Table of Contents

Manual Revisions

If you contact us in reference to this manual, be sure to include the revision number.

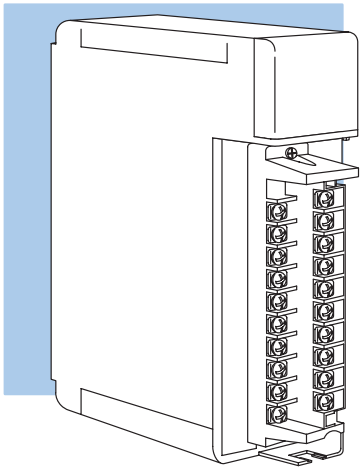
Title: DL405 Interrupt Input Module
Manual Number: D4–INTR–M

Issue	Date	Effective Pages	Description of Changes
Original	8/94	Cover/Copyright Contents Manual History 1 — 24	Original Issue
Rev. A	6/98	Entire Manual Manual Revisions	Downsize to spiral Rev. A

Introduction

Overview of the module's operation

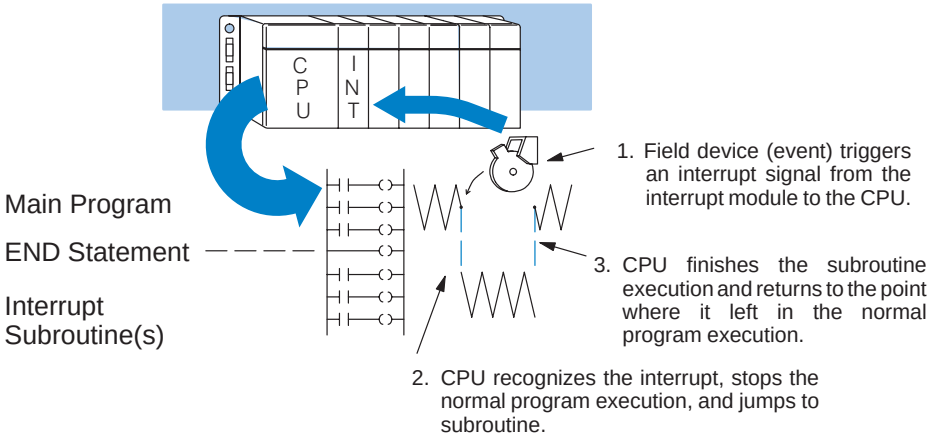
The DL405 Interrupt Input module (D4–INT) is an 8-point isolated interrupt input module for use with the DL405 family of products. The module consumes 16 X input points and must be installed in Slot 0, next to the CPU. You can use two Interrupt modules, installed in Slots 0 and 1, if you are using a D4–440 CPU. (More on this later.)



The Interrupt module is intended for applications that have one or more high-priority events which require immediate attention. That is, these are events that cannot wait until the CPU has finished it's normal program execution.

When this high priority event occurs, the interrupt module sends an interrupt signal to the CPU. Once the interrupt is received, the CPU immediately suspends its routine scan cycle and jumps to a subroutine that corresponds to the particular interrupt input signal.

The following diagram shows a brief overview of this concept.



Module Features

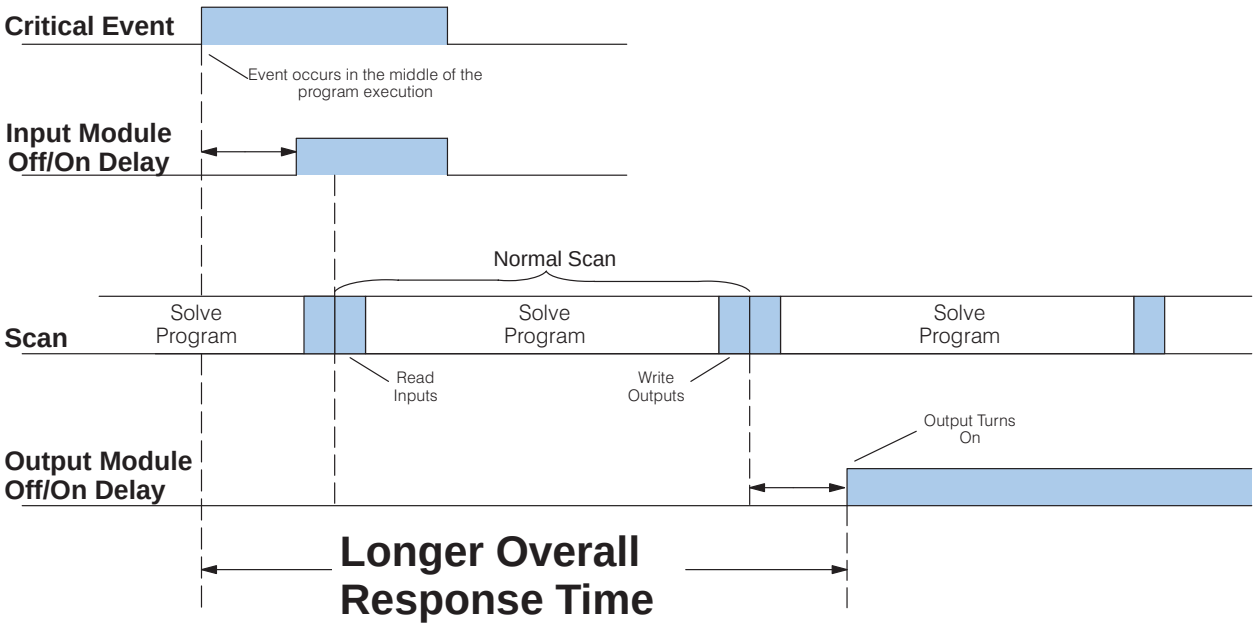
This module has some great features that make interrupt processing easier.

- Interrupts are prioritized (lowest number takes priority)
- Rising or Falling edge trigger
- Selectable response delay (allows you to easily ignore false signals)
- Module parameters are easily selected by setting dipswitches on the back of the module
- No complex programming, only simple ladder instructions are required
- If you don't use all 8 points as interrupts, you can use the non-interrupt points as regular discrete inputs.

Interrupt Signal Processing

When do you need an interrupt?

In normal circumstances, the CPU reads the status of input points, solves the relay logic program, and then updates all of the output points. This process is called the scan and usually takes place in a matter of milliseconds. The following diagram shows how this works.



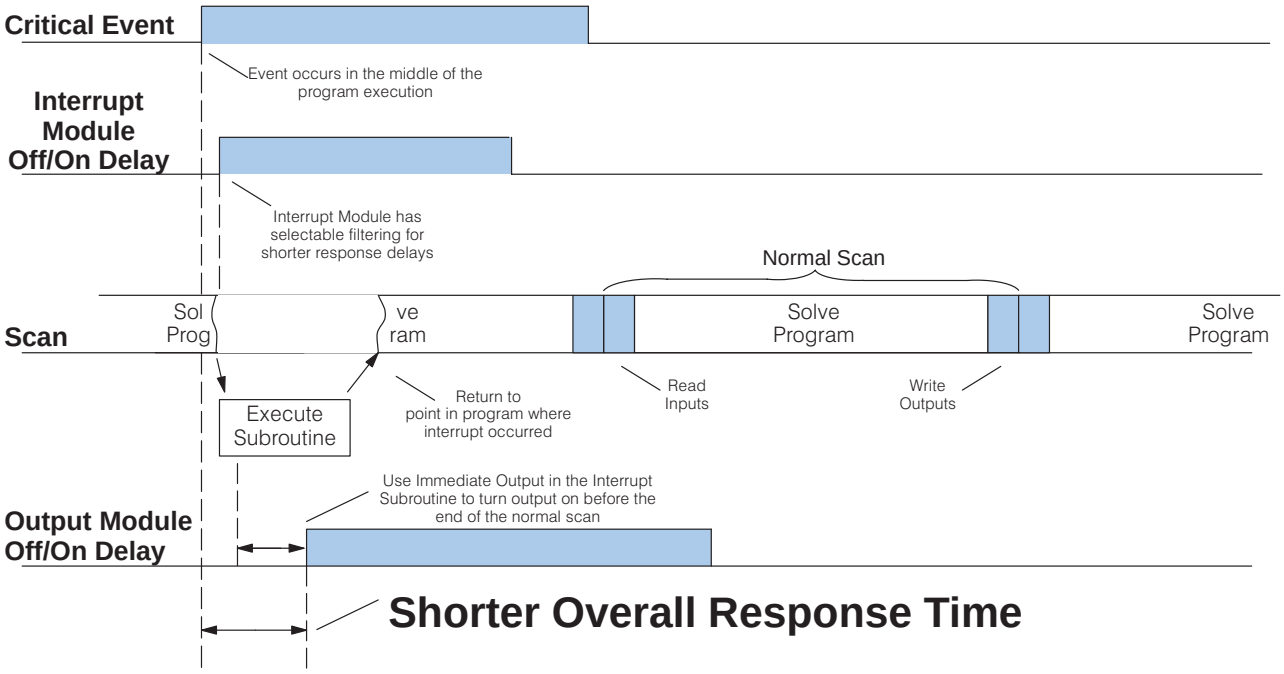
As you can see from the diagram, there can be a delay between the actual occurrence of the event and the CPU processing required to respond to the event. In most applications, the scan happens dozens of times in a second, so the delay is not critical.

