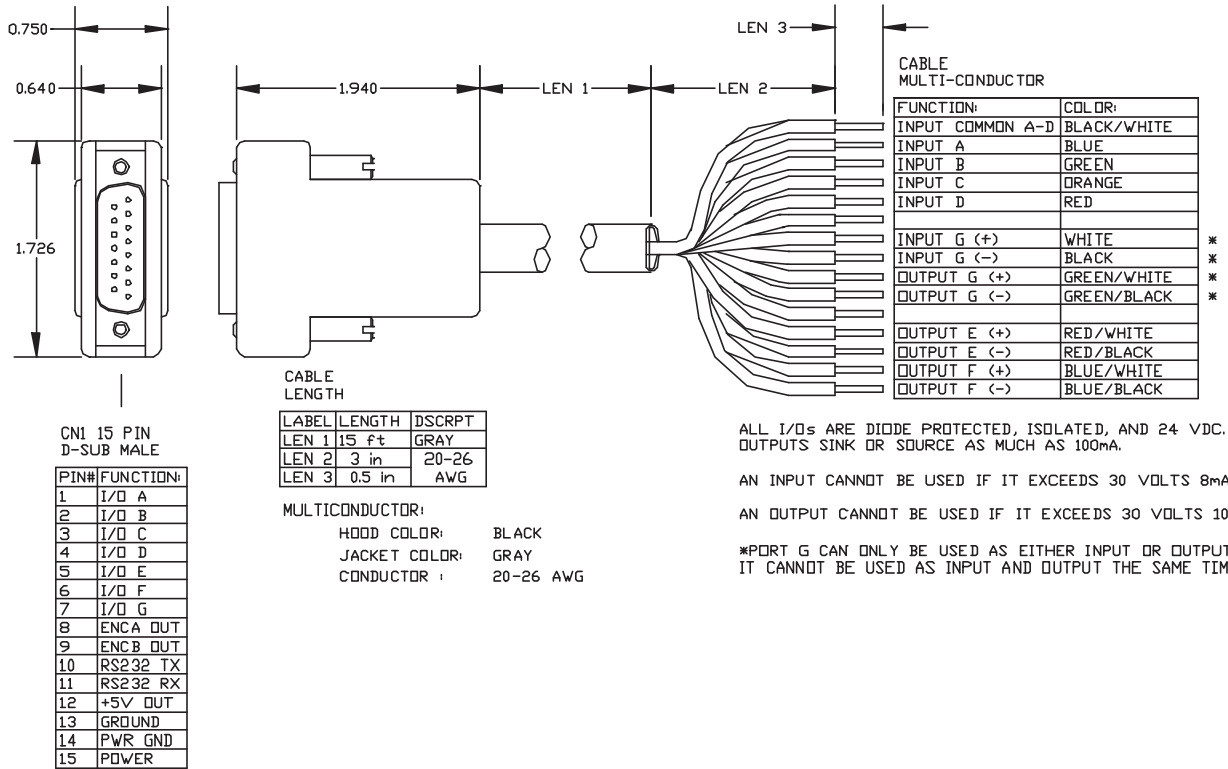


# iSeries Cable Manual for 71-020911-15

The 71-020911-15 is a cable with a DB15 connector that converts 5VDC iSeries I/O to 24VDC I/O. The user has the option of using the 71-020911-15 cable with four digital inputs and three digital outputs or five digital inputs and two outputs. This cable connects directly into the iSeries DB 15 I/O connector



Inputs A to D can be set to either all sourcing or sinking inputs. Port G input is independent from input A to D. Outputs E, F, and G are wired independently so they can either be sourcing or sinking.

