Chapter 9

Boolean Logic Examples

Boolean Shorthand

! = NOT function
+ = OR function
 * = AND function
Pr = Prior State
    (prior communication cycle)
B = Blue Loop
    (input and output modules can be connected to this loop)
Y = Yellow Loop
    (input and output modules can be connected to this loop)
V = Virtual Loop
T XX enable = Timer Enable Loop
T XX done = Timer Done Loop
TXX run = Timer Run Loop

Boolean Logic Examples

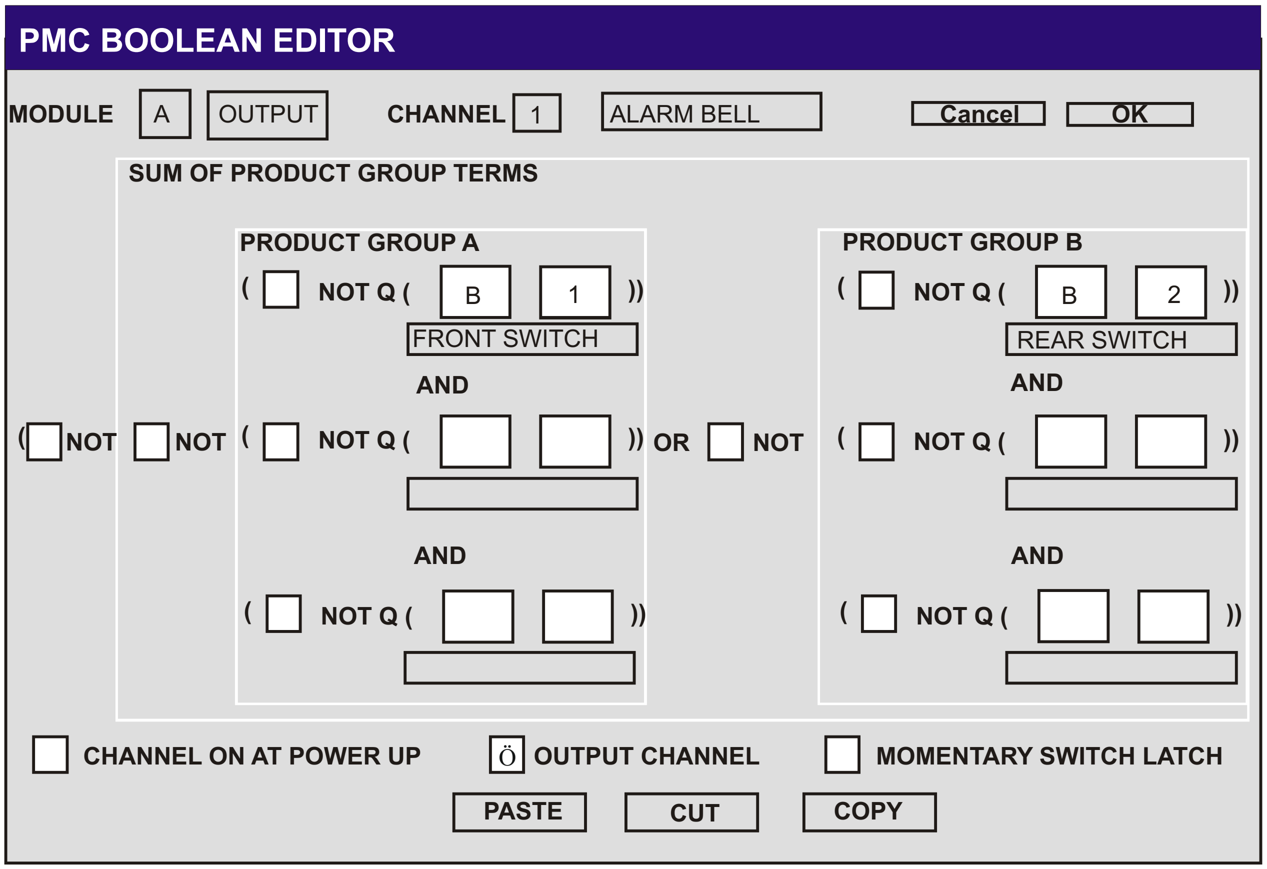
Simple "OR"

In this example, Load A1 will be ON, if input B1 or B2 is ON, or if both are ON. B1 and B2 could be any two inputs from any input channel in the PMC system. A possible application would be an alarm bell that would ring if either one switch, or both switches are ON.

Alarm Bell equals Front Switch or Rear Switch

A1 = ALARM BELL OUTPUT
B1 = FRONT SWITCH
B2 = REAR SWITCH

Formula: A1 = B1+B2



MODULE: A OUTPUT — CHANNEL: 1 — ALARM BELL

PRODUCT GROUP A: (NOT Q (B 1)) FRONT SWITCH

OR

PRODUCT GROUP B: (NOT Q (B 2)) REAR SWITCH

Ö OUTPUT CHANNEL

Boolean Logic Examples

Multiple Input "OR"

This example will work with either the 160 channel CPU, or the 320 channel CPU. In the 320 channel CPU, you will find an AND/OR icon that will change the Boolean Editor screen to allow easy programing for multiple OR's. This technique is used any time 3 or more inputs need to be Or'ed.