However, in some applications you can have events that require an immediate response. For example, your application may have a critical event that requires an action to be initiated within 10 milliseconds of when the event occurs. Even with a very fast scan time, it may take 10 to 30 milliseconds to completely scan the program before it can update any outputs. So, you can see that you need some method for interrupting the normal scan to react to these critical events.

How does an interrupt solve response problems?

What is needed is a mechanism for *interrupting* the normal CPU scan cycle when something needs to be done. The CPU can then take care of what has to be done, and go back to its normal business. Here is the same diagram, but it has been changed to show how the CPU reacts if the Critical Field Event is being monitored by an Interrupt Input Module. There are several things to notice.

- There is very little Off/On delay (the Interrupt Module has selectable response, or filtering).
- When an interrupt is received, two things happen.
  1. The CPU *finishes executing the current program rung*.
  2. The CPU then suspends the normal program execution and jumps to a subroutine that corresponds to the interrupt point that became active.
- By using Immediate Instructions in the Subroutine, you can quickly read input points and write output points.
- The CPU resumes the normal program execution on the rung *following* the rung that was being processed when the interrupt occurred.

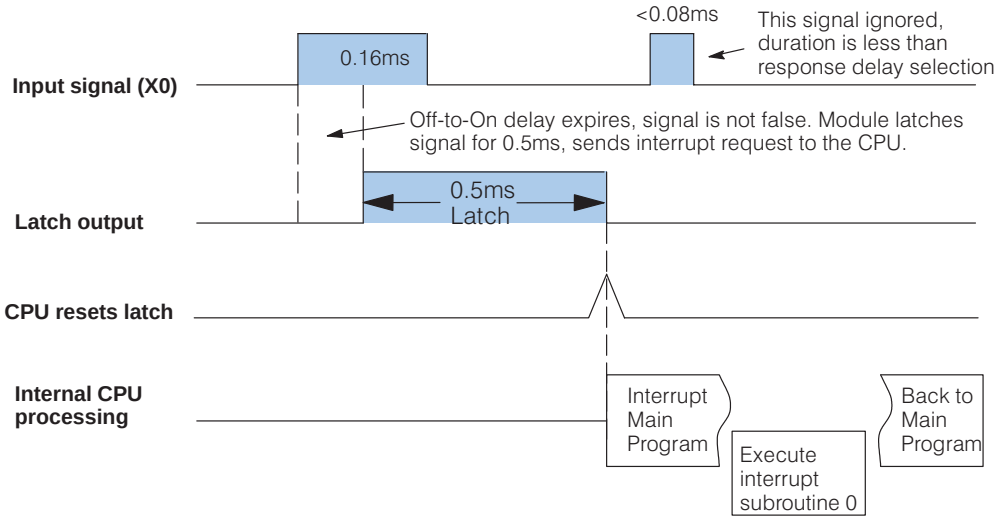


How does the CPU process the interrupts?

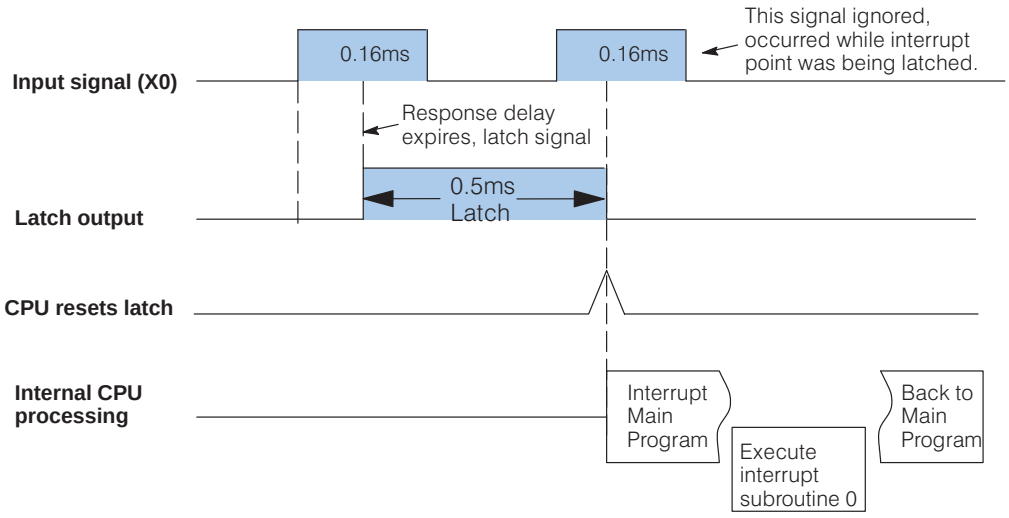
It may help you to understand exactly how the CPU works together with the Interrupt module to process the interrupt signals.

Single Interrupt Signal: When the interrupt input module senses an input signal, it has an extremely short delay before it signals an interrupt. This *Response Delay* (also called filtering) can be selected via a dipswitch setting and helps reduce the possibility of false interrupt signals. (The Response Delay ranges are discussed in more detail later in this manual.)

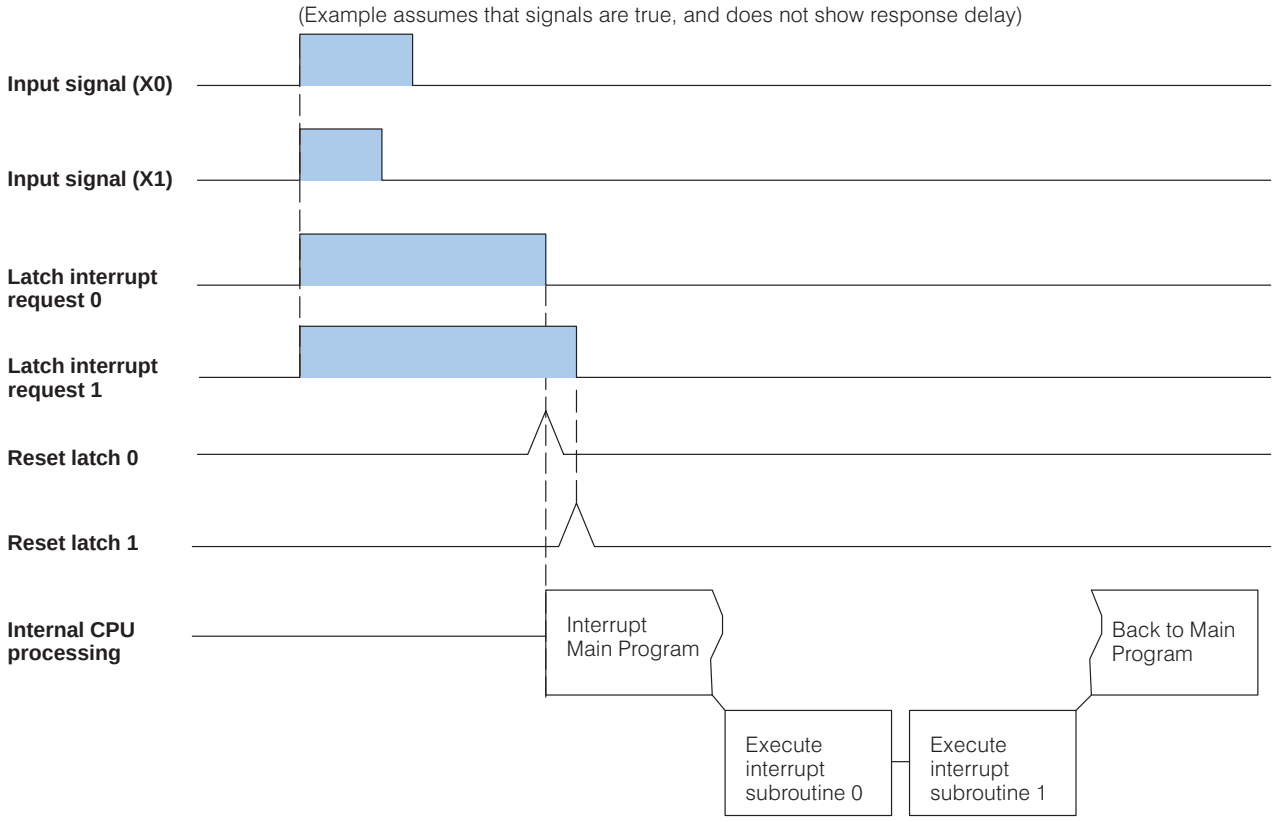
Once the delay period has passed, the module will latch an internal signal and send an interrupt request to the CPU. Once the CPU receives the request, the CPU stops its normal scan to jump into the interrupt routine; and at the same time, sends a reset signal back to the module. The following diagram shows this operation.



Multiple Signals from the Same Interrupt Input: There may be occasions where several interrupt signals are received very close together in time. In the diagram below, the time line reads from left to right. The first signal is latched. It takes 0.5 milliseconds to complete the latching process. If within that time span another signal is sent, the module will ignore the second signal and only recognize the first.



