The Importance of Relay and Programmable Logic Documentation

Jason Young and Derrick Haas
Schweitzer Engineering Laboratories, Inc.

Presented at the
62nd Annual Georgia Tech Protective Relaying Conference
Atlanta, Georgia
May 21–23, 2008

Previously presented at the
61st Annual Conference for Protective Relay Engineers, April 2008

Originally presented at the
DistribuTECH Conference, January 2008

The Importance of Relay and Programmable Logic Documentation

Jason Young and Derrick Haas, *Schweitzer Engineering Laboratories, Inc.*

***Abstract*—With increasing use of microprocessor-based relays, incorporating logic into relays has become more common. Traditional relay and control schemes used dc control schematics to describe how schemes functioned. The ability to see electrical paths aided technicians during testing and troubleshooting. As logic is incorporated into relays, the ability to visualize logic is important for complete testing and troubleshooting. This paper describes the need for documenting relay logic, highlighting the importance of testing, and understanding relay logic. The paper describes real-world examples of misoperations due to failure to fully test logic schemes, and it reviews several documentation methods.**

I. Introduction

The importance of documenting settings and wiring has been recognized since the advent of protective relaying. A required part of relay panel installation typically includes a detailed dc schematic and a point-to-point or physical wiring diagram. IEEE has defined device numbers [1], and the industry has established recognizable symbology to accurately and concisely represent various devices.

Multifunction microprocessor-based relays incorporate both multiple relay functions and programmable logic capability in one box. This logic capability allows various "logic schemes," previously implemented by wiring auxiliary relays, timers, and devices together, to be implemented in a single device using settings. Manufacturers' logic settings have taken on a variety of forms, from actual settings that users fill in, to a code similar to a simple programming language, to software utilities that are graphical in nature.

Given such a spectrum of both capabilities and differences in nomenclature, a key aspect of using the logic capabilities of microprocessor-based relays is the methodology used to document the logic located within the relay. With logic schemes implemented in wiring, the logic functions are documented using the dc schematic, along with any settings associated with a device. The technician and/or engineer tasked with testing or troubleshooting an installation can visualize the logical function as an electrical path on the diagram. For example, Fig. 1 clearly shows that when Contact A and Contact B are both closed, Coil C is electrically energized or asserted. The electrical path from positive dc to negative dc could be traced with a highlighter, and the logic functioning could be easily verified and tested.

Several considerations must be factored in when using microprocessor-based relays. First, because the logic is no longer part of the physical wiring, the control schematic no longer serves as a record of the logic scheme.

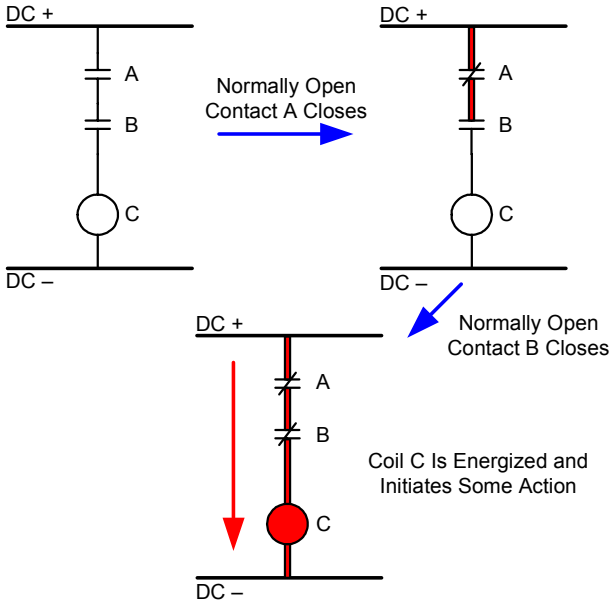


Fig. 1. Example DC Schematic

Second, different manufacturers use different nomenclature for logic settings. If two different relays are used together as part of a scheme, it is difficult to clearly document interdependent schemes.

Third, added functions, such as the capability to do math and peer-to-peer communications protocols (i.e., MIRRORED BITS® communications and IEC 61850 GOOSE) are appearing in relays. While the peer-to-peer protocols and methods of communicating different pieces of information continue to change, a common method of representing inputs is needed, whether it is a wired contact, the result of a logic calculation, or a bit from a serial or Ethernet message. For example, a relay receiving a control command from a DNP master calling for a breaker to open may require the object type and index information as part of the settings. The same command using the IEC 61850 protocol may require logic-node addressing. Using a proprietary peer-to-peer protocol may require other settings. However, from the standpoint of the logic scheme, the protocol and associated settings should not affect the overall logic function. The communications connections, medium used, associated settings, information regarding the protocol, etc., are certainly important to document, but placing this information on a logic diagram may serve only to complicate things. Consider where this information is documented. Relay and SCADA technicians probably would not care to have protocol information, connection diagrams, and addressing of

points in the same set of drawings as the logic diagram. However, certain engineering and testing technicians may need or want to access both pieces of information.

To address these considerations, several different methods of documenting programmable relay logic have been developed by manufacturers, utilities, industrial facilities, and others to accurately depict logic functioning within the relay. These include the following:

- A written description of relay logic
- A document listing the logic settings or code, or a file that contains the settings or code
- A graphical representation of the scheme as logic gates
- A graphical representation of the scheme as ladder logic
- A graphical representation of logic and math functions, using both logic gates and symbology from the control systems and communications fields
- A combination of the above

The use of a particular method appears to be fairly arbitrary, although different manufacturers and devices will have certain preferences. For example, PLCs will typically utilize the ladder logic representation of a particular logic scheme, because that representation accurately depicts not only the overall function but also the device's inherent processing—the scan and addressing of memory associated with PLCs. Microprocessor-based relays and controllers seem more varied in their choice of methods.

II. Comparison of Several Methods of Documenting Logic

In order to evaluate the methods listed in Section I, we show a simple example of using programmable logic to implement breaker failure in a typical distribution relay [2]. Assuming the relay is connected with the dc schematic outlined in Fig. 2, our documentation focuses on the relay's programmable logic.