### RATING:

#### Input

min. voltage	24 VDC
max. voltage	30 VDC
min. current	5 mA
max. current	8 mA

#### Output

max. voltage	30 Vdc
max. current	100 mA

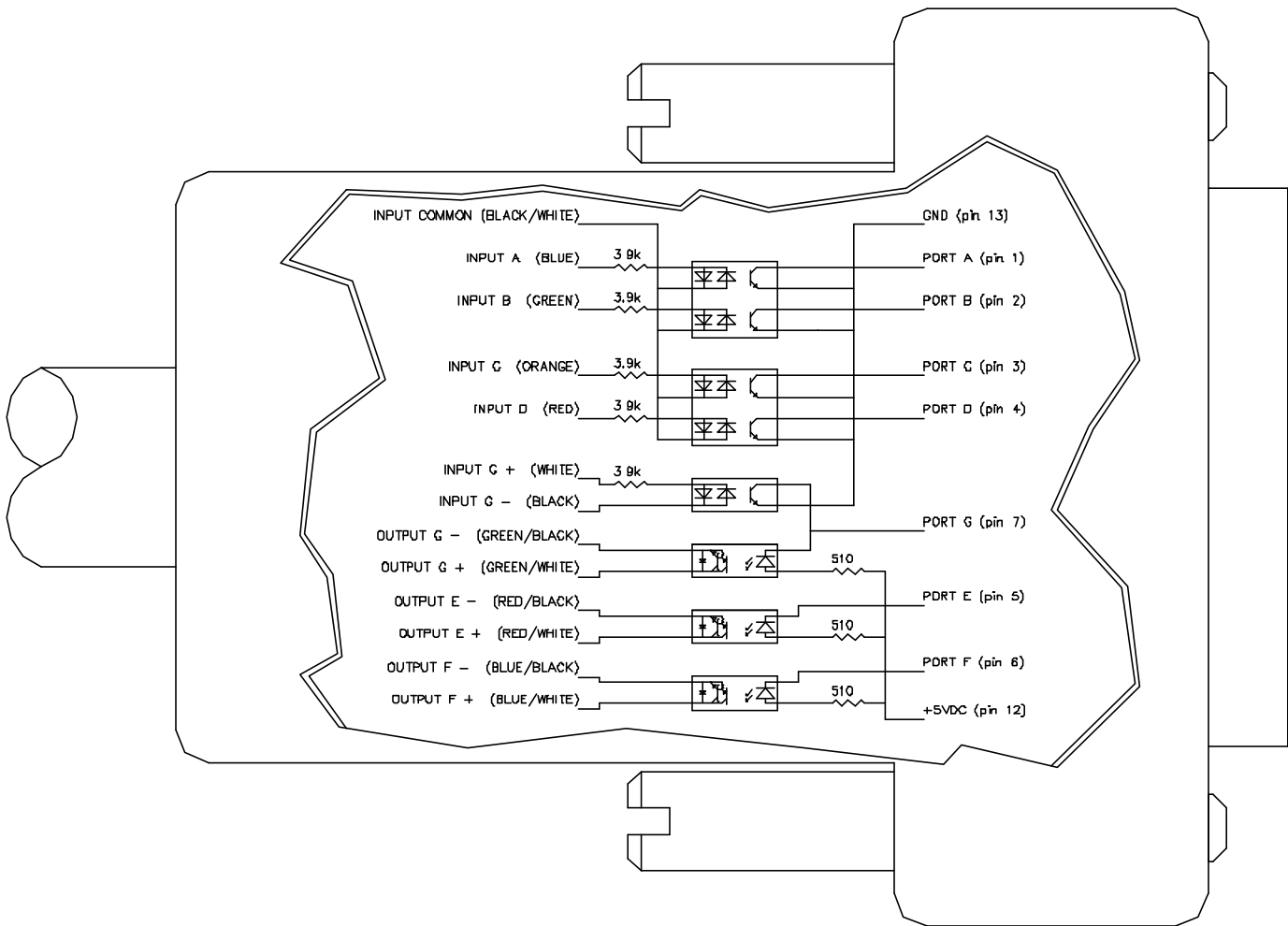
Damage may occur if these maximum ratings are exceeded.