Simultaneous Signals from *Different* Interrupt Inputs: What happens if the module senses signals simultaneously from several of the interrupt input points? In this case, there is a priority. The highest priority is given to X0, then comes X1, then X2 and so forth down to X7. The diagram below shows two signals being received simultaneously.



Special Considerations in CPU processing

In each of the previous examples, you noticed that the CPU interrupted the main program execution. The CPU does not just drop everything that it is doing. Instead, the CPU performs an orderly transition to the subroutine by *finishing the current program rung before it jumps to the subroutine.* This typically takes at least 1ms, but the actual response depends on the instructions on the rung. For example, if the CPU receives the interrupt signal on a rung that contains several math instructions, then the response will be slower than if the rung only contained a simple output coil. There's not much you can do about this, but just be aware of it.

Also, since the CPU only executes the logic in the interrupt subroutine, it does not handle any communication requests from external devices. In the vast majority of cases this is not a problem. However, some operator interfaces can have a "time out" type of error if they issue a request and do not get a response from the CPU in a certain amount of time. If you are using an Operator Interface and this happens, try reducing the communications baud rate. The slower speeds usually allow for a longer response time from the CPU.

Do I have to use an Interrupt Module?

You don't always have to use an Interrupt Input module to generate an interrupt. Basically, an interrupt can be initiated two different ways:

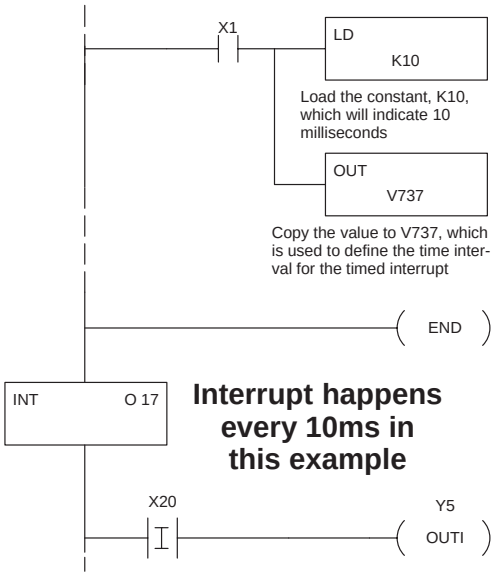
- You can use a timed interrupt (an interrupt on a defined time interval, D4–440 CPU only)
- Or, you can use the Interrupt Input module (interrupt triggered by a limit switch closing, proximity switch, etc. via the interrupt module)

NOTE: These examples **DO NOT** show all of the instructions necessary to correctly program an interrupt routine. This is discussed later in this manual.

Timed Interrupts: The D4–440 offers a single timed interrupt *instruction*. You can specify how often the interrupt occurs, from 3–1000 ms. This is great for those applications where you know you need to take some action every so often. However, it may also slow the system down because it may be executing even when the machine (or process) doesn't need immediate attention.

NOTE: The time value is specified in BCD format. If the number is not BCD, or if the number is outside of the range (3–1000) then the routine will not work.

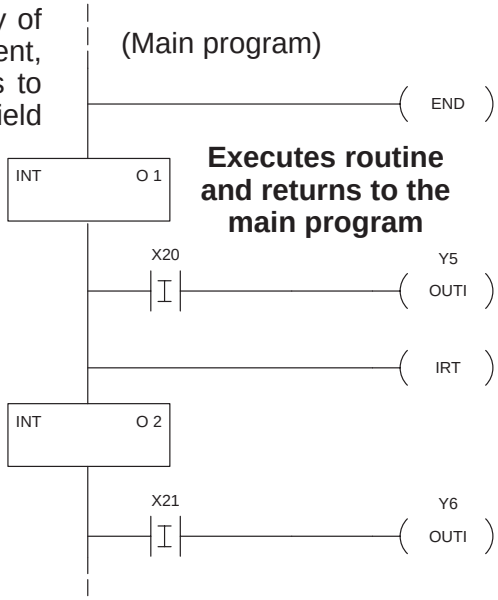
You may have noticed that we didn't mention the D4–430. Timed Interrupts are not available in the D4–430.



Hardware Interrupts: In the vast majority of applications, there is some real world event, such as a switch closing, etc. that needs to trigger an interrupt. By wiring these field devices to an Interrupt Input module, you can easily interrupt the CPU scan cycle when the event occurs.

When the interrupt input point comes on, the CPU suspends the normal program execution and jumps to the interrupt routine that corresponds to the interrupt point, X0 – Interrupt 0, X1 – Interrupt 1, etc. Once the subroutine execution is complete, the CPU returns to the main program.

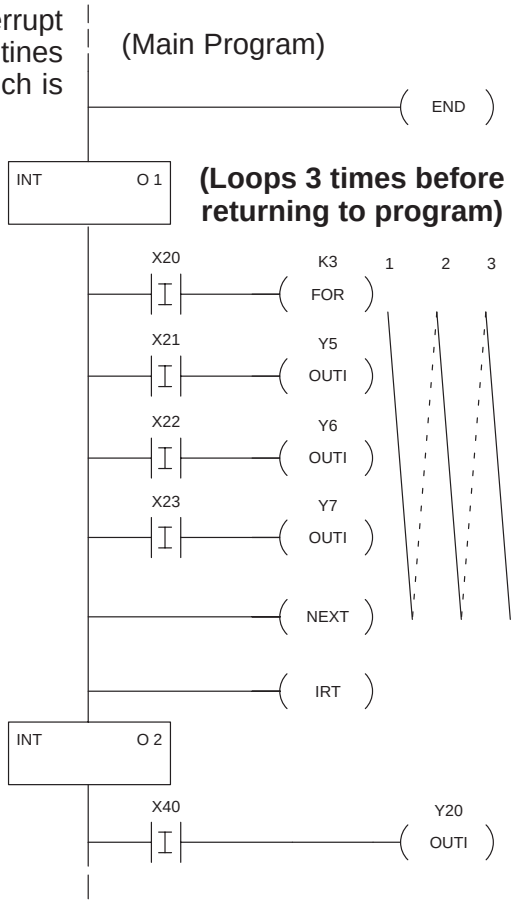
In the example shown here, the CPU automatically handles the transition to the Interrupt subroutine whenever Interrupt input X1 comes on.



What can I use in the Interrupt routines?

In any of the above methods, Interrupt subroutines are required. These subroutines are placed *after* the END statement, which is the last line in the main program.

You can use most any type of instruction in the subroutines. You can have math instructions, data manipulation instructions, etc. However, most people find that the Immediate instructions and the FOR/NEXT looping instructions are the most useful. Immediate instructions immediately read the status of inputs and/or immediately update the outputs. By using the immediate instructions in the subroutine, the CPU can read input points or update output points immediately. By using the FOR/NEXT instructions, you can literally have a "mini-scan" by creating a loop inside of the subroutine.



NOTE: You can use many different types of instructions in the interrupt subroutines. However, if you use instructions with long execution times, such as some math instructions, FOR/NEXT loops, etc., then you may exceed the Watchdog Timer limit. The Watchdog Timer is set at 200ms from the factory. It can be changed with the handheld programmer or ***Direct***SOFT. You can also use a RSTWT instruction inside of the subroutine to reset the Watchdog Timer.

When can the Interrupts Occur?

The previous diagrams showed a simplified CPU scan cycle. However, the CPU can process the interrupts during any portion of the scan cycle. Interrupts can occur in the middle of the communications service, input update, output update, etc. So, this helps ensure the fastest possible response for critical events.

The CPU does not perform any other functions when it is executing the interrupt subroutine. For example, the CPU will not acknowledge any communication requests until it has finished with the interrupt subroutine. Some devices, such as an operator interface, may issue a communications timeout error due to the delay in the response from the CPU. If this occurs, check your device documentation for procedures on changing the communications timeout error settings.

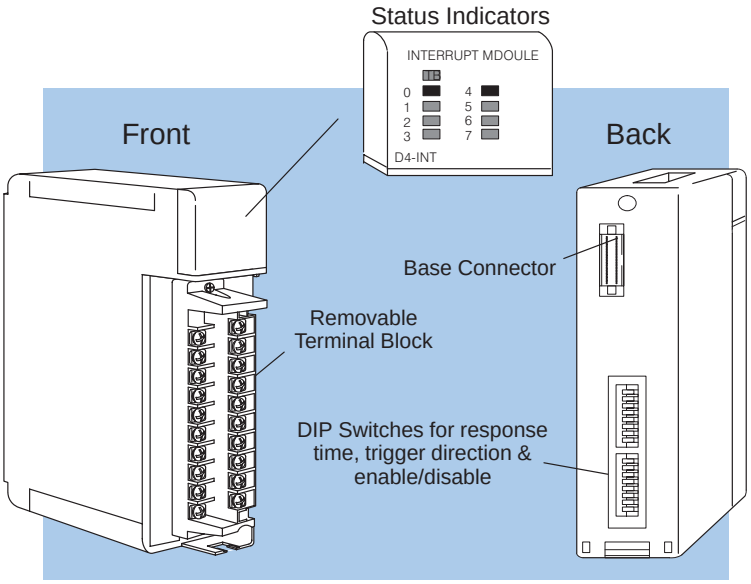
Now that you understand how the CPU processes the Interrupts, you're ready to install (and use) the D4–INT Interrupt module.