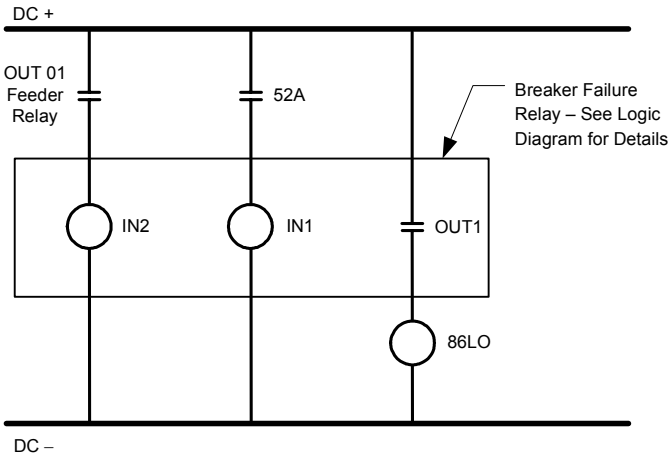
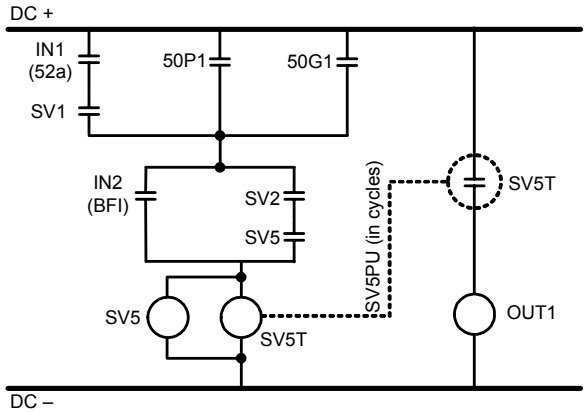


Fig. 2. DC Schematic

The first method of documenting programmable logic is to provide a clear, written description of how the scheme is expected to operate. The description for our example follows:

The basic functioning of the breaker failure scheme is described as follows: When the breaker failure initiate signal is asserted (typically a tripped signal from another relay, in this case from another feeder relay), if the breaker failure initiate remains asserted and the breaker is closed, then after a certain period of time, we declare breaker failure. The time is typically a user setting on the order of ten cycles for a typical distribution breaker (possibly quicker for faster breakers). Indications that the breaker is still closed are the presence of current (either phase or ground) and the 52a contact being asserted. Because many believe that breaker status contacts can be unreliable, a user setting, SV1, is included to enable contact supervision. In addition, in many cases, it is desirable to "latch in" the breaker failure initiate signal. In other cases, latching in the breaker failure initiate signal may not be desirable. Another user setting, SV2, is included to determine whether the breaker failure is latched in or not.

The second method is to use an "equivalent dc schematic." The schematic in Fig. 3 depicts the relay connections to the rest of the control system. A simple way to describe the programmable logic inside the relay is to simply treat it as dc logic. To put it another way, decide how you would implement this scheme using electromechanical relays. Because Relay Word bits, AND functions, OR functions, and NOT functions all can be represented as contacts and coils in series, parallel, or other configurations, the dc schematic is a valid method as well. For our example, the schematic is shown in Fig. 3.

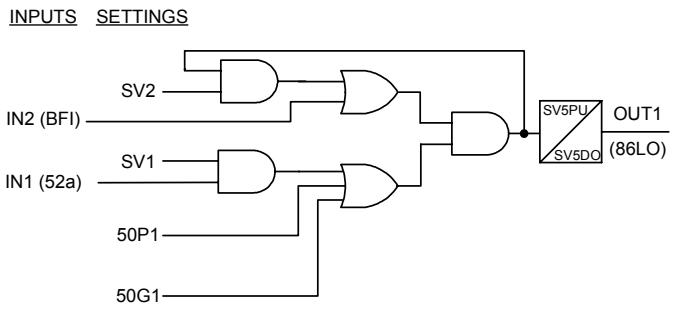


SETTINGS LEGEND

SV1	Enable 52a supervision (SV1 = 0 corresponds to no 52a contact supervision, SV1 = 1 corresponds to 52a supervision enabled)
SV2	Enable Breaker Failure Seal In (SV2 = 0 corresponds to disabling latching of breaker failure, SV2 = 1 corresponds to latching of breaker failure)
50P1	Phase Current Detector (Low Set Instantaneous Element)
50G1	Ground Current Detector (Low Set Instantaneous Element)
SV5PU	Breaker Failure Timer (Set in Cycles)

Fig. 3. Example Scheme Represented as an Equivalent DC Schematic

The third method is to represent the logic as "logic gates." For our example system, the logic gate representation is shown in Fig. 4.

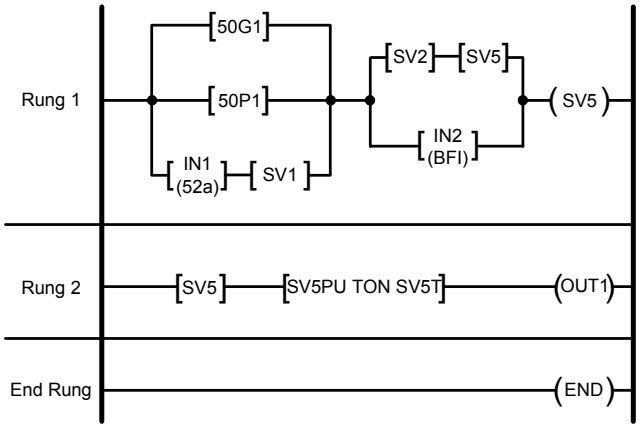


SETTINGS LEGEND

SV1	Enable 52a supervision (SV1 = 0 corresponds to no 52a contact supervision, SV1 = 1 corresponds to 52a supervision enabled)
SV2	Enable Breaker Failure Seal In (SV2 = 0 corresponds to disabling latching of breaker failure, SV2 = 1 corresponds to latching of breaker failure)
50P1	Phase Current Detector (Low Set Instantaneous Element)
50G1	Ground Current Detector (Low Set Instantaneous Element)
SV5PU	Breaker Failure Timer (Set in Cycles)
SV5DO	Drop-Out Timer NOT USED (Leave Set to Zero Cycles)