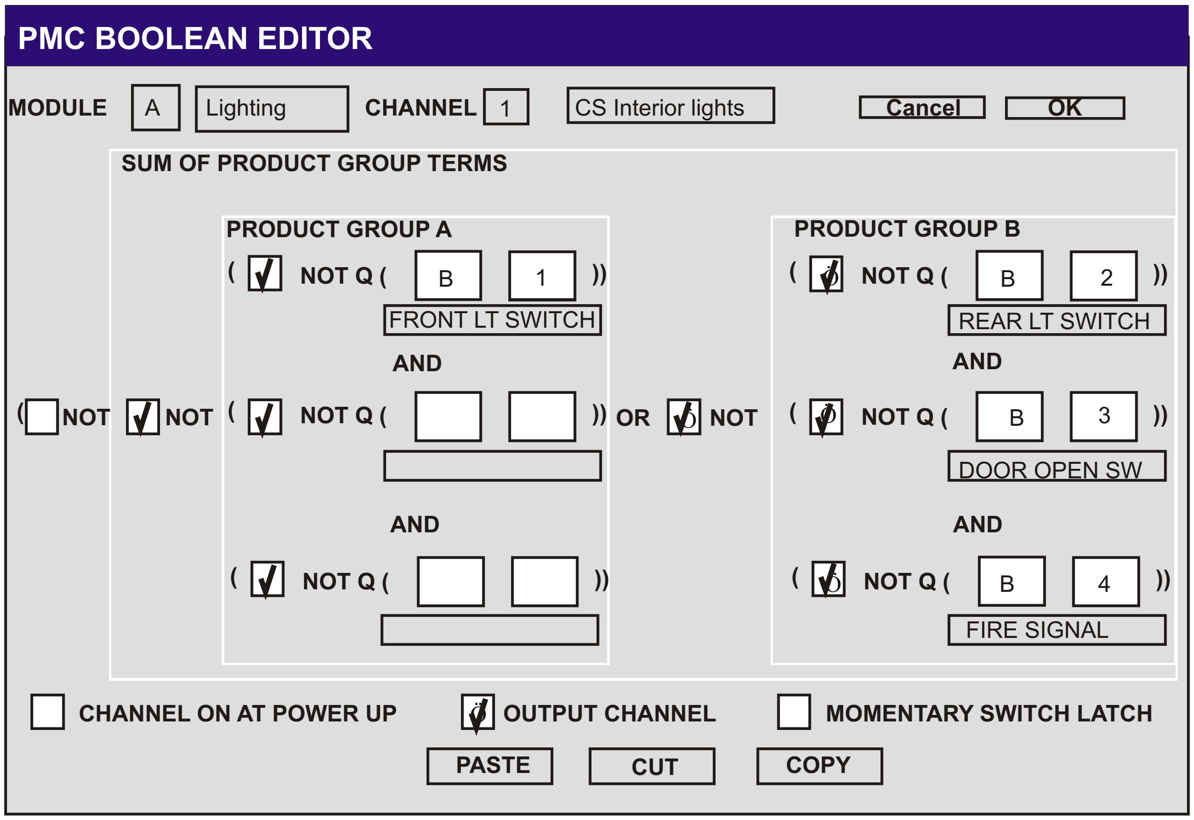
In this example, Load A1 will be ON, if any of the inputs B1, B2, B3, or B4 is ON, or if any combination is ON. B1, 2, 3, and 4 could be any four inputs from any input channel in the PMC system.

A1 = CS INTERIOR LIGHTS
B1 = FRONT LIGHT SWITCH
B2 = REAR LIGHT SWITCH
B3 = DOOR OPEN SWITCH
B4 = FIRE SIGNAL

Formula: **A1 = B1+B2+B3+B4**
Or in other terms
A1 = B1+!(!B2*!B3*!B4)

For a detailed description of the logic behind this function please refer to the DeMorgan's Theorems in Chapter 11.

The screen shown below, "NOTs" the inputs and outputs of B2, B3, and B4 to create the "OR" function from the "AND" function. If more than 6 inputs must work in an "OR" relationship, a virtual channel (Q module) can be used.



PMC BOOLEAN EDITOR

MODULE A Lighting CHANNEL 1 CS Interior lights

SUM OF PRODUCT GROUP TERMS

PRODUCT GROUP A
([✓] NOT Q (B 1)) FRONT LT SWITCH
AND
([ ] NOT [✓] NOT ([✓] NOT Q ( )) OR [✓] NOT
AND
([✓] NOT Q ( ))

PRODUCT GROUP B
([✓] NOT Q (B 2)) REAR LT SWITCH
AND
([✓] NOT Q (B 3)) DOOR OPEN SW
AND
([✓] NOT Q (B 4)) FIRE SIGNAL

[ ] CHANNEL ON AT POWER UP [✓] OUTPUT CHANNEL [ ] MOMENTARY SWITCH LATCH

Boolean Logic Examples

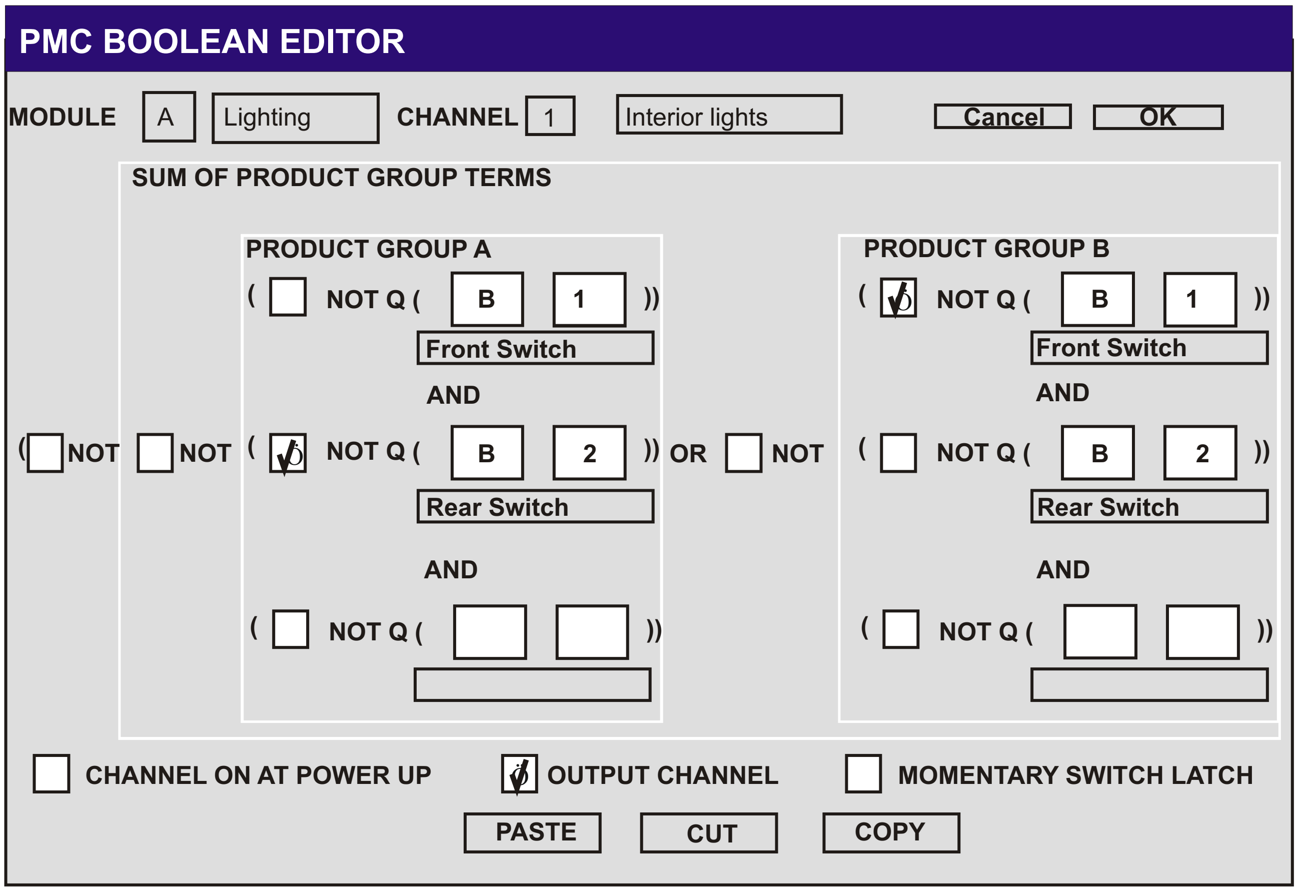
Exclusive “OR”
Two or more switches operate the same load

In this example, load A1 will be ON, if input B1 or B2 is ON, but *not* both. B1 and B2 could be inputs from any input module, or input channel in the PMC system. This example is used with either the 160 or 320 channel CPU. The loop designation must be selected with the 320 channel CPU.