Using the Interrupt Input Module..4 easy steps!

Physical layout of the D4–INT components

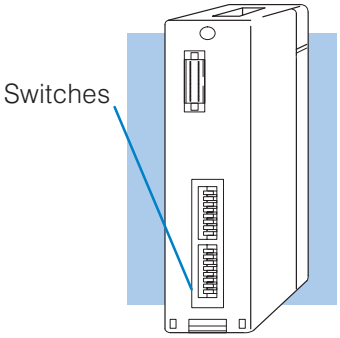
Yes, I know, you were expecting to see the “4 Steps” on this page. But, first, take a minute and familiarize yourself with the basic physical characteristics. It will make the “4 Steps” even easier.

- The dipswitches are located on the back of the module.
- The terminal block can be easily removed by loosening the terminal block retaining screws.
- The LED status indicators show you when an input is active.

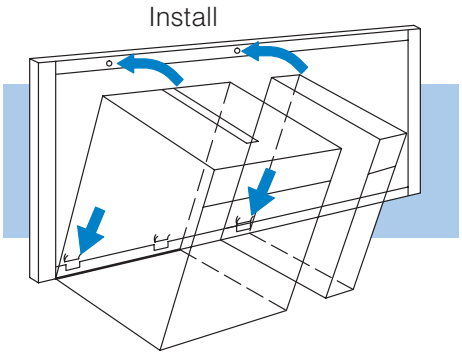


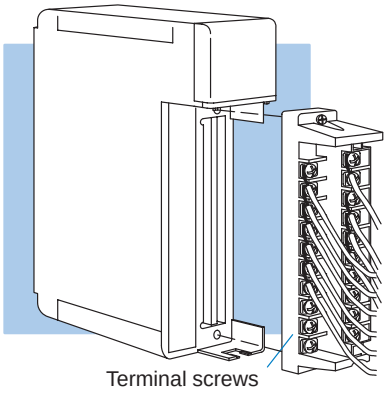
Now that you know your options in using Interrupt processing, it's time to learn the basic setup requirements for the Interrupt Module.

Step 1:
Set the DIP Switches.



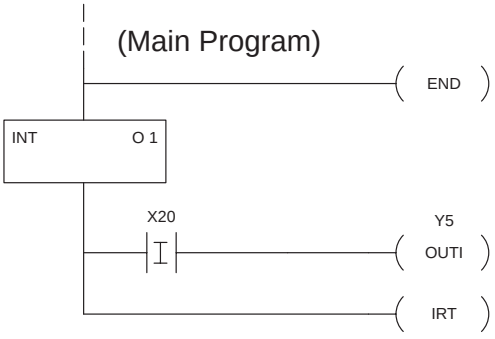
Step 2:
Install the Module in the Base.





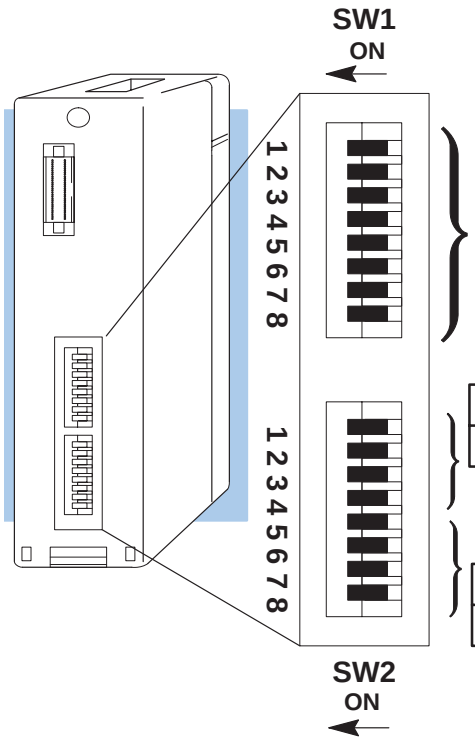
Step 3:
Connect the Field Wiring.

Step 4:
Write the Control Program with simple RLL Instructions.



Step 1: Setting the DIP Switches

There are two banks of DIP switches, SW1 and SW2, located on the back of the module. You can enable/disable interrupt points; adjust the response time, and set the triggering to be either on the rising edge or trailing edge of the input signal. The following table shows an overview of the switch settings. If you are unsure of the meaning of some of these options, just keep reading. The remaining paragraphs explain these in more detail.



SW1 Position	1	2	3	4	5	6	7	8
Input	0	1	2	3	4	5	6	7

ON = ENABLE OFF = DISABLE

By setting the switches1 thru 8 of SW1 to the ON or OFF position, you can ENABLE or DISABLE the corresponding inputs as indicated.

SW2 Position	1	2	3	4
Input	0, 1	2, 3	4, 5	6, 7

ON=Interrupt triggered by the rising edge
OFF= Interrupt triggered by the falling edge

By setting the switches1 thru 4 of SW2, you can select whether the interrupt is triggered on the rising or falling edge.

SW2 Position	5	6	7	8
Input	0	1	2	3

ON=Low speed (slower response time)
OFF= High speed (faster response time)

By setting switches 5 thru 8 of SW2, you can select the response delay time from one of two ranges for inputs 0 thru 3. This feature is not available for inputs 4 thru 7, which are always set at the fastest response time.

Switch 1: Enabling/Disabling Input Points

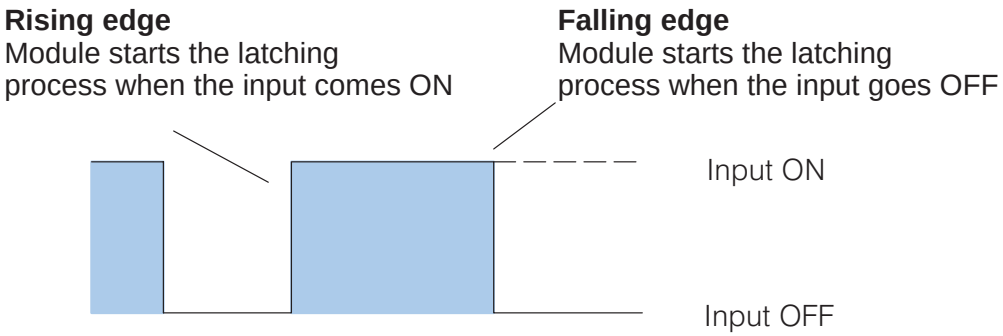
Positions 1–8 on Switch 1 enable (or disable) each individual interrupt point. Your application may not require that all 8 inputs be used as interrupt inputs. For example, you may only need one interrupt input. In this case, you could use the other 7 points as normal discrete inputs.

You can enable the interrupt for each individual point by sliding the DIP switch to the position ON for those input points you wish to enable. For example , sliding position 4 to the ON position will enable input point X3.

NOTE: Any points *not* designated as interrupt points can be used as *normal input points*. For normal inputs, set the switch to the OFF position.

Switch 2: Selecting Rising or Falling Edge Triggering

Positions 1,2,3 and 4 on Switch 2 determine whether the interrupt is triggered on the rising (leading) edge of the input signal or on the falling (trailing) edge of the input signal. This is often called “triggering.” The examples earlier in this manual showed the timing diagrams with a Rising edge trigger. They could just as easily have been shown using the falling edge trigger. The following diagram shows the difference.

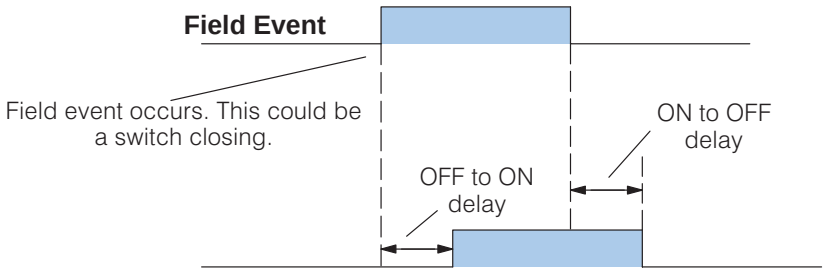


Set the dipswitches to the ON position for rising edge triggering, or set the dipswitches to OFF for falling edge triggering. You will also notice that the points are “paired.” That is, one switch sets the triggering for *two* points. For example, If you slide the position 2 switch to the ON position, then *both* inputs 2 and 3 will generate interrupts on the *rising edge.*

Switch 2: Selecting the Response Delay Time

Positions 5,6,7 and 8 on Switch 2 control the response delay time for the first four input points only (X0 – X3). The response delay is defined as follows:

- **Off-to-On Delay** — the amount of time between the occurrence of the field event, such as a switch closing, and the Interrupt module point turning on.
- **On-to-Off Delay** — the amount of time after the field event is complete, such as a switch opening, and the Interrupt module point turning OFF.



There are two speeds for response.

- **Slow Response** — 0.88ms–6.47ms for off–to–on delay and 1.64ms–9.81ms for on–to–off delay.
- **Fast Response** — 0.08ms to .59ms for off–to–on delay and 0.15ms–0.89ms for on–to–off delay.