Fig. 4. Example Scheme Represented as Logic Gates

The fourth method is to represent logic as "ladder logic." Ladder logic is commonly used in PLC products and is one of the accepted graphical programming languages in the IEC 61131 standard. The other accepted graphical method, the functional block diagram, is very similar to the logical gate representation. For this reason, a discussion of functional block diagrams is not included in this paper. Our example system of the ladder logic representation is shown in Fig. 5.



SETTINGS LEGEND

SV1	Enable 52a supervision (SV1 = 0 corresponds to no 52a contact supervision, SV1 = 1 corresponds to 52a supervision enabled)
SV2	Enable Breaker Failure Seal In (SV2 = 0 corresponds to disabling latching of breaker failure, SV2 = 1 corresponds to latching of breaker failure)
50P1	Phase Current Detector (Low Set Instantaneous Element)
50G1	Ground Current Detector (Low Set Instantaneous Element)
SV5PU	Breaker Failure Timer (Set in Cycles)

Fig. 5 Example Scheme Represented as Ladder Logic

A fifth representation is a text list of the settings. For this particular scheme, the settings are as follows:

SV5 = (SV2 * SV5 + IN2) * (SV1 * IN1 + 50P1 + 50G1)
SV5PU = 10 (pickup time in cycles)
SV5DO = 0
OUT1 = SV5T

Looking at the five different representations, we can compare the different methods and discuss the advantages and disadvantages of each. Note that because our simple example of a breaker failure scheme did not require any math functions or calculations, the documentation of math functions is not included as part of our comparison; however, a specific example of documenting math functions as a part of logic will be provided later in Section III.

The written method is excellent because it describes the way the scheme is intended to function. In addition, a written description is easy to generate and does not require much labor or any special software to create. However, without graphics, it is very difficult to visualize logic functioning.

The equivalent dc schematic of our logic is very easy to visualize. You can see in Fig. 3 how the electrical paths that involve contacts and coils are a representation of Relay Word bits and outputs. An energized coil represents the assertion of an output—similarly, the "a" contact changing state would represent a logical point or, as in our example, a Relay Word bit changing state. However, a schematic does require effort to generate; typically another drawing is issued with the settings. In addition, some minimal notes describing the settings used are necessary in order to capture the intended function. In other words, the schematic alone is typically not enough to accurately and clearly capture the intended function of the scheme.

The logic gate representation, like the dc schematic, is very easy to visualize. Logic gates are commonplace in the microprocessor and microcontroller industry and are now typically part of most electrical engineering and technician curricula. For example, in Fig. 6, one can clearly see how the logical states transition from left to right across the diagram. Often, to capture the assertion and deassertion of inputs and outputs in time, a timing diagram may be drawn.

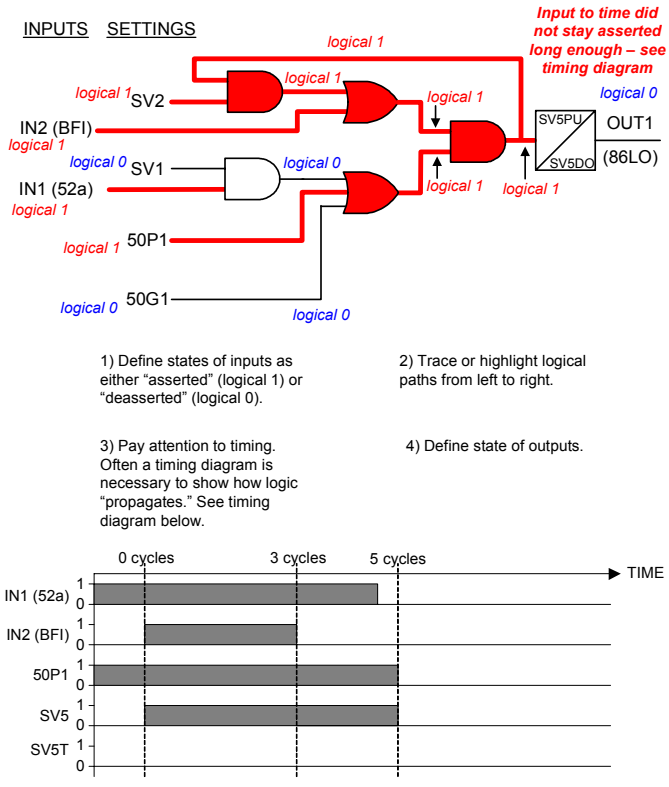


Fig. 6. Visualizing Logical Paths and Timing Diagrams

The logic gate representation, in some cases, requires effort to generate; however, in many cases, software tools are available to either generate a logic diagram based on settings or create logic schemes using the graphical interfaces, as shown in Fig. 6.

Like the dc schematic, often a logic diagram alone is not adequate to clearly represent a system's intended function, and either notes or a separate functional description may be needed. Another disadvantage is that, while the logic gate symbology has been present and very commonplace since the advent of transistor and discrete logic devices, many in the power industry may not be familiar with the symbology. The concepts of Boolean algebra, represented as AND, OR, and NOT gates, may not be as familiar as coils and contacts.