The exclusive “OR” could be used for a light switch at each end of a vehicle. Either switch will turn the light on, or OFF. For more than two switches in an exclusive “OR” arrangements, write an exclusive “OR” in a virtual channel, then, write another exclusive “OR” using the virtual channel. This would put 3 inputs in a exclusive arrangement.

Formula: **A1 = (B1*!B2)+(!B1*B2)**

A1 = INTERIOR LIGHTS
B1 = FRONT SWITCH
B2 = REAR SWITCH



PMC BOOLEAN EDITOR

MODULE	A
	Lighting
CHANNEL	1
	Interior lights

Cancel   OK

SUM OF PRODUCT GROUP TERMS

(☐ NOT ☐ NOT

PRODUCT GROUP A			
(☐ NOT Q (	B	1	))
	Front Switch		
AND			
(✓ NOT Q (	B	2	))
	Rear Switch		
AND			
(☐ NOT Q (			))

OR ☐ NOT

PRODUCT GROUP B			
(✓ NOT Q (	B	1	))
	Front Switch		
AND			
(☐ NOT Q (	B	2	))
	Rear Switch		
AND			
(☐ NOT Q (			))

☐ CHANNEL ON AT POWER UP   ✓ OUTPUT CHANNEL   ☐ MOMENTARY SWITCH LATCH

PASTE   CUT   COPY

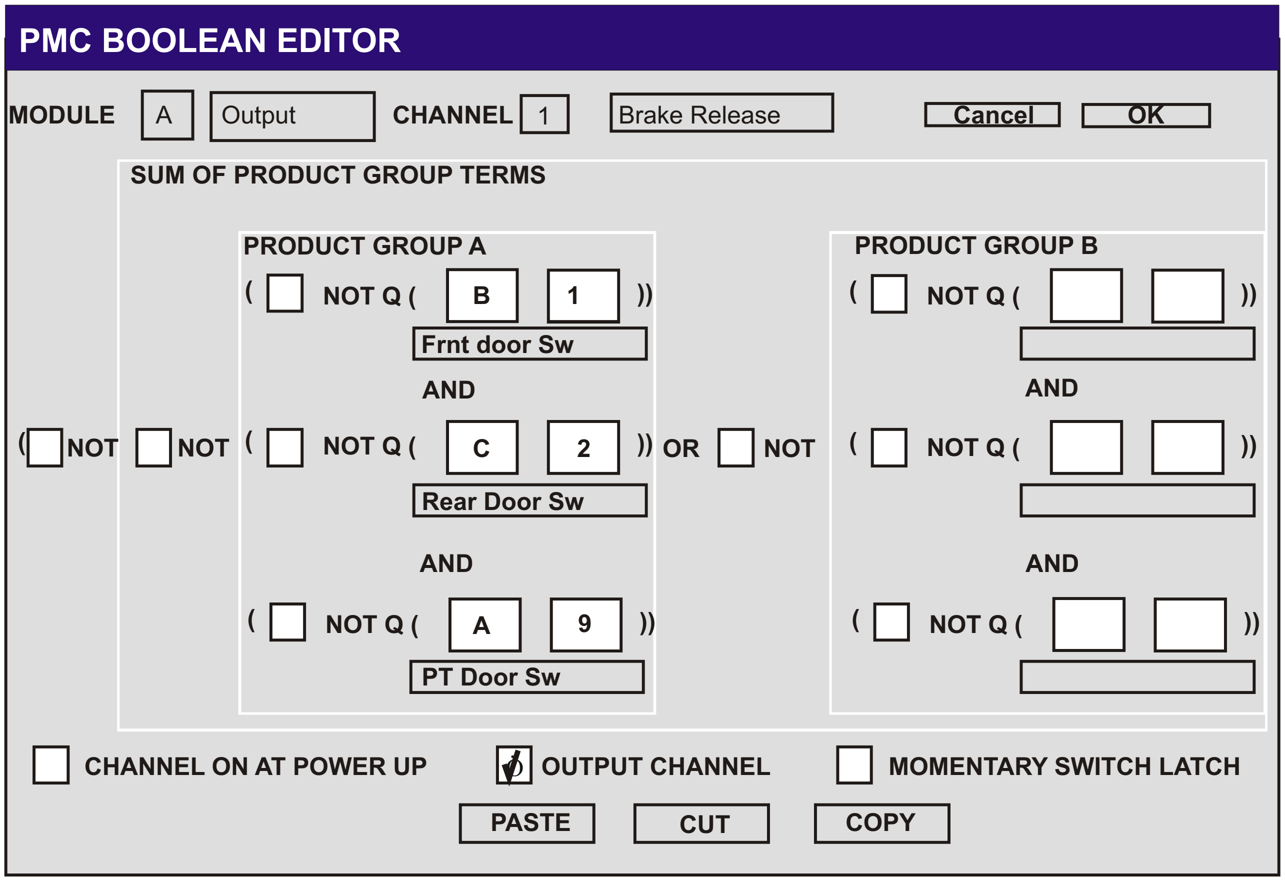
Boolean Logic Examples

Simple "And"

In this example, load A1 will be ON, if input B1 and C2 and A9 are ON. B1, C2, and A9 could be any three inputs from any 3 input channels in the PMC system. A possible application would be 3 door switches that must be closed to release the park brake. C

If more than 3 inputs need to be in an "AND" relationship *see DeMorgan's Theorems in Chapter11,* or use a channel from the virtual module Q.