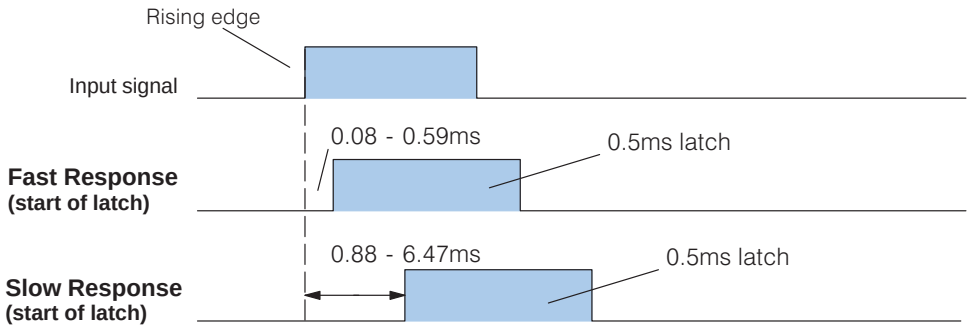
You will notice a range of time shown. This means that there can be a variance in the response times, so we recommend that you plan your application for the worst case scenario.

Selecting a slower response time can be helpful if your critical field event input signal is electrically noisy, or, if you expect contact bounce on the switch. (The slower response provides more filtering and helps reduce the possibility of false input signals. If you have input signals that are subject to these possibilities, it is best to connect them to the first four inputs on the module.)

NOTE: This feature is available for the first four inputs (X0, X1, X2, and X3) only. The response time for inputs X4, X5, X6 and X7 is not configurable. The off–to–on delay and the on–to–off delay for these are fixed at the fast response delay time.

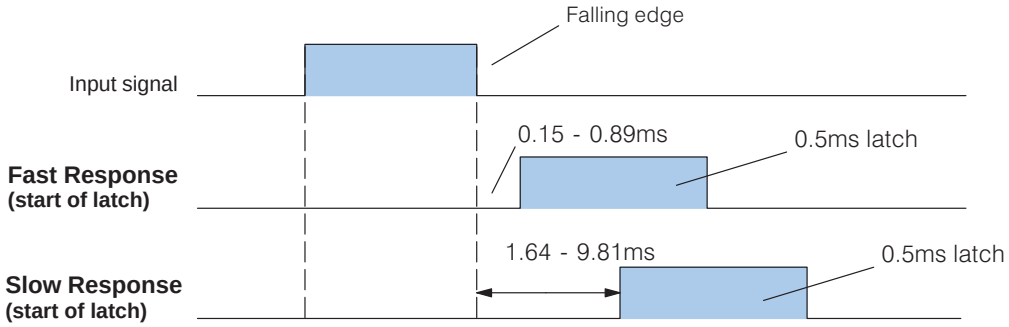
The Response Delay helps you understand how long it will take the Interrupt module to latch the interrupt signal. The following diagram shows how the Off-to-On delay affects a signal with Rising edge triggering.

Response Delay Range for Rising Edge Triggering



For falling edge triggered signals, the On-to-Off delay is more important. This is because the Interrupt module “waits” for a normally high signal to go low before it recognizes an interrupt. So in this case, the On-to-Off delay filters out any bounce or false signals.

Response Delay Range for Falling Edge Triggering



Step 2: Installing the Module in the Base

WARNING: To minimize the risk of electrical shock, personal injury, or equipment damage, always disconnect the system power before installing or removing any system component.

Module Restrictions

There are a few restrictions that you need to consider if you are using this module.

- The module must be installed in slot 0, which is adjacent to the CPU.
- The module *cannot* be installed in an expansion or remote base. That is, it must be installed in the CPU base.
- If you're using a D4–440 CPU, you can have two interrupt modules. They must be placed in slots 0 and 1.
- If you're using a D4–440 CPU and a Timed Interrupt subroutine (INTO17) then you *cannot* use the 8th interrupt point on the second interrupt module. (The CPU always uses INTO17 as the Timed Interrupt subroutine. If you're using the timed interrupt, disable the hardware interrupt point by using the dipswitch on the rear of the module.)

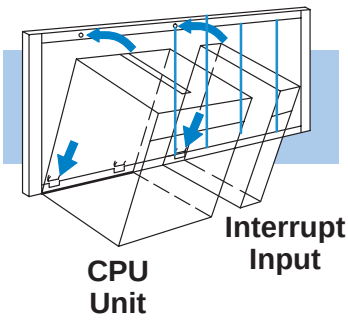
CPU Compatibility

If you have purchased your CPU unit from PLC***Direct***, your firmware will automatically support the interrupt input module. This module may also work with 405 CPUs provided by previous vendors. However, if you have an older 405 CPU with firmware prior to version 1.6, you will have to upgrade your firmware. Contact PLC***Direct*** for more information.

Installing the Module in the Base

With the power disconnected, you are now ready to install the module in the base. Here are four steps for making the installation:

1. Notice the module has a plastic tab at the bottom and a screw at the top.
2. With the module tilted slightly forward, hook the plastic tab on the module into the notch on the base.
3. Next, gently push the top of the module back toward the base until it is firmly seated into the base.
4. Now tighten the screw at the top of the module to secure the module to the base.

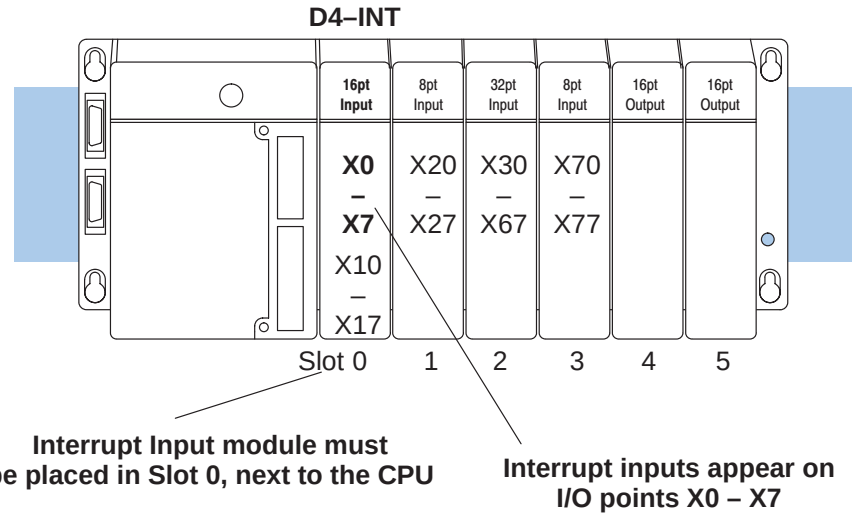


I/O Points Used

Each Interrupt module consumes 16 input points. The first group of 8 points can be used as the interrupt points. The second group of 8 points are used internally by the interrupt module. If you do not use all of the first 8 points as interrupts, then you can use them as normal discrete inputs. You cannot use the next 8 points for other purposes. They are always reserved for the Interrupt module internal operation.

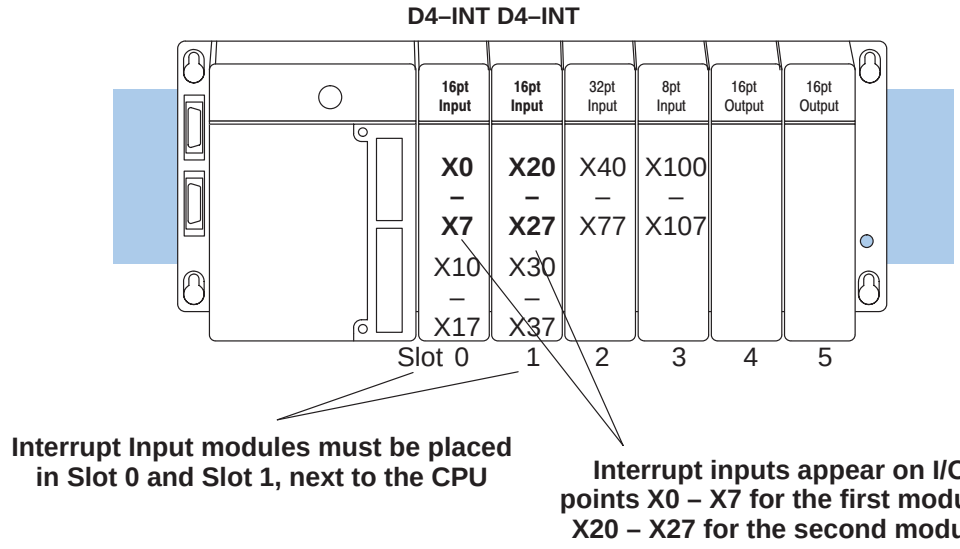
I/O Assignments with a D4–430

You can only use one Interrupt module with the D4–430 CPU. Since the module must be placed in slot 0, the points used are X0 – X17. The following diagram shows an example system with I/O assignments.



I/O Assignments with a D4–440

Each Interrupt module consumes 16 input points. You can use two Interrupt modules with the D4–440 CPU. Since the modules must be placed in slot 0 and slot 1, the points used are X0 – X17 for the first module, and X20–X37 for the second interrupt module. If you only use one interrupt module, install it in slot 0. (You can install another type of module in slot 1.) The following diagram shows an example system with I/O assignments for a D4–440 with two Interrupt modules.