For the ladder logic representation, the scheme can be visualized as well. One disadvantage of representing logic in this form is that the logic is really specific to the PLC device used. Microprocessor-based relays, communications processors, and other logical processors are not easily represented with ladder logic. Another disadvantage is that the ladder logic representation implies an order of processing. Logical evaluation is carried out in time from the top of the ladder logic diagram on the first "rung" to the end "rung." This process is referred to as "the scan." Use caution when representing a scheme as ladder logic, so the order in which the devices evaluate logic match the order specified on the ladder logic diagram. It may be difficult to determine the order in which the logic evaluation occurs in sophisticated relays with separate protection logic and automation logic processes.

The final method, which is to simply print or document the settings, has some disadvantages compared to the other methods. It is difficult for even those familiar with a particular device to visualize logic from a settings printout. Printouts do not give an overview of the scheme as a whole. For example, if a particular protection or control scheme involved multiple devices, this method of documentation would involve trying to piece together printouts from those different devices. Another disadvantage to this method is that it offers very little assurance of how a scheme was intended to function. Although the capability to attach or append comments to settings, similar to commenting in computer programming language, is growing, it still does not capture the intended function of the entire scheme like a written description does.

The printout method has one advantage; often the settings are stored as a computer file. Either the electronic file or printout serves as documentation of exactly how the relay or controller was programmed. The other methods are not directly related to how the device is set. Consequently, any changes in logical settings or programming will necessitate that the drawings (either logic gate, equivalent dc, or ladder logic) be updated and/or their written description revised. The importance of maintaining these documents causes some to argue that the settings files for the device should be all that is used for documentation, and any tools needed to aid in visualizing logic should be included as part of the software. While this is certainly a good point, consider who will have the software. Integration technicians commonly carry computers, but not all relay technicians or plant electricians may have easy access to a laptop with the necessary software. In addition, the visualization tools associated with many devices are typically only compatible with one manufacturer's device. So when there are multiple devices from different manufacturers, even the best software visualization tools will not give a complete picture like a drawing and/or description.

III. Documenting Advanced Functions

Because the breaker failure example did not require any math, we will use another simple example to illustrate how a logic scheme that does require the use of math functions can be documented. In this case, use programmable logic to compare the measured single-phase reactive power and trigger an alarm when the threshold is exceeded [3]. A written description of the logic follows:

> *The reactive power alarm logic should function as follows: The measured reactive power, averaged over a ten-cycle period, should be compared to a threshold setting stored in the numerical analog math variable, AMV054, and should start a timer. If the measured power exceeds the set threshold for longer than 60 cycles, the output contact, OUT101, should assert, and this output will be monitored.*

Similarly, a representation of the logic using logic gates and communications symbology is shown in Fig. 7.

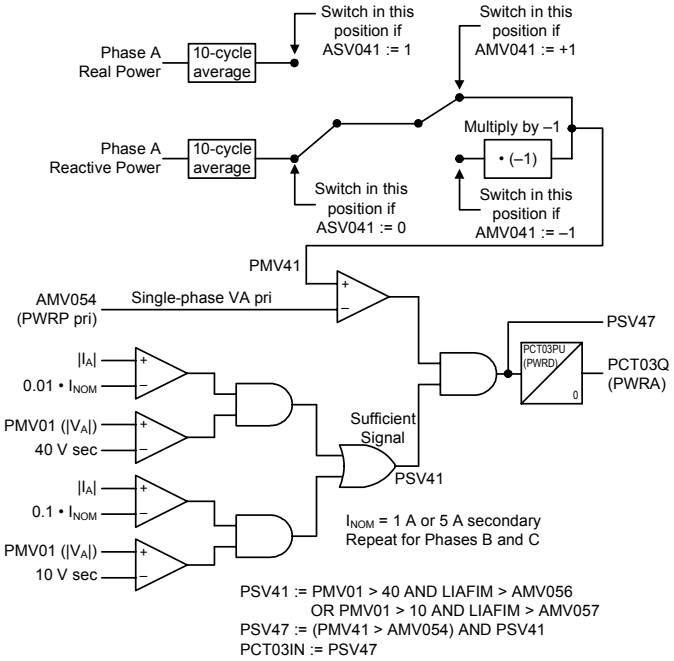


Fig. 7. Power Element Logic Operation

Finally, the printout of the settings, or code, follows:

```
PSV41 := PMV01 > 40 AND LIAFM > AMV056 OR
PMV01 > 10 AND LIAFIM > AMV057
PSV47 := (PMV41 > AMV054) AND PSV41
PCT03IN := PSV47
PCT03PU := 60 # pickup time in cycles
PCT03DO := 0 # dropout time not used
OUT101 := PCT03Q
```

Although math functions are included in this example, the overall conclusions are very similar. The written description still captures the intended function. The graphical representation still allows the settings and scheme to be viewed, and logical paths as well as mathematical operations can be highlighted and tracked, making testing and troubleshooting easier. Finally, the settings, or code, while not a good visual tool, provide a record of how the settings are programmed in the device. Several important aspects of logic that involve math functions include:

- Commenting on both drawings and settings to clarify intended function;
- Ensuring that math errors (division by zero, etc.) are either handled or avoided;
- Making sure that when measured values are used as part of calculations, units are consistent. (Is the quantity given in V or kV?)

IV. Real-World Examples

Because of the lack of proper documentation (discussed above) several cases of misoperations have occurred in the field. This section investigates a few of those misoperations and discusses how proper documentation could have prevented them.

A. Recloser Control Event

A recloser control was configured to protect a feeder. The reclosing element in the control was set to operate with three shots. The trip equation was TR = 50P1 * !SV10 + 51PT + 51NT. Therefore, the control was intended to trip for an instantaneous phase overcurrent element pickup or the timeout of either a phase or neutral inverse-time overcurrent element.

The control was configured with a cold load pickup scheme, using the variable SV10. When the recloser was open for a pre-set amount of time, the instantaneous overcurrent element (50P1) was disabled and the phase inverse-time overcurrent element pickup was adjusted from 3.10 amperes secondary to 35 amperes secondary. The curve and time dial of this element were not affected by the cold load pickup scheme.

A C-G fault occurred on the system, seen correctly by the control, as shown in Fig. 8.