Formula:	**A1 = B1*C2*A9**
	A1 = OUTPUT
	B1 = FRONT DOOR CLOSED SWITCH
	C2 = REAR DOOR CLOSED SWITCH
	A9 = PT DOOR CLOSED SWITCH



PMC BOOLEAN EDITOR

MODULE: A — Output; CHANNEL: 1 — Brake Release

SUM OF PRODUCT GROUP TERMS

PRODUCT GROUP A			PRODUCT GROUP B		
(☐ NOT Q (	B	1))	(☐ NOT Q (		))
Frnt door Sw					
AND			AND		
(☐ NOT Q (	C	2))	(☐ NOT Q (		))
Rear Door Sw					
AND			AND		
(☐ NOT Q (	A	9))	(☐ NOT Q (		))
PT Door Sw					

(☐ NOT ☐ NOT (Product Group A) OR ☐ NOT (Product Group B)

☐ CHANNEL ON AT POWER UP   ✓ OUTPUT CHANNEL   ☐ MOMENTARY SWITCH LATCH

Boolean Logic Examples

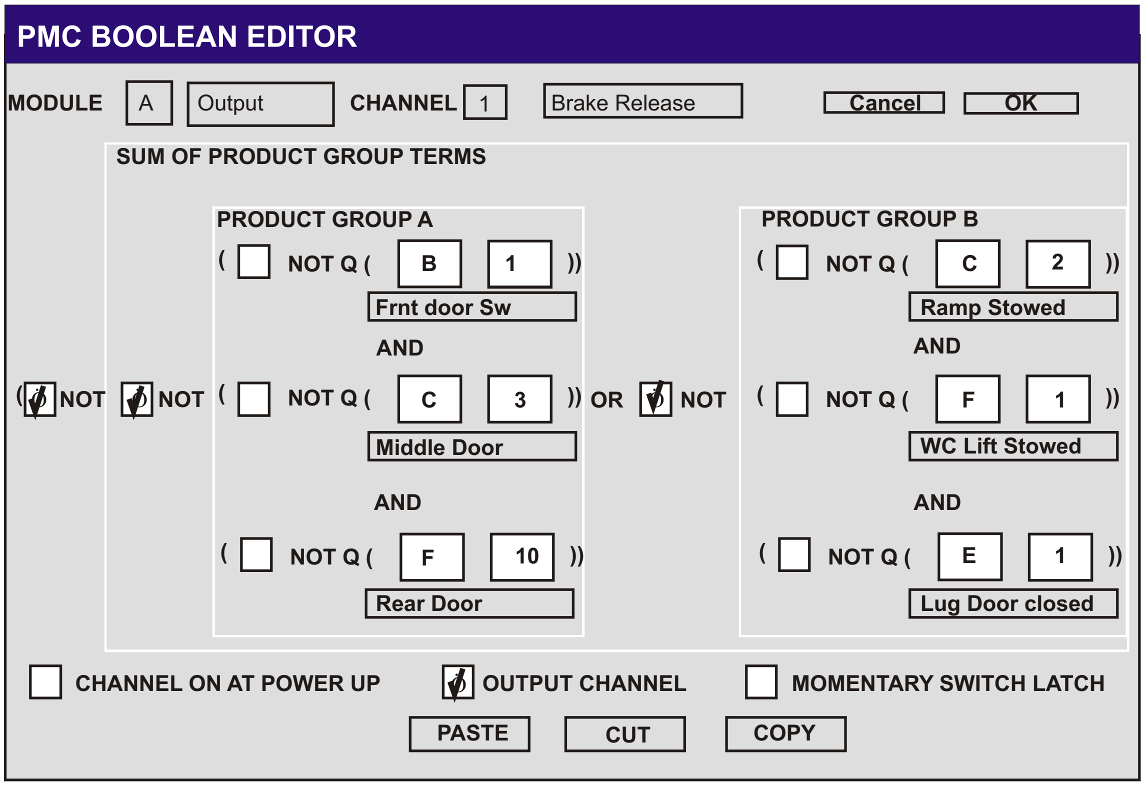
SIX INPUT "AND"

Looking at the Boolean Editor screen, it would appear that a 3 input "AND" function would be the most that is possible. Using DeMorgan's theorem, we can create a 6 input "AND", without using a virtual module. This technique works with either the 160 or 320 channel CPU. With the 320 channel CPU, simply open the 6 input AND editor screen using the AND /OR icon.

A1 = B1*C2*C3*F1*F10*E1
or in other terms
A1 = !(!(B1*C3*F10)+!(C2*F1*E1))
A1 = !(!(B1*C3*F10)+!(C2*F1*E1))

A1 = OUTPUT
B1 = FRONT DOOR CLOSED SWITCH
C2 = RAMP STOWED SWITCH
C3 = MIDDLE DOOR CLOSED SWITCH
F1 = WHEEL CHAIR LIFT STOWED
F10 = REAR DOOR CLOSED SWITCH
E1 = LUGGAGE DOOR CLOSED SWITCH
Any input from any module may be used in the ***And*** *function*

For a detailed description of the logic behind this function please refer to DeMorgan's Theorems in Chapter11. The screen shown below "NOTs" all of the outputs to create the AND function. If more than 6 inputs must work in an "AND" relationship, a virtual channel (Q module) can be used.



PMC BOOLEAN EDITOR

MODULE A Output CHANNEL 1 Brake Release

SUM OF PRODUCT GROUP TERMS

PRODUCT GROUP A
(☐ NOT Q (B 1))
Frnt door Sw
AND
(✓ NOT ✓ NOT (☐ NOT Q (C 3)) OR ✓ NOT
Middle Door
AND
(☐ NOT Q (F 10))
Rear Door

PRODUCT GROUP B
(☐ NOT Q (C 2))
Ramp Stowed
AND
(☐ NOT Q (F 1))
WC Lift Stowed
AND
(☐ NOT Q (E 1))
Lug Door closed

☐ CHANNEL ON AT POWER UP ✓ OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH

Boolean Logic Examples

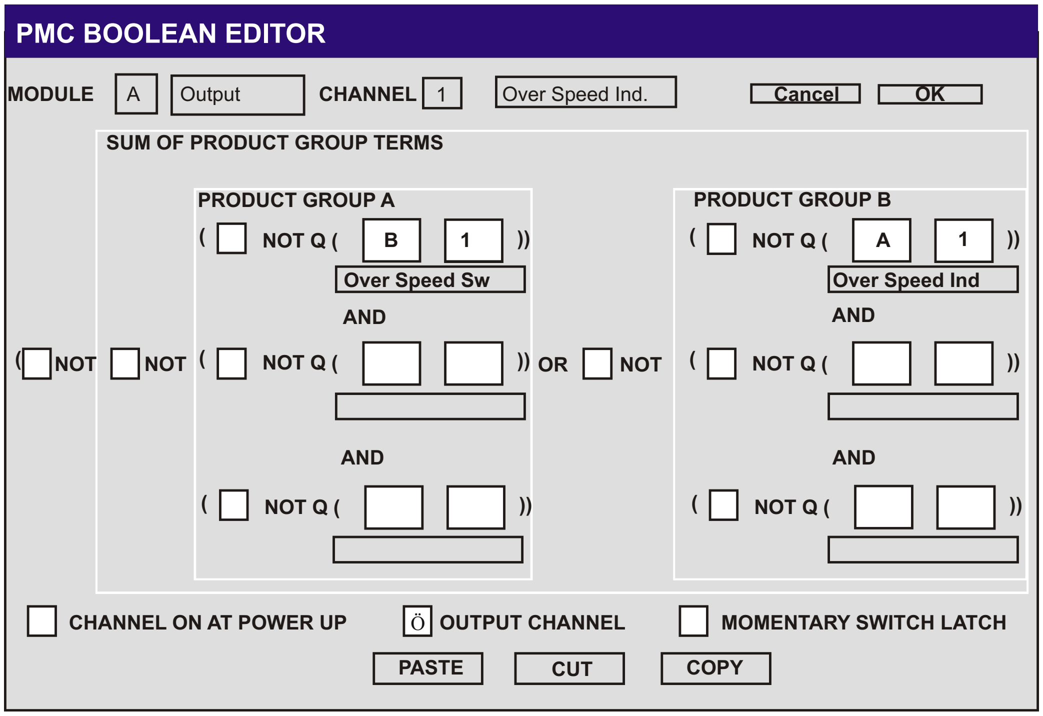
LATCHED OUTPUT

A1 = OVER SPEED INDICATOR LAMP OUTPUT
B1 = OVER SPEED SWITCH
B2 = LATCH CLEAR SWITCH

In this example, once the output turns ON it will remain on. The latched output is often used as a part of other functions, such as timed on delays and flashers. You may choose to latch an output ON until a service technician, or timer pulse clears it.

A1=B1+A1

The output A1 will be ON when switch B1 is ON, or when A1 (itself) is ON. This statement will cause output A1 to latch ON. There is however, no way to turn it off.



PMC BOOLEAN EDITOR

MODULE A Output CHANNEL 1 Over Speed Ind.

SUM OF PRODUCT GROUP TERMS

PRODUCT GROUP A: (☐ NOT Q (B 1)) Over Speed Sw

PRODUCT GROUP B: (☐ NOT Q (A 1)) Over Speed Ind

☐ CHANNEL ON AT POWER UP Ö OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH

A1 = B1+(A1*!B2)

In this example, output A1 will latch ON when switch B1 is pressed momentarily and switch B2 is OFF. The statement reads, A1 equals B1 OR (A1 and not B2). If B1 and B2 were momentary switches, the load A1 would turn on when B1 is closed and would latch ON because A1=A1. Once A1 is latched ON, it will turn OFF when B2 comes ON. This happens because the statement (A1 AND NOT B2) is false.

Boolean Logic Examples

FLASHER
160 Channel CPU

160 Channel CPU
R1 = TIMER enable channel
B1 = SWITCH INPUT
A3 = FLASHING LAMP OUTPUT
S1 = DONE PULSE OF TIMER R1

```
R1 = B1
          Group A       Group B
A3 = (!A3*S1*B1)+(A3*!S1*B1)
```

A timer is used to create a flasher. "Timer enable channel R1" is activated by switch B1. When switch input B1 is turned ON, timer R1 is running.

By clicking "PMC SET UP" and selecting "Timer Set UP" timer channels 1-10 are shown. Enter the flasher time you would like in the column marked reload. In this example the timer is set for 0.3 seconds.

The light will be ON for .3 seconds and OFF for .3 seconds.

In this example, module A, output channel 3, is being used for the flasher output. S1 is the timer done output for timer R1. A pulse occurs on output S1 at the end of each time period. *These pulses continue at the 0.3 second interval as long as the timer is enabled.* In this example, an exclusive "OR" function is used. The output A3 may be ON only when A3, or S1 is on, but *not* both. When A3 turns ON, the boolean in Group A causes output A3 to latch on itself. When the time pulse from S1 comes on again, the half of the boolean in Group B causes A3 to turn OFF. This means that the output will latch ON and OFF with each pulse that occurs on S1.

We have included an "AND" function with the switch B1 to ensure that when the switch is OFF the light will be OFF. If this were not done, the light might remain ON depending upon when the switch B1 was turned OFF.

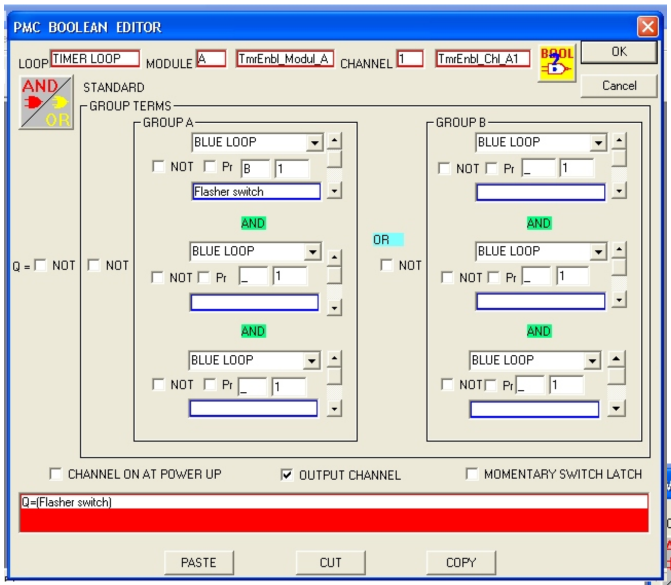
An example using the 320 channel system is shown on the following page.