NOTE: The D4–440 supports manual I/O configuration. That is, you can manually assign the I/O addresses in any order. If you use manual configuration, make sure you *must* install the Interrupt modules (and use the addressing) as shown above.

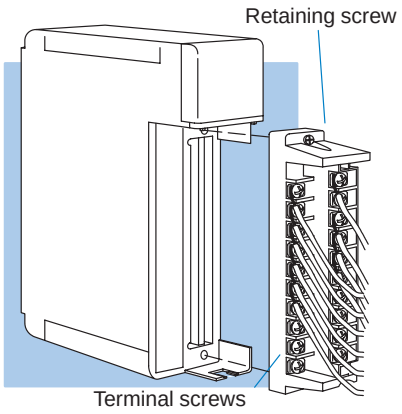
Now that you understand the module placement and the I/O assignments, you're ready to connect the field wiring.

Step 3: Connecting the Field Wiring

WARNING: To minimize the risk of personal injury or property damage, remove all power from the PLC and field devices before wiring the module.

Wiring Guidelines

The D4–INT Interrupt Input Module features a removable terminal block. It is held into place by two retaining screws. You must first remove the front cover of the module prior to wiring. To remove the cover press the bottom tab of the cover and tilt the cover up to remove it from the module. Now loosen the retaining screws and lift the terminal block away from the module.



Consider the following wiring guidelines.

1. There is a limit to the size of wire the modules can accept. The table lists the maximum AWG for each module type (smaller wire is also acceptable). The interrupt module follows the 16 point guidelines.

Module Type	Maximum AWG
8 point	12
16-point	14
32-point (common)	20
32-point (other)	24

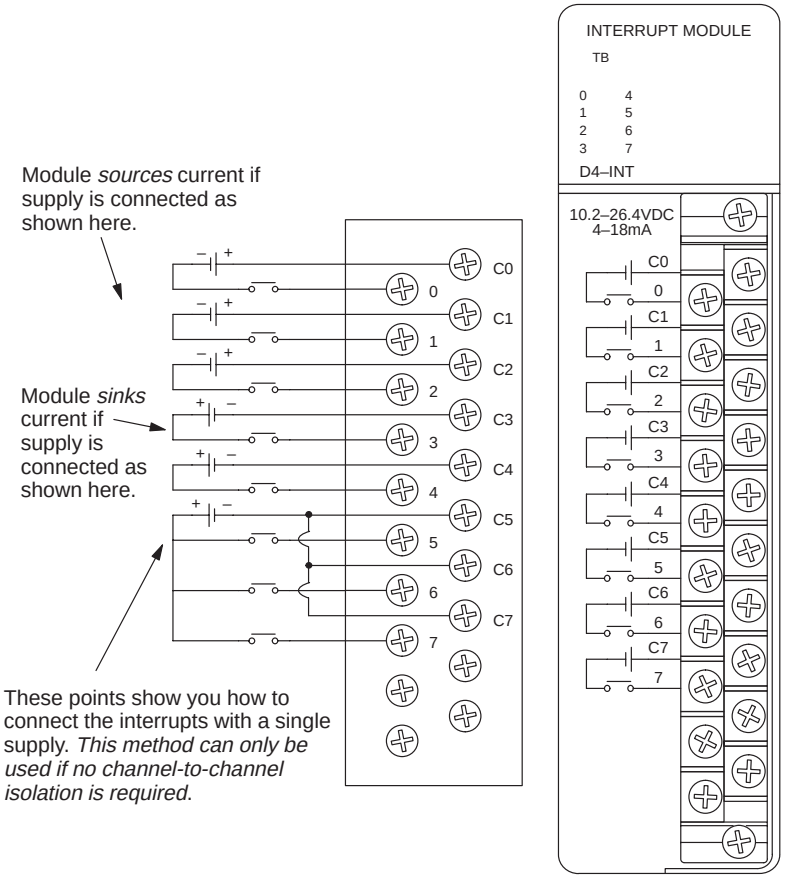
NOTE: 12 AWG Type TFFN or Type MTW can be used on 8pt. modules. 14 AWG Type TFFN or Type MTW can be used on 16pt. modules. Other types of wire may be acceptable, but it really depends on the thickness of the wire insulation. If the insulation is too thick and you use all the I/O points, then the plastic terminal cover may not close properly.

2. Always use a continuous length of wire, do not combine wires to attain a needed length.
3. Use the shortest possible cable length.
4. Where possible, use wire trays for routing.
5. Avoid running wires near high energy wiring.
6. If possible, avoid running input wiring in close proximity to output wiring.
7. To minimize voltage drops when wires must run a long distance, consider using multiple wires for the return line.
8. Where possible, avoid running DC wiring in close proximity to AC wiring.
9. Avoid creating sharp bends in the wires.

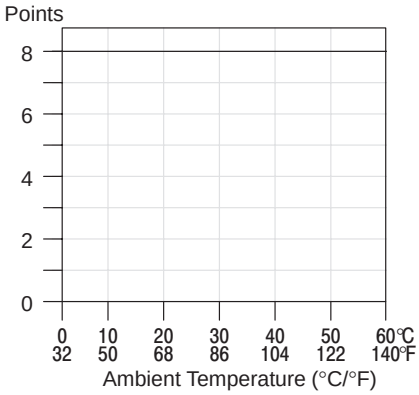
Typical Field Wiring and Internal Module Wiring

Shown below is a wiring diagram with typical field wiring and details relating to the D4–INT module's internal wiring. There are a few things you need to know before you connect the field wiring.

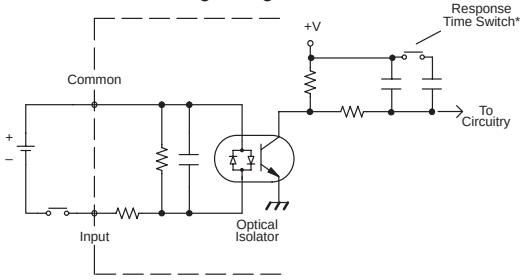
- Each channel is isolated, so you can use sinking or sourcing configurations independently for each point.
- Unused inputs do not require any connections.



Derating Chart for D4–INT



Current sourcing configuration shown

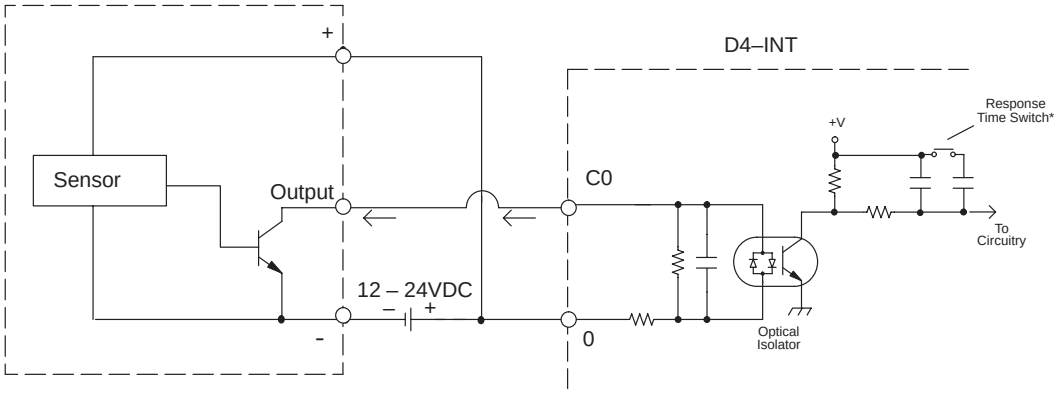


* Response delay switch is set with Switch 2, positions 5–8 on rear of the module. This is only for points X0 – X3. Points X4 – X7 are fixed at the fastest response.

Solid State Field Device Wiring

The following diagrams show how to connect solid state field devices to the D4–INT Interrupt Input module.

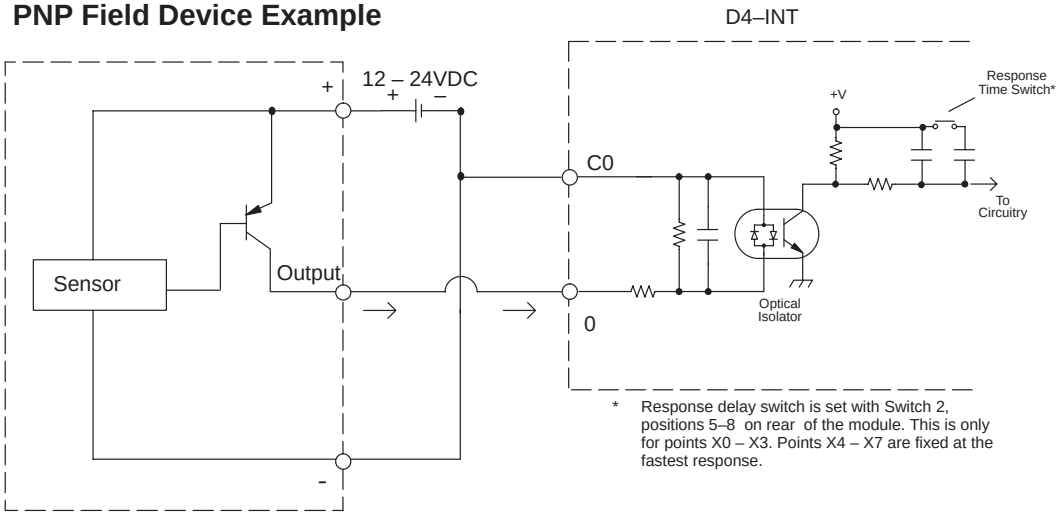
NPN Field Device Example



(NPN) Current Sinking Field Device

* Response delay switch is set with Switch 2, positions 5–8 on rear of the module. This is only for points X0 – X3. Points X4 – X7 are fixed at the fastest response.