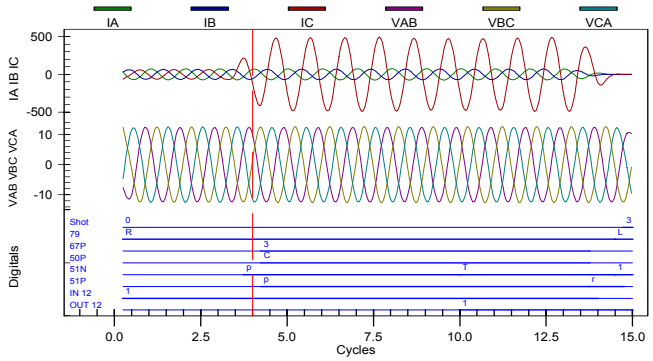


Fig. 8. Recloser Control Event

The event report shows that an instantaneous phase overcurrent element asserted. The "C" above the assertion of the 50P element indicates that the C-phase current exceeded the pickup threshold for one of the instantaneous phase elements in the control. Three separate instantaneous phase overcurrent elements were set in the control, only one of which, 50P1, was set to trip the recloser. Fig. 9 shows a portion of the internal logic in the control associated with the Level 1 instantaneous phase overcurrent element.

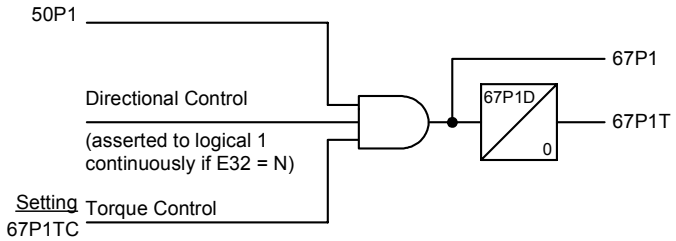


Fig. 9. Instantaneous Directional Overcurrent Logic

The 50P1 element output was combined with directional and torque control inputs to create the output 67P1. Therefore, returning to Fig. 8, we see that the 67P3 element asserted, as indicated by the "3" above the assertion point of the 67P element. If 67P1 had asserted, a "1" would have appeared over the assertion point instead.

However, Fig. 8 shows both the phase and ground inverse-time overcurrent elements asserted. Both elements have a time delay associated with them. The "T" at Cycle 10 over the 51N element indicates that 51N timed out first, causing the trip.

At Cycle 10, the digital OUT 12 asserted as well, indicating that the output contact, OUT101, on the control closed. In the event report, the digital OUT 12 represented both OUT101 and OUT102. A "1" marked the assertion of OUT101, and a "2" marked the assertion of OUT102. When both inputs asserted at the same time, a "b" was placed above the digital OUT 12. The control settings showed that OUT101 = TRIP, which is a digital element in the control that asserts when the TR equation is a logical 1.

Approximately four cycles later, the overcurrent elements deasserted, along with IN101, as shown by the deassertion of the IN 12 digital. The recloser 52A contact was wired to IN101 on the control. Therefore, we know that the recloser opened, and everything to this point operated as expected.

However, Fig. 8 shows that at 14.5 cycles into the event, the reclosing function went directly from the reset state to the lockout state. The control was programmed to make three reclosing attempts before going to lockout, so why did it not do so?

A logic equation in the control initiates reclosing, causing the control to go from the reset state to the cycle state. This transition would be shown in the event report as a "C" for the 79 digital. This did not occur during the event, meaning that reclosing was not initiated. This explains why the control went to the lockout state. Its internal logic, documented in the relay's instruction manual, caused the control to go to lockout when the breaker opened without a reclose initiate command being issued.

Why didn't the control issue a reclose initiate command? The answer is found in the reclose initiate equation for the control: 79RI = 50N1. This is an instantaneous neutral overcurrent element available in the control. Fig. 8 does not show this element because it did not assert. The reason is that the control did not have any instantaneous neutral overcurrent elements enabled.

The reclose initiate command should have been set to TRIP. The inclusion of the instantaneous overcurrent element, 50P1, in the drive-to-lockout equation (79DTL) indicates that the control was expected to enter the cycle state when a trip occurred. The instantaneous element was to be used to force the control into the lockout state after the TRIP initiated reclosing for high-current faults.

In this case, the logic setting was entered incorrectly in the control. It is obvious that the reclosing element was not tested during the control's commissioning, as this error would have been caught. Had the logic been properly documented, it would have been clear that reclosing needed to be tested to completely commission the control.

A documentation scheme that would have assisted in the commissioning of this control follows:

> *The control is set as a three-shot recloser. When a trip occurs, the relay should enter the reclosing cycle state. The control uses setting 79RI, set to TRIP, as a trigger to initiate reclosing. The initiation is supervised by the recloser status, which is connected to IN101.*
>
> *Once initiated, the first open-interval timer will begin timing once the breaker is open (IN101 deasserts) and no current is flowing (51P, 51N, and TRIP have deasserted). The open-interval times are 30, 120, and 300 cycles for the first, second, and third open intervals, respectively.*
>
> *The control will automatically go to the lockout state if IN103 deasserts (connected to a 79 ON/OFF switch), 50P1 asserts, or a manual open occurs (via a serial port* ***OPEN*** *command).*
>
> *The reset time is 300 cycles for the control to enter the reset state from either the cycle or the lockout state.*

If this document had been available to the crew members who commissioned the control, they would have known that reclosing was being used, and they would have understood how it should operate. As a result, they would have known that it was being initiated with a logic equation and that the logic must be tested for accuracy. Had they tested it, they would have found the settings error, and this misoperation would not have occurred.

B. Line Relay Event

This relay served as backup protection on a 69 kV line. The relay had phase distance elements, one for phase-to-phase and one for three-phase faults and neutral overcurrent elements, for ground faults, set to provide backup protection for the adjacent line. An A-G fault at approximately 95 percent of the line produced the event shown in Fig. 10.