## iSeries Interface

The 71-020911-15 cable uses the iSeries I/O pins as listed:

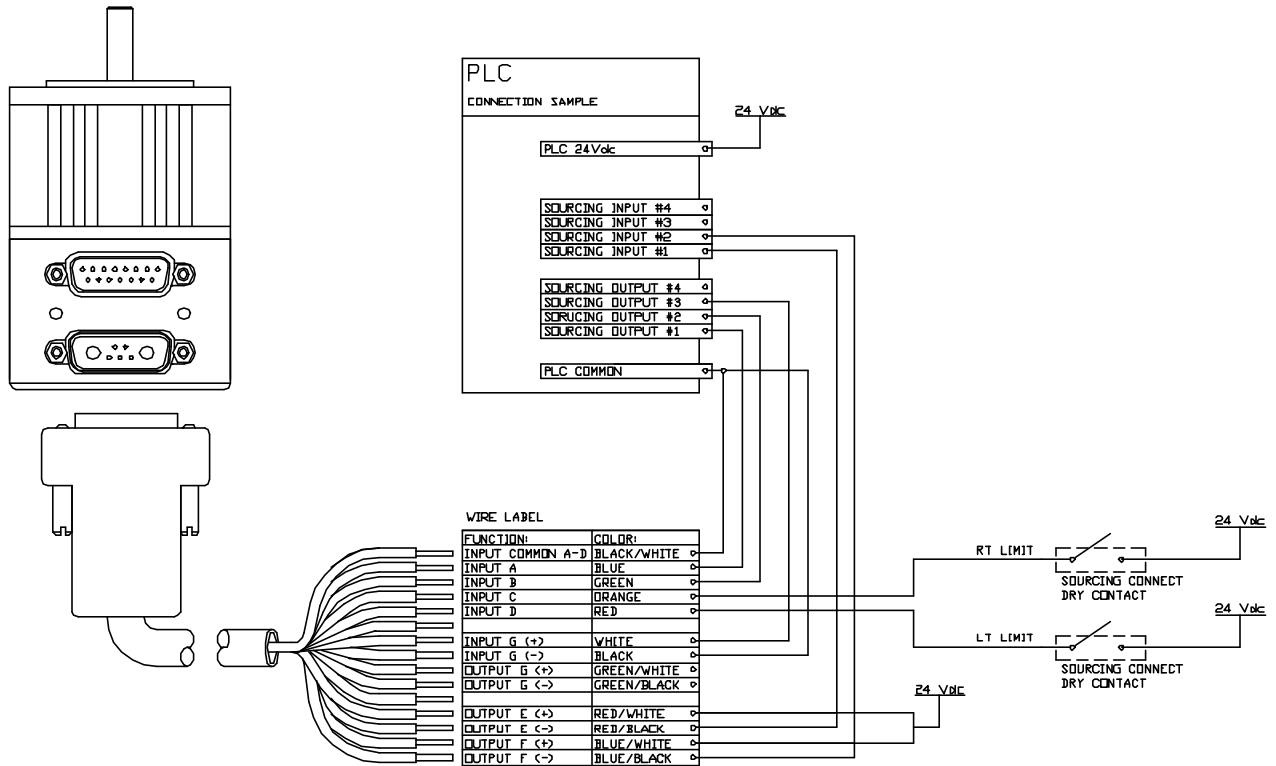
PIN	SIGNAL	DESCRIPTION
1	input A	Digital input A / Encoder input A / Step input (input frequency 50kHz)
2	input B	Digital input B / Encoder input B / direction input (input frequency 50kHz)
3	input C	Digital input C / Positive Limit
4	input D	Digital input D / Negative Limit
5	output E	Digital output E
6	output F	Digital output F
7	input/output G	Digital input G / Digital output G (can only use either input or output)
12	+5Vdc	+5Vdc output
13	GND	Signal Ground

## Schematic



## Application

The 71-020911-15 cable is used to interface the iSeries motor with a PLC and some coil relays. A sample application wiring is shown below:



The above diagram is an example of using the 71-020911-15 cable to interface the iSeries motor with a PLC. The sample program below reads output signal from the PLC to determine which predefined profile to run. After a move completed, the iSeries motor will signal back to the PLC. Also, the iSeries motor will send outputs to the motor if an error occurred.

```
'SAMPLE PROGRAM USING I/O
'INPUT A to C for PLC profile selection
'INPUT D for read ready
'OUTPUT G motor signal PLC after motion completed
'OUTPUT E and F to turn on/off pump and valve
,
'initialize I/O ports
UAI      'initialize port A as input, signal input bit 0
UBI      'initialize port B as input, signal input bit 1
UCP      'initialize port C as RT Limit input
UDM      'initialize port D as LT Limit input
UE=1    'set output E off
UEO      'initialize port E as output, trajectory start(high)/ended(low)
UF=1    'set output F off
UFO      'initialize port F as output, fault(high)
UGI      'initialize port G as input, read ready trigger

'set Acceleration/velocity
PM       'set motor to PositionMode
A=32*100 'set acceleration to 100 rps^2
V=128848*30 'set velocity to 30 rps
```

```

'read/check input loop
WHILE 1
    WHILE UGI==1 LOOP
        UF=1
        ab[0]=UAI
        ab[1]=UBI*2
        a=ab[0]+ab[1]
        SWITCH a
            CASE 0
                PRINT("CASE 0 move to P=8000",#13)
                P=8000
                GOSUB0
            BREAK
            CASE 1
                PRINT("CASE 1 move to P=10000",#13)
                P=10000
                GOSUB0
            BREAK
            CASE 2
                PRINT("CASE 2 move to P=-8000",#13)
                P=-8000
                GOSUB0
            BREAK
            CASE 3
                PRINT("CASE 3 move to P=-10000",#13)
                P=-10000
                GOSUB0
            BREAK
        ENDS
    LOOP
END

L0
UE=0
G
TWAIT
UE=1
IF Fe
    PRINT("excessive position error occured",#13)
    UF=0
ENDIF
IF Fp
    PRINT("RT Limit reached",#13)
    UF=0
ENDIF
IF Fm
    PRINT("LT Limit reached",#13)
    UF=0
ENDIF
IF Fh
    PRINT("Over Temperature Occured",#13)
    UF=0
ENDIF
RETURN

```

```

'infinite WHILE LOOP
'gate, waiting for PLC read ready signal
'reset the fault output if any
'if input A triggered, UAI will read 0,
' otherwise ab[0] is 1
'if input B triggered, UBI will read 0,
' otherwise ab[1] is 2
'summing up the binary values
'comparing each binary value with the
' SWITCH/CASE
' statement
'CASE 0 when B A triggered ( 0 0 )
'GO to SUBroutine C0 to start motion and
' error handling
'BREAK out of SWITCH statement
'CASE 1 when B _ triggered ( 0 1 )
'GO to SUBroutine C0 to start motion and
' error handling
'CASE 2 when _ A triggered ( 1 0 )
'GO to SUBroutine C0 to start motion and
' error handling
'CASE 3 when _ _ triggered ( 0 0 )
'GO to SUBroutine C0 to start motion and
' error handling
'ENDS for closing SWITCH statement
'LOOP for closing WHILE statement

'END marks end of program

'Label for subroutine C0
'output high, trajectory started
'start trajectory (motion)
'wait until trajectory ends (motion stopped)
'reset signal to RESET MOTION (low) signal
to PLC
'checking excessive position error bit
'print to terminal window
'set fault signal (high)

'checking RT limit bit
'print to terminal window
'set fault signal (high)

'checking LT limit bit
'print to terminal window
'set fault signal (high)

'checking over temperature bit
'print to terminal window
'set fault signal (high)

'RETURN to main program

```

You probably noticed that the motor is reading the signal low when high signal is being sent to the 71-020911-15 cable. If you prefer the motor to read a high signal when high signal is being sent to the cable, you can mask the input value by using the following command:

```
a=UAI==0          `this sets a to 1 if UAI is true, which in this case if UAI is 0
                  `      (low)
```

### **I/O Commands:**

```
UAI      `initialize port A as input
UBI      `initialize port B as input
UCI      `initialize port C as input
UDI      `initialize port D as input
UCP      `initialize port C as Right Limit (Port C is right limit by default)
UDM      `initialize port D as Left Limit (Port D is left limit by default)
UGI      `initialize port G as input (port G can only be used as either
UGO      `initialize port G as output  input or output)
UG=1     `set output G off (output line open)
UG=1     `set output G on (output line close)
UEO      `initialize port E as output
UE=1     `set output E off (output line open)
UE=0     `set output E on (output line close)
UFO      `initialize port F as output
UF=1     `set output F off (output line open)
UF=0     `set output F on (output line close)

d=UCI    `store the input state value of port C into variable d

IF UAI   `using with IF statement, true => UAI is 1
ENDIF

IF UAI==0 `using with IF statement, true => UAI is 0
ENDIF
```