PNP Field Device Example

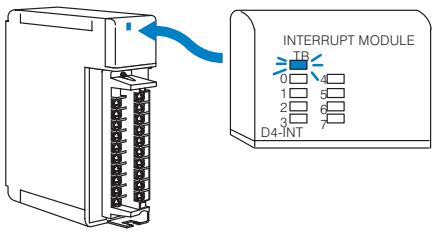


(PNP) Current Sourcing Field Device

* Response delay switch is set with Switch 2, positions 5–8 on rear of the module. This is only for points X0 – X3. Points X4 – X7 are fixed at the fastest response.

Check the Terminal Block

After you have finished making the wiring connections, install the terminal block on the module, making sure the terminal block is tightly seated. Be sure to tighten the retaining screws. Also verify that the loose terminal block LED is off when the system power is applied. If the TB LED is on, then the terminal block is not connected properly.



Once you have the field wiring connected, you're ready to write the control program.

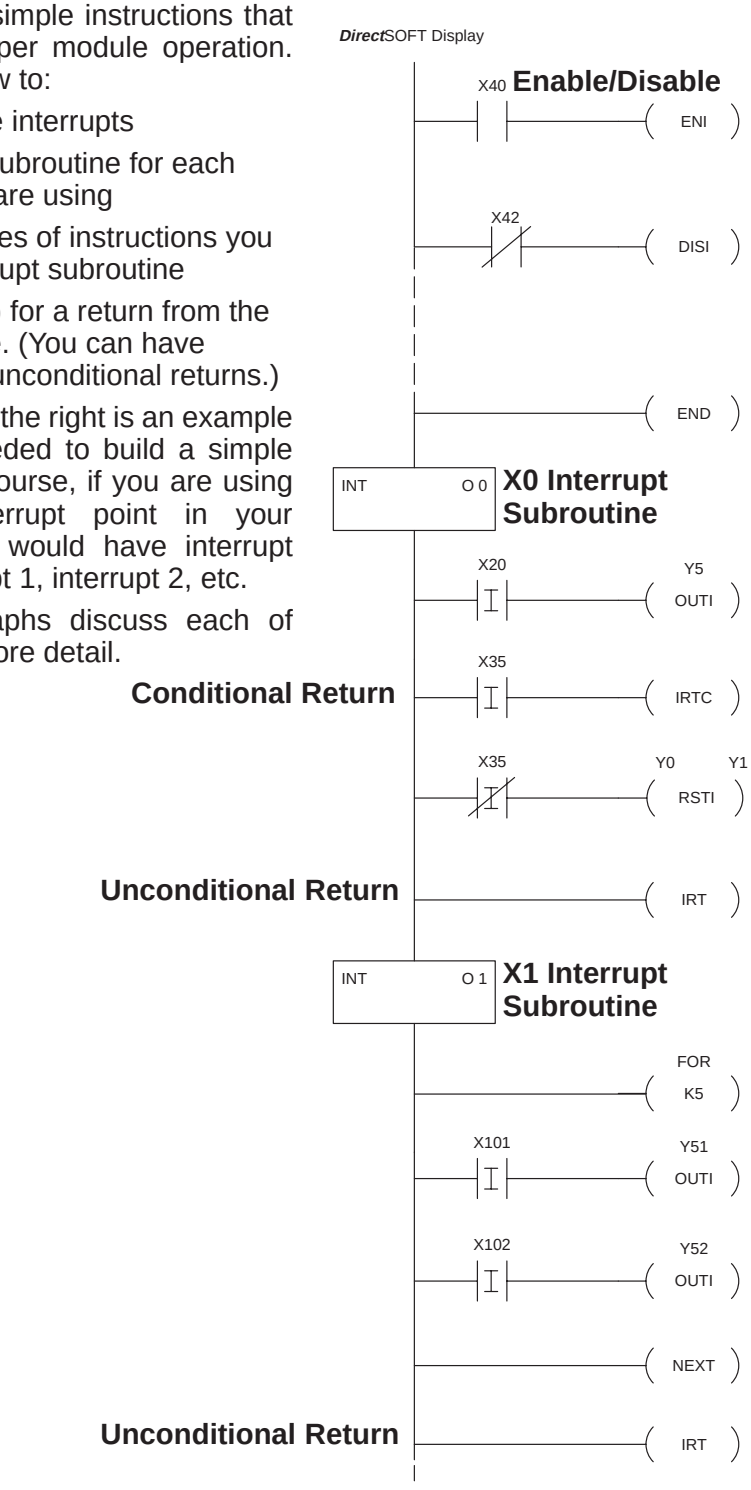
Step 4: Writing the Control Program

There are only a few simple instructions that are necessary for proper module operation. This section shows how to:

- Enable and Disable interrupts
- Enter an interrupt subroutine for each interrupt point you are using
- Understand the types of instructions you can use in an interrupt subroutine
- Set the condition(s) for a return from the interrupt subroutine. (You can have conditional and/or unconditional returns.)

The program shown to the right is an example of the instructions needed to build a simple interrupt solution. Of course, if you are using more than one interrupt point in your application, then you would have interrupt subroutines for interrupt 1, interrupt 2, etc.

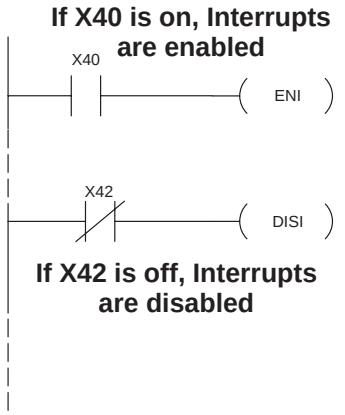
The following paragraphs discuss each of these instructions in more detail.



Task 1: Enable / Disable the Interrupts

The CPU does not automatically monitor the interrupt input signals. Instead, you have to use an Enable Interrupts (ENI) instruction in your ladder program to do this. Why does it work this way? Simple. There may be times when you don't want the CPU to acknowledge the interrupt signals. For example, you may only want to monitor the interrupts if a certain safety parameter is true (or false). *If you don't include this instruction, the interrupts will be ignored*.

The ENI instruction is used just like an output coil, that is, you use an input contact, comparative boolean contact, etc. that enables the interrupts.



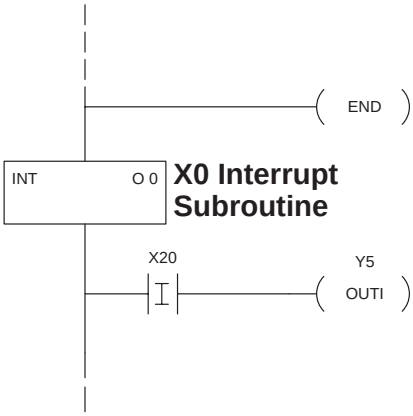
Since you can enable the interrupts, it also makes sense that you be able to disable the interrupts. (Actually, if you do not tell the CPU under what conditions it is to disable the interrupts once they have been enabled, it will assume they will always be enabled.) You can disable the interrupts with the Disable Interrupts (DISI) instruction. This instruction also appears as an output coil and can be triggered like any other type of output coil.

Helpful Hints: In our ***Direct***SOFT programming software, the ENI and DISI instructions are found on the Coil Browser. Look under the class listed as "Interrupt."

Task 2: Place Numbered Interrupt Subroutines after the END Statement

Each interrupt input point needs a corresponding interrupt subroutine. These interrupt subroutines are placed *after* the END statement, which is the last instruction in the ladder program.

The interrupt subroutine for X0 should be labeled INTO0, the routine for X1 should be labeled INTO1, etc. If you're using two interrupt modules (D4–440 only), then the second module would use INT10 for X20, INT11 for X21, etc. The CPU automatically executes the interrupt subroutine that corresponds to the interrupt input point.



NOTE: That's the letter "O" in front of the number. The ***Direct***SOFT screen automatically puts an "O" there for you and it *must* be there. Don't delete it!

WARNING: If you have a D4–440 CPU, two hardware interrupt modules, and you are using a Timed Interrupt, then you cannot use INTO17 as a hardware triggered interrupt subroutine. (This means you could not use the 8th interrupt input point on the second Interrupt module.) The software interrupt is triggered off of Interrupt 17, and you don't want a conflict.

