so that signal group 1 has a voltage rattooing of 10V to 15V and all other output groups have ratings of 15V to 30V. Also, the dipswitches have been set to a module number of one so the signals are 1033-1064 for inputs and 33-64 for outputs. It is recommended that these values not be changed because doing so often will significantly reduce the life of the module.



The circuit specifications for the DIO module are as follows:

Parameter	Value
Input	
Operational voltage range	$-35 \text{ VDC} \leq V_{in} \leq +35 \text{ VDC}$
"Off" state voltage range	$-35 \text{ VDC} \leq V_{in} \leq +3 \text{ VDC}$
"On" state voltage range	$+10 \text{ VDC} \leq V_{in} \leq +35 \text{ VDC}$
Typical threshold voltage	$V_{in} = +6 \text{ VDC}$
Operational current range	$0 \leq I_{in} \leq 14 \text{ mA}$
"Off" state current range	$0 \leq I_{in} \leq 400 \mu\text{A}$
"On" state current range	$2.3 \text{ mA} \leq I_{in} \leq 14 \text{ mA}$
Typical threshold current	$I_{in} = 1.1 \text{ mA}$
Impedance (V_{in}/I_{in})	$2.5 \text{ K } \Omega$ minimum
Current at $V_{in} = +35 \text{ VDC}$	$I_{in} \leq 14 \text{ mA}$
Current at $-35 \text{ VDC} \leq V_{in} \leq 0$	$I_{in} \leq -50 \mu\text{A}$
Turn on response time (hardware)	30 μsec maximum
Software scan rate/response time	16 ms scan cycle/ 32 ms max response time
Turn off response time (hardware)	175 μsec maximum
Software scan rate/response time	16 ms scan cycle/ 32 ms max response time
Output	
Low power supply range	$10 \text{ VDC} \leq V_{sup} \leq 15 \text{ VDC}$
High power supply range	$15 \text{ VDC} \leq V_{sup} \leq 30 \text{ VDC}$
Power supply ground current	$I_g \leq 35 \text{ mA}$
Operational current range, per channel	$I_{out} \leq 400 \text{ mA}$
V_{out} , output on, $I_{out} = 400 \text{ mA}$	$V_{sup} - 0.8 \leq V_{out} \leq V_{sup}$
Output off leakage current	$I_{out} \leq 400 \mu\text{A}$
Turn on response time	10 μsec maximum
Turn off response time	120 μsec maximum

Table 5

Note that the specifications for the DIO are quite different from the SIO's. The DIO generally has greater power capacity than the SIO, as a result there may be situations where one might want to use the DIO even though there are channels open on the SIO.

Commands

`BITS <first_sig>, < num_sigs> = <value>`

A way to change groups of output signals to binary format. Possible application might be outputting numerical values to external hardware(RF).

<first_sig> - Real-valued expression defining the lowest-numbered signal to be affected.

<num_sigs> - Optional real-valued expression specifying the number of signals to be affected. A value of 1 is assumed if none is specified. The maximum valid value is 8.

<value> - Real-valued expression defining the value to be set on the specified signals. If the binary representation of the value has more bits than < num_sigs,> only the lowest <num_sigs> signals will be affected.

This command will not work with any signal but output signals, also, if the <value> is decimal it will be rounded to the nearest integer.

Examples:

BITS 1,4 = BCD(2) Output signals 1-4 = 1,1,0,0

BITS 9,8 = SPEED(1) Output signals 9-16 = (current monitor speed in binary)

BITS 1,8 = 255 Output signals 1-8 = 1,1,1,1,1,1,1,1

ERROR (<task_num> , <select>)

Returns the error number of a recent error that caused program execution to stop or caused a REACTE(RF).

<task_num > - Real value, variable, or expression (interpreted as an integer) whose value selects the source of the error code as follows:

-1 most recent error from the program in which the ERROR function is executed.

0 Return the number of the most recent error from the program executing as task #0.

>0 Return the number of the most recent error from the program executing as the corresponding task number.

<Select> - Optional real value, variable, or expression (interpreted as an integer) that selects the error information to be returned.

0 Return the error number of the most recent program execution error (excluding I/O errors—see IOSTAT) for the specified program task.

1 If the most recent error (for the specified program task) had an error code less than or equal to -1000 (that is, -1000, -1001, etc.), return the variable portion of the corresponding error message.