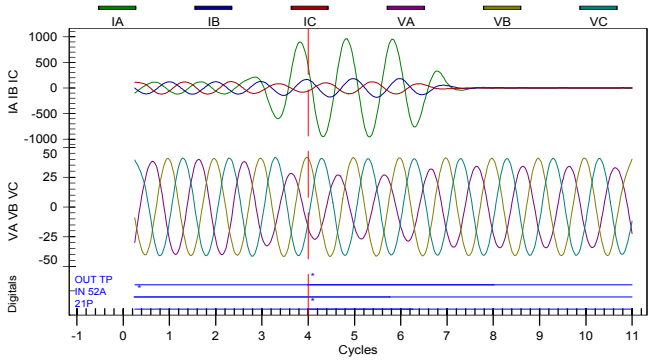


Fig. 10. A-G Fault at 95 Percent of the Line

When protecting a line with distance elements, multiple zones are required. Zone 1 is typically set to see 80 percent of the line impedance. This element should operate instantaneously. Zone 2 is set to see past the remote line terminals, providing coverage for the remaining 20 percent of the line plus backup protection for the adjacent line. This relay should be set with a time delay in order to coordinate with the Zone 1 relay on the adjacent line.

The event was triggered at Cycle 4, when the phase-to-phase distance element, 21P, asserted. Note two problems with this event: The phase distance element asserted for a line-to-ground fault, and the relay tripped instantaneously.

The assertion of the phase distance element for a phase-to-ground fault can be explained by a known behavioral characteristic of some phase distance elements when they are ex-

posed to certain fault conditions. Phase distance elements, such as the one in this relay, may operate for a close-in line-to-ground fault with high fault resistance if the relay is applied on a line with a strong source [4] [5].

From the settings, the source impedance ratio (SIR) was 0.2, indicating a strong source. The reach of the phase distance element was set to 348 percent of the line impedance to allow the relay to cover the adjacent line. The fault occurred at approximately 95 percent of the line, making it a relatively close-in fault. The final condition of high fault resistance was also met, as a boom truck caused the fault. Therefore, considering the conditions, the operation of the phase distance element is not a surprise. But the element operation explanation still does not account for the instantaneous trip. The relay was supposed to provide backup for the adjacent line and, hence, should have had a time delay. The setting of 100 cycles, or 1.67 seconds, shows that a time delay was intended for this element. So why was there an instantaneous trip?

The relay logic must be investigated to determine the answer. The trip logic equation, MT, was set to 66 hexadecimals. The logic was set using a mask, where the eight bits corresponded to eight elements in the relay. Fig. 11 shows the elements in the relay and which elements are included in a 66-hexadecimal mask.

50L	ZABC	ZP	ZPT	67NP	67NT	67NI	67DT
0	1	1	0	0	1	1	0
6				6			

Fig. 11. MT Logic Mask 66 Hexadecimal

A mask of 66 hexadecimals caused the assertion of ZABC, ZP, 67NT, and 67N1 to issue a **TRIP** command. These elements are described in Table I.

In this event, the ZP element asserted and issued the trip. This is the pickup of this element and not the time delay output, ZPT. Therefore, any time the phase-to-phase distance element asserted, the relay issued an instantaneous trip.

This event proves that the logic was not understood and, thus, not tested correctly during commissioning. Proper documentation would have made the logic more comprehensible and simplified testing.

In this case, a logic diagram would have made it clear that the phase-to-phase element would trip the breaker without a time delay. Fig. 12 shows a logic diagram for the MT logic equation.

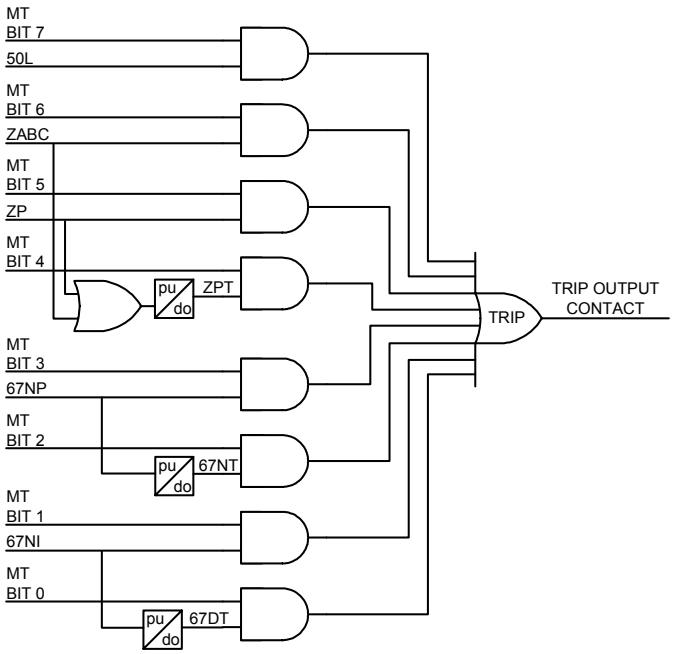


Fig. 12. MT Logic Diagram

Table I gives a description of each element shown in Fig. 12. This table can be found in the relay's instruction manual.

TABLE I:
RELAY ELEMENT BIT DESCRIPTIONS

50L	Phase fault current supervision
ZABC	Three-phase mho element
ZP	Phase-to-phase mho element
ZPT	Phase or three-phase fault timeout
67NP	Residual time-overcurrent element
67NT	Residual time-overcurrent trip
67NI	Residual instantaneous overcurrent (directional or nondirectional)
67DT	Definite-time ground timeout

To determine which elements are enabled, use the setting MT, and convert from hexadecimal to binary. The instruction manual for this relay has a table similar to Table II. Some calculators also provide this functionality. Then match the bits to the elements, with the 0 bit on the right-hand side.