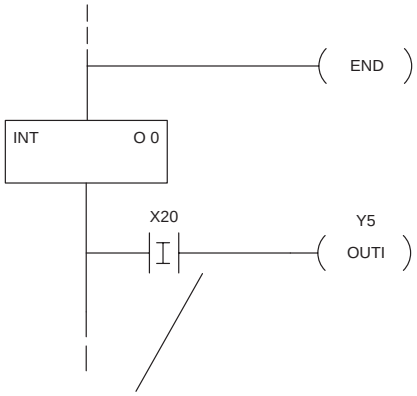
Helpful Hints: In our ***Direct***SOFT programming software, the INT instructions are found on the Box Browser. Look under the class listed as "Interrupt."

Task 3: Understand the Types of Instructions used in Interrupt Subroutines

You can use many different types of instructions in the interrupt subroutines. For example, you may need comparative boolean contacts, math instructions, etc.

Controlling I/O Points in Subroutines:
In most cases, the goal is a fast response. So you should try to use the Immediate Instructions to update any I/O points. These instructions look like regular contacts with an "I" in them. They work differently from the regular contacts because the CPU *immediately* reads the input status from the module and *immediately* updates the output module when the instructions are executed. This provides a much faster response. (The CPU normally only updates the inputs and outputs in a "batch update" at the beginning and end of the scans respectively.)

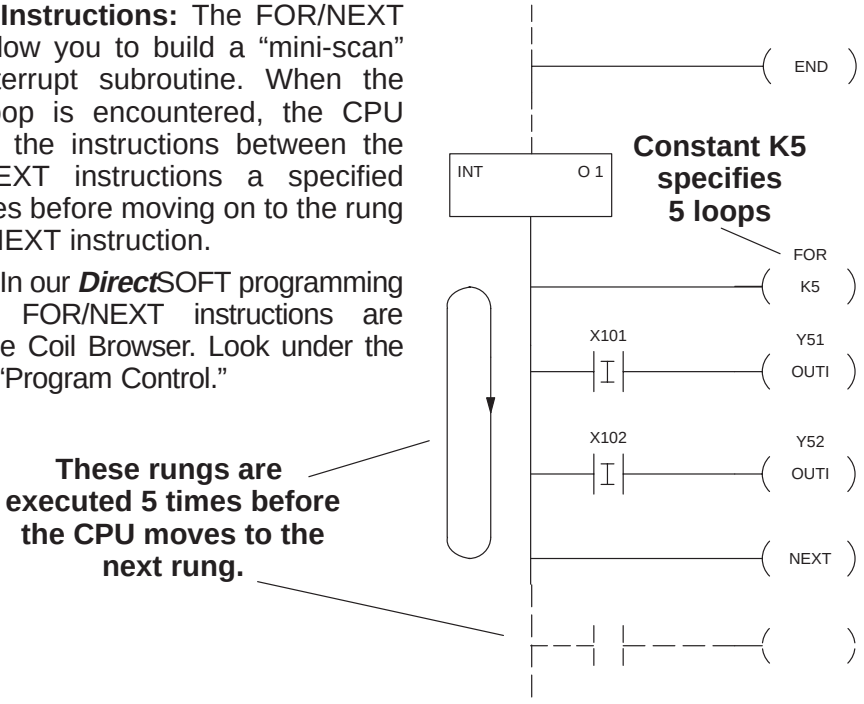
Helpful Hints: In our ***Direct***SOFT programming software, the Immediate Instructions are available on the Editing Tool Box, the Contact Browser, or the Coil Browser.



Read the status of input X20 immediately. If it is ON, turn on output Y5 immediately.

FOR / NEXT Instructions: The FOR/NEXT instructions allow you to build a "mini-scan" within the interrupt subroutine. When the FOR/NEXT loop is encountered, the CPU loops through the instructions between the FOR and NEXT instructions a specified number of times before moving on to the rung following the NEXT instruction.

Helpful Hints: In our ***Direct***SOFT programming software, the FOR/NEXT instructions are available on the Coil Browser. Look under the class listed as "Program Control."

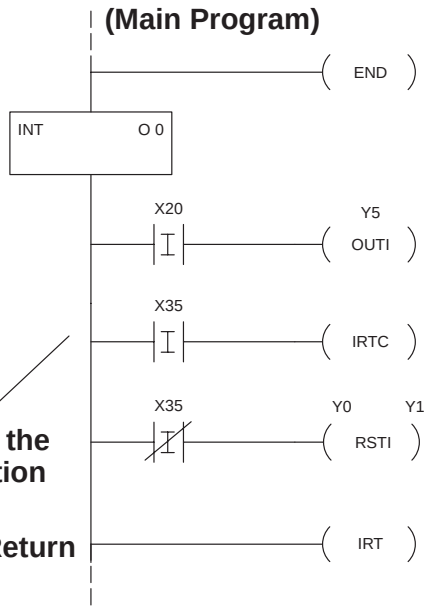


NOTE: You can use many different types of instructions in the interrupt subroutines. However, if you use instructions with long execution times, such as some math instructions, FOR/NEXT loops, etc., then you may exceed the Watchdog Timer limit. The Watchdog Timer is set at 200ms from the factory. It can be changed with the handheld programmer or ***Direct***SOFT. You can also use a RSTWT instruction inside of the subroutine to reset the Watchdog Timer.

Task 4: Enter the Conditions for a Return from the Subroutine

You have to describe the condition for a return from the interrupt subroutine. That is, you have to tell the CPU when to return to the main program execution. There are two instructions available: IRT (immediate return with no condition) and IRTC (immediate return with a condition).

The example shows an interrupt subroutine that uses both commands. The conditional return (IRTC) can appear anywhere in the subroutine, but unconditional return (IRT) *must* be the very last instruction in the subroutine.



Helpful Hints: In our ***Direct***SOFT programming software, the IRTC and the IRT instructions are available on the Coil Browser. Look under the class listed as “Interrupt.”

You’ve just seen all of the steps involved in writing the control program. Now you’re ready to use the Interrupt Input module to solve your high-speed interrupt problems!

Troubleshooting

If the D4–INT interrupt input module does not seem to be working properly, check the following items.

1. **Wiring connections:** Incorrect and loose wiring cause the majority of problems. Verify you have wired everything correctly and that the connections are tight.
2. **DIP switch settings:** Make sure you have set all the DIP switches correctly. (Page 10 shows the various settings available.)
3. **Noise or bounce:** If the field device you have connected has too much contact bounce, or if you are possibly picking up some noise on the input, try additional input filtering by setting the DIP switches to the Slow Response Delay. This is available on X0–X3 only. Refer to Page 10 for the proper DIP switch settings.

The following table provides additional troubleshooting details:

Symptoms	Possible Causes	Corrective Action
TB indicator is ON	Terminal block is loose.	Make sure terminal block is securely connected to the module.
Input indicator lights do not come on.	1. Power not present. 2. Invalid DC level signal being sent to module.	1. Apply power. 2. Check the voltage level of the incoming signal.
CPU does not initiate actions in sub-routine quickly enough.	1. Immediate instructions have not been used in subroutine. 2. Critical action logic too far down in scan of subroutine.	1. Use immediate instructions where possible. 2. If possible, make sure critical actions are placed on rungs immediately following the INT box.
Interrupt does not occur.	1. ENI and DISI instructions are not entered properly. 2. Input signal does not last longer than the shortest selected Response Delay time. 3. Bad module.	1. Check these commands in the main program. 2. Find a way to make the input pulse last longer. 3. Replace the module.

Specifications

Environmental Specifications	Operating Temperature	0° to 60° C
	Storage Temperature	–20° to 80° C
	Operating Humidity	5 to 95% (non-condensing)
	Air Composition	No corrosive gases permitted
	Vibration	MIL STD 810C 514.2
	Shock	MIL STD 810C 516.2
	Voltage Isolation	1500 VAC, 1 minute duration
	Insulation Resistance	10M ohms at 500 VDC
	Noise	NEMA ICS3–304
Operating Specifications	Power Budget Requirement	500 ma @ 5 VDC
	Maximum number of modules	D4–430, 1 D4–440, 2
	Location of module	CPU Base only, D4–430 –Slot 0 only D4–440 –1st module, Slot 0 2nd module, Slot 1
	Diagnostics	Loose terminal block
	Maximum Voltage	26.5 VDC
	Input Range	10.2 VDC to 26.4 VDC, 3.8mA @ 12 VDC, 8.3mA @ 24 VDC 9.5 mA maximum
	ON input level	9.5 VDC
	OFF input level	3.0 VDC
	Minimum ON current	4.0 mA
	Maximum OFF leakage	1.5 mA
	Impedance	2.4kΩ
	Internal power consumption	5 VDC, 100mA max.
	OFF to ON response delay (fast mode)	0.8 ms to 0.59 ms
	ON to OFF response delay (fast mode)	0.15 ms to 0.89 ms
	OFF to ON response delay (slow mode)	0.88 ms to 6.47 ms
	ON to OFF response delay (slow mode)	1.64 ms to 9.81 ms
	Interrupt priority	Highest priority: X0 Lowest priority: X7 (if two modules are used with D4–440, then priority is X0 – X7, then X20 – X27)