The value returned should be interpreted as a 6-bit numeric value—see below for details. Zero is returned if the error did not have a variable portion in its message. (Also see select = 3 below.)

2 Return the error number of the most recent error from an MCS instruction executed by the specified program task.

3 Return the number of the robot associated with the most recent

error for the specified program task. Zero is returned if the error was not associated with a specific robot. (Also see select = 1.)
4 Return the error code corresponding to any pending RSC errors for the robot selected by the specified user task.

Exsample:
IF I== 1 THEN
 <some_action>
ELSE
 ERROR(<n>,<m>)
END

IGNORE <signal>

Disables continuous monitoring of the specified signal, canceling the effect of the last REACT or REACTI for this signal. (PI).

<Signal> Digital input signal number in the range 1001 to 1012, or an internal signal in the range 2001 to 2008, or 0. Negative to monitor a signal turning off.

IGNORE only work with the REACT command, thus is limited to the SIO module.

Example:

```
REACT -1001,<error_program>  
WHILE <condition> DO  
  <group_of_steps>  
END  
IGNORE -1001  
  <continued_program>
```

INT.EVENT <source> , <level>

Allows the robot to sense an event like when a specified digital I/O sign transitions to positive.

<source> - digital I/O signal in the range 1001 – 1003 or the value 0, which cancels monitoring.

<level> - Optional argument specifying the VME interrupt request level to be used. Acceptable values are 1 and 2.

Example:

```
INT.EVENT 1003 ;Initiate monitoring of 1003  
WAIT.EVENT ;Wait until interrupt occurs  
INT.EVENT 0 ;Cancel monitoring of interrupts
```

IO <signal_group>

Display the current states of external digital input/output signals and/or internal software signals.(MC)

<signal_group> - Optional integer value that, if specified, selects which digital signals are to be displayed. 0 Digital output signals, 1 Digital input signals, 2 System software signals, 3 The 3000 series of digital output signals.

Example:

```
IO
0032-0001 ---- ---- ---- ---- ---- 0000 0110
0064-0033 0000 0000 0000 0000 0100 0000 0000 0000
```

This display indicates that signals 2, 3, and 47 are on; all others are off or not installed.

LOCK <priority>

Sets a priority value such that all REACT command with priority greater then that value are ignored(PI).

<priority> Real-valued expression with a value from 0 to 127, which will become the new reaction lock-out priority.

Example:

```
REACT 1001,<error_program>,100
REACT 1002,<error_program>,15
REACT 1003,<error_program>,50
```

LOCK 30 ; disables line 2

LATCH (<select>)

Return a transformation value representing the location of the robot at the occurrence of the last external trigger(PI).

<select> - 0 Robot postion latch (default),1 AdeptForce guarded-mode trigger

Operation of the external trigger can be configured with the Adept controller configuration program (in the file CONFIG_C.V2 on the Utility Disk). This trigger may originate from the vision processor or an external source such as binary IO.

PRIORITY

Return the current reaction lock-out priority for the program. One possible use is the temporary removal of a Lock value(RF).

Exsample:

```
save = PRIORITY      ;Save the current priority
IF save < 10 THEN    ;Raise priority to at least 10
LOCK 10
END                  ; Access data shared by a reaction routine.
LOCK save            ;Set priority to original value
```

REACT[I] <signal_num>, <program> , <priority>

Initiate continuous monitoring of a specified digital signal and automatically trigger a subroutine call if the signal properly transitions(PI).

<signal_num> - Real-valued expression representing the signal to be monitored. The signal number must be in the range 1001 to 1012 (external input signals) or 2001 to 2008 (internal software signals). A negative signal means the system is looking for a transition to the off state.

[I] - Addition of an I on react commands will force the execution of the subroutine at the signal change event, otherwise the react command will wait to the end of the present action.

<Program> - Name of the subroutine that is to be called when the signal transitions properly.

<priority> -(the default is one) Optional real-valued expression that indicates the relative importance of this reaction. The reaction with the highest priority will be run first. Once the subroutine has finished the main program will be continued.

To use the react commands the input signal must remain stable for at least 18 milliseconds to assure detection.

The signal monitoring continues until one of the following occurs:

- An IGNORE instruction is executed for the signal.
- A reaction occurs (in which case IGNORE signal_number is automatically performed).
- A REACT (or REACTI) instruction is executed that refers to the same signal. That is, if the signal specified in a REACT instruction is already being monitored by a previous REACT or REACTI instruction, the old instruction will be canceled when the new REACT instruction is executed.