Boolean Logic Examples

FLASHER
320 Channel CPU

320 Channel CPU
TA1 = Timer
BB1 = Switch Input
BA3 = Flashing Lamp Output
T done A1 =Done Pulse of Timer A1

T enable A1=BB1
BA3=(BA3 * !T done A1 * BB1)+(!BA3* T done A1 * BB1)



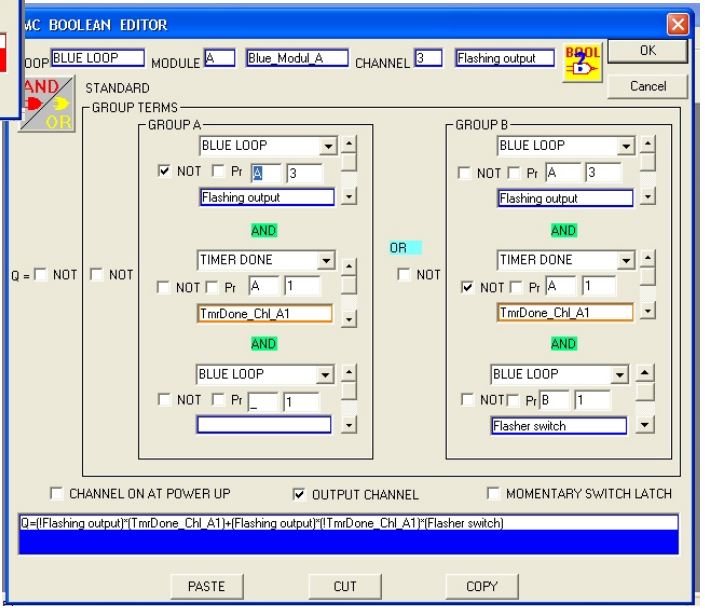
LOOP TIMER LOOP MODULE A TmrEnbl_Modul_A CHANNEL 1 TmrEnbl_Chl_A1

GROUP A: BLUE LOOP, Pr B 1, Flasher switch

☐ CHANNEL ON AT POWER UP ☑ OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH

Q=(Flasher switch)

Enables the timer



LOOP BLUE LOOP MODULE A Blue_Modul_A CHANNEL 3 Flashing output

GROUP A: BLUE LOOP, ☑ NOT, Pr A 3, Flashing output AND TIMER DONE, Pr A 1, TmrDone_Chl_A1 AND BLUE LOOP

GROUP B: BLUE LOOP, Pr A 3, Flashing output AND TIMER DONE, ☑ NOT, Pr A 1, TmrDone_Chl_A1 AND BLUE LOOP, Pr B 1, Flasher switch

☐ CHANNEL ON AT POWER UP ☑ OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH

Q=(!Flashing output)*(TmrDone_Chl_A1)+(Flashing output)*(!TmrDone_Chl_A1)*(Flasher switch)

Latches the output on and off with each timer done pulse

Boolean Logic Examples

OFF DELAY
(INTERIOR LIGHT DELAY) 160 CH CPU

An off delay may be used to turn a load OFF some time period after an event. For example, turn a light off 10 seconds after a door closes.

In this example, load A will turn ON immediately after switch B1 is turned ON. When B1 is turned OFF the light will remain ON for 10 seconds and then turn OFF.

Q1 = Q2
Q2 = B1
A1 = T1+B1+Q1
R1 = (!Q2*Q1) + (T1*!B1)
SET TIMER R1=10 SECONDS

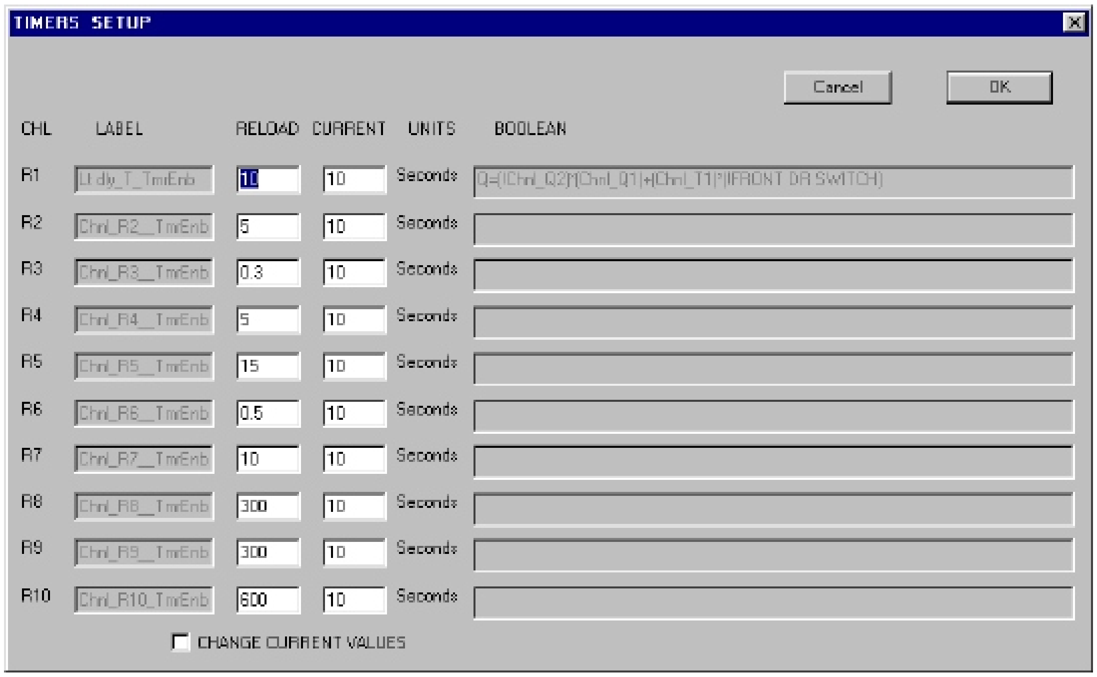
Q1 AND Q2 ARE VIRTUAL MODULES
A1 = OUTPUT
R1 = TIMER
T1 = THE RUN OUTPUT OF TIMER R1
B1 = SWITCH

The statement Q1=Q2 and the statement Q2=B1, creates a trigger pulse that occurs when the switch B1 is opened. This is also referred to as **edge detection**.

The load A1 will be ON when T1, or B1 or Q1 is ON. Q1 is necessary to prevent the output A1 from flashing during the time the switch is turned OFF and when the timer takes over.

The timer R1 will run when Q2 and Q1 are on, or when T1 is on and B1 is off. B1 is added to ensure that the timer is reset if the door is opened and closed during the timer delay.

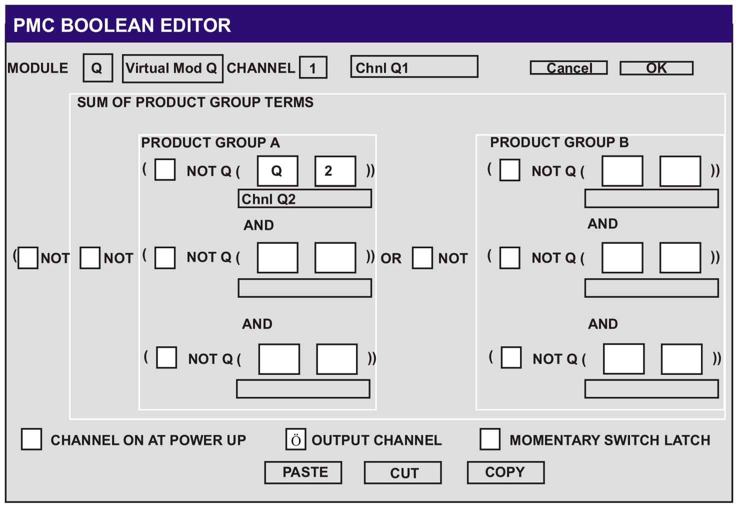
For further clarification see the following boolean screens.