TABLE II:
HEXADECIMAL TO BINARY CONVERSION

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

C. Fast Bus Trip Scheme Event

A fast bus trip scheme is a simple, inexpensive method of protecting a bus. The scheme uses the main and feeder relays that already exist to also protect the bus. The system configuration for this event is shown in Fig. 13.

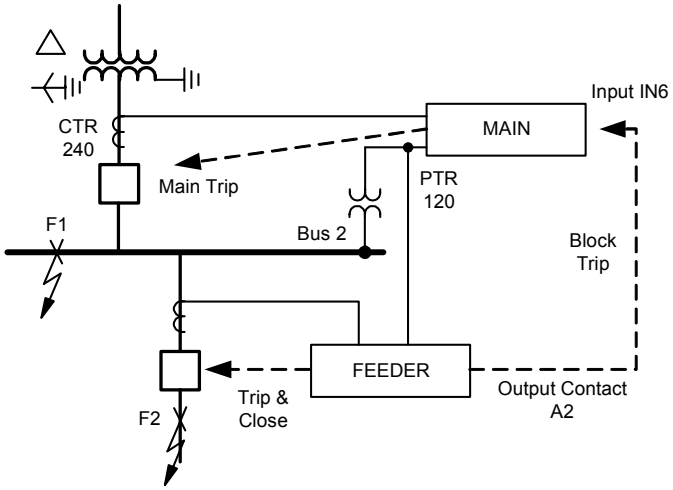


Fig. 13. Fast Bus Trip Scheme One-Line Diagram

For a fault at F2, the feeder relay detects fault current and sends a blocking signal to the main relay. The main relay is set with a small delay to allow time for the block trip signal to arrive.

For a fault at F1, the feeder relay does not detect any fault current. Therefore, it does not send the blocking signal to the main relay, allowing the main relay to trip with a small time delay for bus faults. The use of this blocking signal provides fast clearing times for bus faults, where coordination of time overcurrent elements would further delay the main relay trip.

The feeder relay in this event is set with a trip equation of TR = 51T + 51NT. The elements in this equation correspond to the timeout of the phase and ground inverse-time overcurrent elements, respectively. The relay is also programmed with output contact A2 to send the blocking signal. The logic equation for this output is A2 = 50L + 50NL, which are phase and ground instantaneous overcurrent elements, respectively. These elements are set to match the fast bus trip scheme elements in the main relay.

The main relay is set as follows: TR = 51T + 51NT + V, where 51T and 51NT provide backup protection to the feeder. The V is a logic element programmed for the fast bus trip scheme. It is equal to E*!L, where E = ST and L = IN6. ST is the timeout of logic variable timer S, which is equal to 50NH + 50H. The timer is set with a three-cycle pickup delay. IN6 is the input wired to receive the blocking signal from the feeder relay.

For this event, a fault occurred on the feeder, but the main relay tripped. Why?

Both relays detected the fault and captured event reports. Fig. 14 shows the event reports from both relays combined, where Event 1 is from the main relay and Event 2 is from the feeder relay.

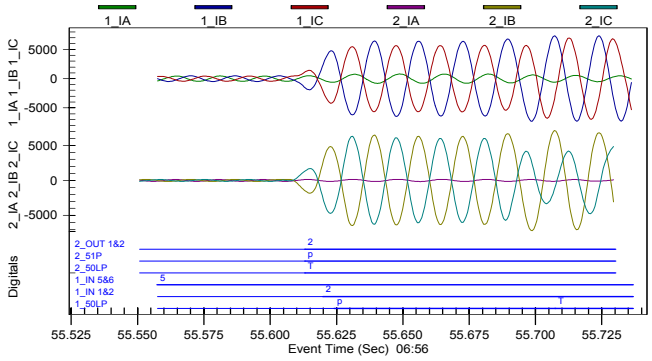


Fig. 14. Fast Bus Trip Scheme Event

The event reports confirmed that the fault was on the feeder. Fig. 14 shows that the feeder relay 51P element was timing to trip. The fast bus trip scheme in the feeder relay also functioned as expected. The 50L element asserted at the same time as the 51P element, and as a result, OUT2 asserted, sending the block signal to the main relay.

On the main relay, IN2 asserted approximately 5 ms later. However, the main relay monitored IN6 for the block input signal, which did not assert, leading to the fast trip. The breaker opened approximately 50 ms after the end of this event.

Because the scheme logic was not properly documented, the commissioning of these relays did not involve testing the entire scheme, allowing this wiring error to go undetected.

In this case, a dc schematic that included the logic inside the relays would have assisted in commissioning this scheme. Fig. 15 shows a dc schematic for this case.

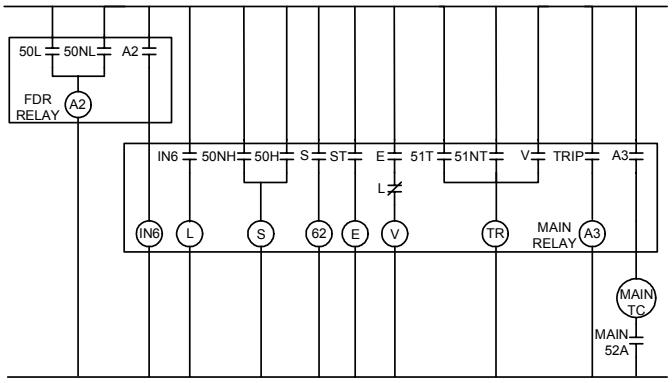


Fig. 15. Fast Bus Trip Scheme DC Schematic

Seeing the logic in the relay combined with the wiring between the relays would have made testing this scheme simple. Had the scheme been fully tested, instead of only individual elements, this error would have been discovered.

D. Ground Distance Element Testing

This example occurred while testing the ground distance element of a line protection relay. The relay did not have a breaker status input wired to it. The phase distance and ground overcurrent elements tested as expected, but not the ground distance element. The technician applied a fault for over one minute, and the instantaneous Z2G element did not operate.