Example:
REACT 1001,<subroutine_A>,10
REACT 1002,<subroutine_B>,9
<main_program>

If both 1001 and 1002 change at the same time then subroutine_A will run first then subroutine_B, and finally the main_program will continue.

REACTE <program_name>

This command facilitates complete shutdown of external hardware in the event of an error. (PI)

<program_name> - Name of the program that is to be called when a program error occurs. This program should include shutdown commands for external devices and error messages.

After REACTE subroutine has run the main program execution will not continue. This command has the highest priority and will execute immediately before all else. See ERROR for more.

Example:
REACTE error
REACT 1002,<subroutine_B>,9
<main_program>
END

;If an error occurs the program will jump to the error subroutine and end.

RESET

Turn off all the external output signals. (MC)

RETURNE

Terminate execution of an error reaction subroutine and resume execution of the last-suspended program at the step following the instruction that caused the subroutine to be invoked. Would be use with REACTE if one wanted to continue the main program after the error subroutine was finished. (PI)

RUNSIG <signal_num>

Turns an output signal “no” (or “off” if negative is used) during the execution of a program. Possible application might be remote power control for external systems so that some components only turn on when a program or task is running.(PI)

<signal_num> - Real-valued expression that specifies one of the digital

output signals (or an internal software signal) that is to be controlled.

A negative value will result in the signal being set to off during program execution, and turned on when execution stops.

Example:

```
RUNSIG run.signal
```

(Turns on the digital signal identified by the value of the variable run.signal)

SIG (<signal_num>, ..., <signal_num>)

Returns the logical AND of the states of the indicated digital signals(RF).

<signal_num> - Real-valued expression that evaluates to a digital I/O or internal signal number. A negative value indicates negative logic for that signal.

Examples:

```
SIG(1001) ;Returns -1.0 (TRUE)
```

```
SIG(1004) ;Returns 0.0 (FALSE)
```

```
SIG(-1004) ;Returns -1.0 (TRUE)
```

```
SIG(1001,1004) ;Returns 0.0 (FALSE)
```

```
SIG(1001,-1004) ;Returns -1.0 (TRUE)
```

SIG.INS (<signal_num>)

This function can be used to make sure the digital I/O signals are installed as expected by the application program. (RF)

<signal_num> - Real-valued expression that defines the number of the digital I/O or software signal to check. (The absolute value is used, so negative signal numbers are allowed.)

This function returns TRUE (-1) if the specified digital I/O or software signal is available for use by the system. Otherwise, FALSE (0.0) is returned.

Example:

```
IF NOT SIG.INS(1002) THEN
```

```
<error routine>
```

```
END
```

SIGNAL <signal_num>, ..., <signal_num>

Turns on or off external digital output signals or internal software signals(PI).

<signal_num> Real-valued expression that evaluates to a digital output or internal signal number. A positive value indicates turn on; a negative value indicates turn off.

Only digital output signals (numbered from 1 to 8 and 33 to 512) and internal (software) signals (numbered from 2001 to 2512) can be specified.