CHL	LABEL	RELOAD	CURRENT	UNITS	BOOLEAN
R1	Lt dly_T_TmEnb	10	10	Seconds	Q=(!Chnl_Q2)*(Chnl_Q1)+(Chnl_T1)*(!FRONT DR SWITCH)
R2	Chnl_R2__TmEnb	5	10	Seconds	
R3	Chnl_R3__TmEnb	0.3	10	Seconds	
R4	Chnl_R4__TmEnb	5	10	Seconds	
R5	Chnl_R5__TmEnb	15	10	Seconds	
R6	Chnl_R6__TmEnb	0.5	10	Seconds	
R7	Chnl_R7__TmEnb	10	10	Seconds	
R8	Chnl_R8__TmEnb	300	10	Seconds	
R9	Chnl_R9__TmEnb	300	10	Seconds	
R10	Chnl_R10_TmEnb	600	10	Seconds	

☐ CHANGE CURRENT VALUES

Boolean Logic Examples



PMC BOOLEAN EDITOR

MODULE Q Virtual Mod Q CHANNEL 1 Chnl Q1 Cancel OK

SUM OF PRODUCT GROUP TERMS

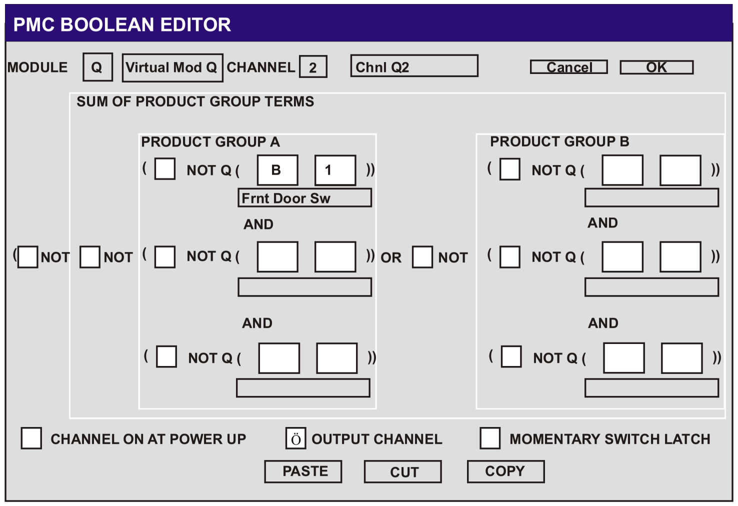
PRODUCT GROUP A: (☐ NOT Q (Q 2)) Chnl Q2 AND (☐ NOT Q ( )) AND (☐ NOT Q ( ))

(☐ NOT ☐ NOT ...)) OR ☐ NOT

PRODUCT GROUP B: (☐ NOT Q ( )) AND (☐ NOT Q ( )) AND (☐ NOT Q ( ))

☐ CHANNEL ON AT POWER UP ✓ OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH

PASTE CUT COPY



PMC BOOLEAN EDITOR

MODULE Q Virtual Mod Q CHANNEL 2 Chnl Q2 Cancel OK

SUM OF PRODUCT GROUP TERMS

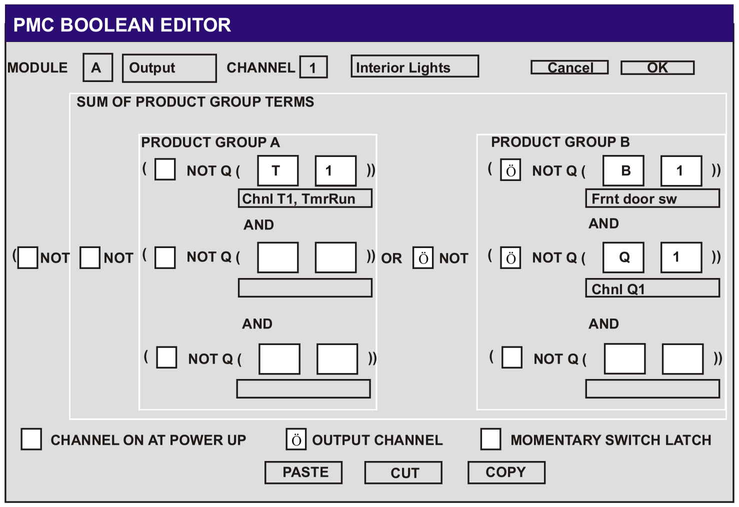
PRODUCT GROUP A: (☐ NOT Q (B 1)) Frnt Door Sw AND (☐ NOT Q ( )) AND (☐ NOT Q ( ))

(☐ NOT ☐ NOT ...)) OR ☐ NOT

PRODUCT GROUP B: (☐ NOT Q ( )) AND (☐ NOT Q ( )) AND (☐ NOT Q ( ))

☐ CHANNEL ON AT POWER UP ✓ OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH

PASTE CUT COPY



PMC BOOLEAN EDITOR

MODULE A Output CHANNEL 1 Interior Lights Cancel OK

SUM OF PRODUCT GROUP TERMS

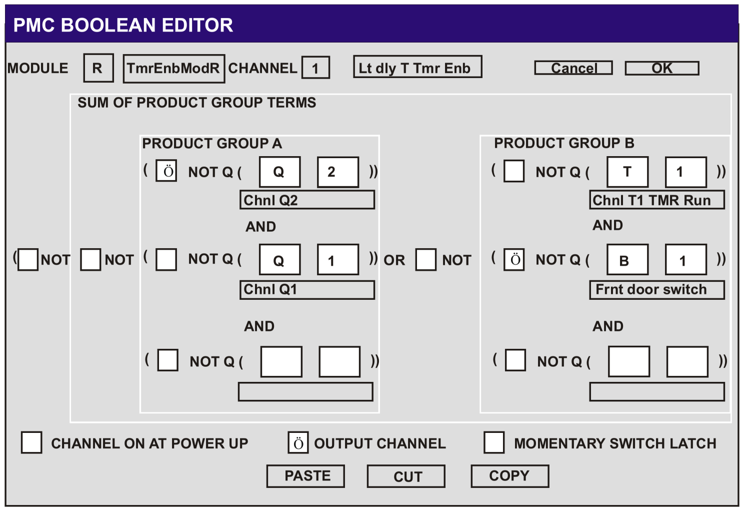
PRODUCT GROUP A: (☐ NOT Q (T 1)) Chnl T1, TmrRun AND (☐ NOT Q ( )) AND (☐ NOT Q ( ))