Initially, the lack of breaker status to the relay was thought to be the cause. Investigation of the Z2G logic showed that the Z2G element is blocked if the relay detects a three-pole open (3PO) condition. The A-phase Z2G logic also requires either SPO (single-pole open condition) or FSA (A-phase-to-ground fault-detection logic) to assert. Additionally, SPOA (A-phase single-pole open condition) must remain deasserted. These are the only elements used in the Z2G logic that are connected to the breaker status. Fig. 16 shows the Z2G logic diagram from the relay's instruction manual.

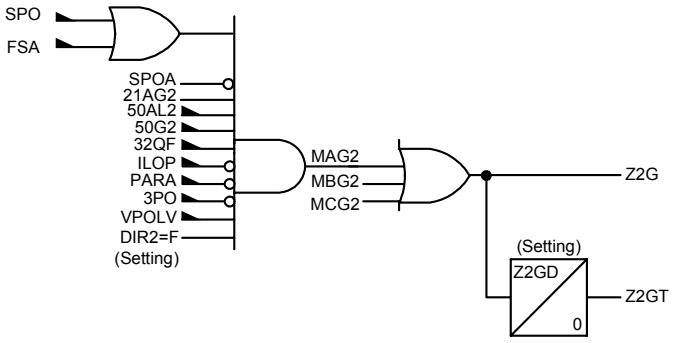


Fig. 16. Z2G Logic Diagram

Does the pole-open logic require the breaker status to operate correctly? Fig. 17 shows the logic diagram for the relay's pole-open logic.

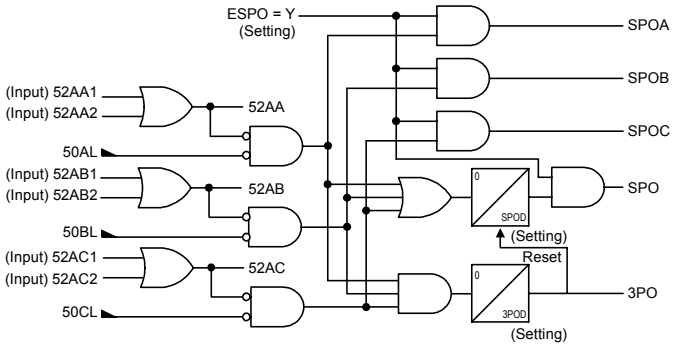


Fig. 17. Pole-Open Logic Diagram

The diagram shows that both the SPO and the 3PO conditions are determined from the same inputs. For 3PO to assert, the breaker status must be deasserted for each phase, and the current must be below the pickup of the phase overcurrent element 50L on all three phases, indicating that no current is flowing. With no breaker status input to the relay, the 3PO logic operates only on current input. The single-pole logic operates in the same fashion, but on a per-phase basis. Thus the logic can operate correctly without a breaker status input.

The final piece of this logic diagram is a dropout timer. Once current returns and 50L asserts, 3PO remains asserted for 3POD cycles. This timer accounts for pole scatter that may exist between the breakers on the three phases. A typical setting is three cycles. In this relay, 3POD was set to the maximum of 8,000 cycles or 2.22 minutes.

Because the current was stopped between tests, the 3PO element asserted. When the new test started, the element was blocked for 2.22 minutes. The other elements that tested correctly were not blocked by the 3PO element.

In this example, the erroneous setting resulted from not understanding how the relay operates. Figs. 16 and 17 show that the logic was properly documented in the relay's instruction manual. Therefore, this example shows the importance of using the logic documentation provided. Had the above logic diagrams been used when setting the relay, the settings error could have been avoided.

V. Conclusions

As the logic required to implement protection and control schemes is being transferred from physical wiring and auxiliary relays to programming inside microprocessor-based devices, it is extremely important that standards are developed on the documentation of this logic.

The complexity of the logic, as well as the diversity of the functions being performed, is continually increasing. This is causing setting, testing, and troubleshooting schemes and devices to become more difficult, and as a result, sections of logic are missed in testing, and incorrect settings are entered. Therefore, in order to simplify the understanding of logic schemes and to ensure complete and accurate testing, it is essential that companies develop standards for documenting logic that allow the schemes to be understood and visualized.

This paper has presented several methods used to document logic. In some cases, a combination of the methods discussed may be required.

VI. References

[1] *IEEE Standard Electrical Power System Device Function Numbers and Contact Designations*, IEEE Standard C37.2, 1996.

[2] Bill Fleming, "Implementing a Breaker Failure Function in the SEL-351 Relay," SEL Application Guide AG98-07. Available at www.selinc.com/aglist.htm.

[3] Jacob Reidt, "Implementation of Voltage, Frequency, and Power Elements in the SEL-451 Relay Using SELOGIC Control Equations," SEL Application Guide AG2004-12. Available at www.selinc.com/aglist.htm.

[4] Jeff Roberts and Edmund O. Schweitzer, III, "Distance Relay Element Design," Schweitzer Engineering Laboratories, Inc. Available at www.selinc.com/techpprs.htm.

[5] Edmund O. Schweitzer, III, "New Developments in Distance Relay Polarization and Fault Type Selection," Schweitzer Engineering Laboratories, Inc. Available at www.selinc.com/techpprs.htm.

VII. Biographies

Jason Young graduated from the University of Waterloo in 2006 with a BASc in Electrical Engineering. In 2004, he joined Ontario Power Generation as a protection and control engineering intern. He also worked as a planning technician intern for Hydro One Networks. He joined Schweitzer Engineering Laboratories, Inc. in August 2006, where he serves as an associate field application engineer in Boerne, Texas.

Derrick Haas graduated from Texas A&M University in 2002 with a BSEE. He worked as a distribution engineer at CenterPoint Energy in Houston, Texas, from 2002 to 2006. In April 2006, Derrick joined Schweitzer Engineering Laboratories, Inc. and works as a field application engineer in Houston, Texas.

Previously presented at the 2008 Texas A&M
Conference for Protective Relay Engineers.

20071025 • TP6301-01