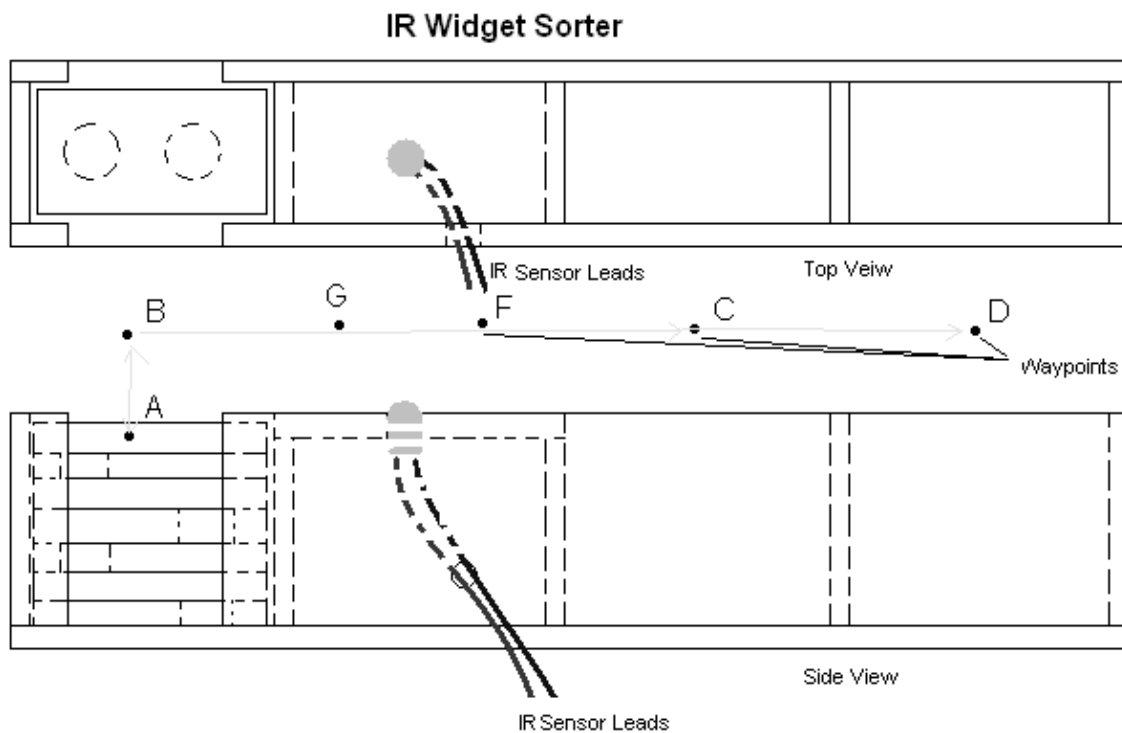
Example:

SIGNAL -4 ;turn off signal 4

SIGNAL 2010 ;turn on signal 4

Example of a Complete System:

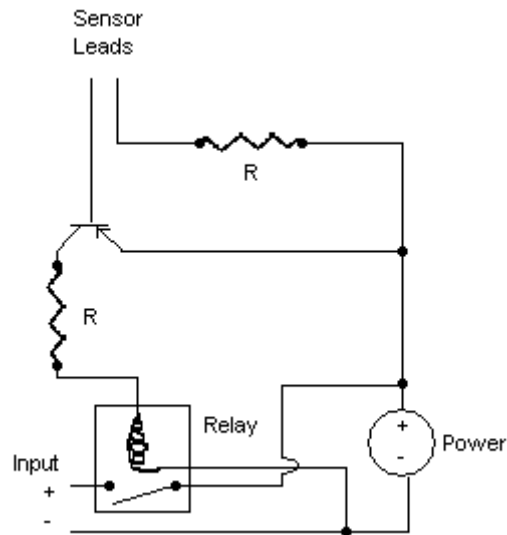
Hardware:



Explanation:

The robot picks up widget from the pile at A then move along the way points. During that motion the widget is scanned over the IR sensors, which tells the robot to drop the widget at point C or D depending on if it find a signal at point G or F. (Note: an IR source is placed above the widget so that it shines through holes if the widget has any. If it does have holes the widget is good otherwise it is bad.)

Electrical Circuit:



Possible Program:

```
.PROGRAM main()
  SPEED 100 ALWAYS
  x = 0
  ; Move to start Position
  MOVES b
  BREAK
  ;initialize good and bad counters
  good = 0
  bad = 0
  ; Start 10 widget sorting loop
  FOR i = 0 TO 9
    ; picks up widget from stack and moves to check position 1
    y = 0
    x = i*5.7-1
    APPRO a, -x
    CLOSEI
    BREAK
    SPEED 25
    MOVES b
    BREAK
    MOVES g
    BREAK
    ; checks for hole 1, dumps if there is a hole, counts
    IF SIG(1033) THEN
      MOVES d
      OPENI
      BREAK
      bad = bad+1
    ; Checks for hole 2, dumps if there is hole, counts
```

```
ELSE
    MOVES f
    BREAK
    IF SIG(1033) THEN
        MOVES d
        OPENI
        BREAK
        good = good+1
; Dump in good bin if no holes, counts
    ELSE
        MOVES c
        OPENI
        BREAK
        bad = bad+1
    END
END
MOVES b
END
; Displays number of good and ba widgets
TYPE "There are", good, " good widget(s) and", bad, " bad widget(s)."
MOVES e
TYPE " You want to do another stack?"
PROMPT "Y/N?", q
IF q == "Y" THEN
    GOTO 2
END
```