(☐ NOT ☐ NOT ...)) OR ✓ NOT

PRODUCT GROUP B: (✓ NOT Q (B 1)) Frnt door sw AND (✓ NOT Q (Q 1)) Chnl Q1 AND (☐ NOT Q ( ))

☐ CHANNEL ON AT POWER UP ✓ OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH

PASTE CUT COPY



PMC BOOLEAN EDITOR

MODULE R TmrEnbModR CHANNEL 1 Lt dly T Tmr Enb Cancel OK

SUM OF PRODUCT GROUP TERMS

PRODUCT GROUP A: (✓ NOT Q (Q 2)) Chnl Q2 AND (☐ NOT Q (Q 1)) Chnl Q1 AND (☐ NOT Q ( ))

(☐ NOT ☐ NOT ...)) OR ☐ NOT

PRODUCT GROUP B: (☐ NOT Q (T 1)) Chnl T1 TMR Run AND (✓ NOT Q (B 1)) Frnt door switch AND (☐ NOT Q ( ))

☐ CHANNEL ON AT POWER UP ✓ OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH

PASTE CUT COPY

Boolean Logic Examples

ON DELAY

An "On Delay" turns an output on **X** seconds after an input occurs and turns the output OFF when the input goes away.

Example:
Input=BF1
Output= BC2
Timer= TA4
VA4 = leading edge pulse from BF1

VA4=!prBF1*BF1

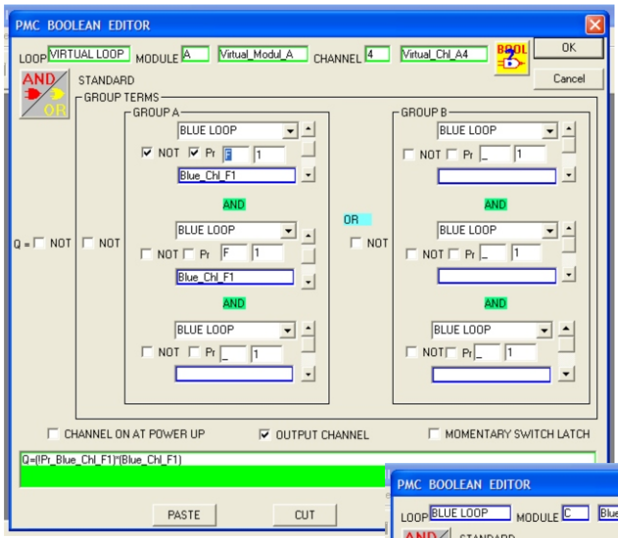
When input BF1 occurs, a pulse will occur on channel VA4

T enable A4= VA4+T run A4

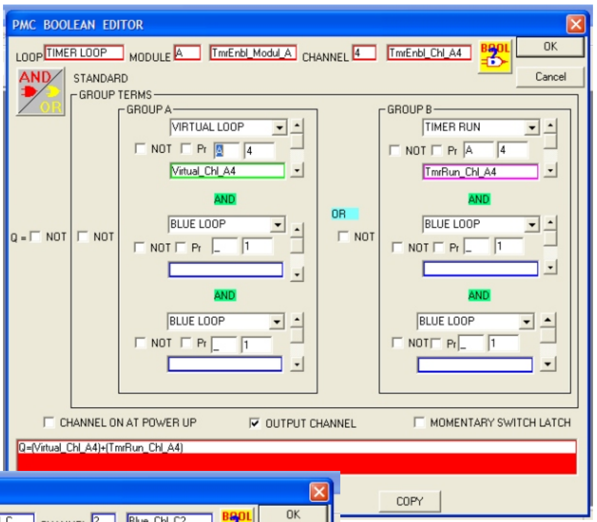
The timer starts with the pulse on VA4 and stays ON because the timer run pulse latches it ON. At the end of the time period, the run pulse goes away unlatching the Timer enable. If input BF1 goes away before the end of the period, the timer is disabled.

BC2=T done A4+BC2*BF1

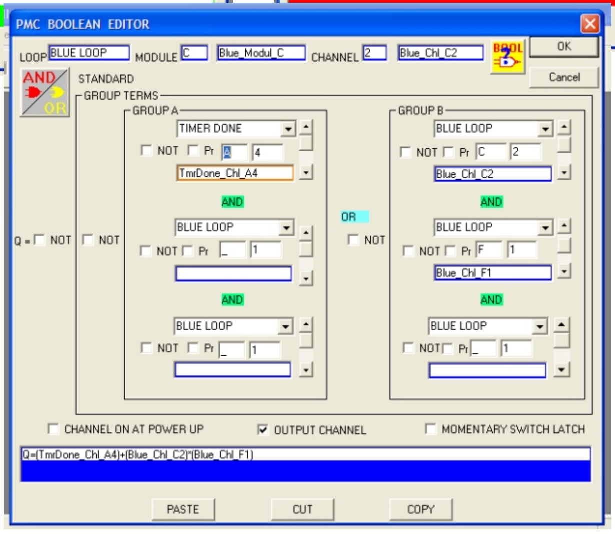
The output BC2 turns ON with the timer done pulse and latches ON, as long as input BF1 is present.



PMC BOOLEAN EDITOR
LOOP VIRTUAL LOOP MODULE A Virtual_Modul_A CHANNEL 4 Virtual_Chl_A4
STANDARD
GROUP A: BLUE LOOP, [✓ NOT] [✓ Pr] F 1, Blue_Chl_F1 AND BLUE LOOP, NOT Pr F 1, Blue_Chl_F1
OR
GROUP B: BLUE LOOP
[✓ OUTPUT CHANNEL]
Q=(!Pr_Blue_Chl_F1)*(Blue_Chl_F1)



PMC BOOLEAN EDITOR
LOOP TIMER LOOP MODULE A TmrEnbl_Modul_A CHANNEL 4 TmrEnbl_Chl_A4
STANDARD
GROUP A: VIRTUAL LOOP, A 4, Virtual_Chl_A4
OR
GROUP B: TIMER RUN, A 4, TmrRun_Chl_A4
[✓ OUTPUT CHANNEL]
Q=(Virtual_Chl_A4)+(TmrRun_Chl_A4)



PMC BOOLEAN EDITOR
LOOP BLUE LOOP MODULE C Blue_Modul_C CHANNEL 2 Blue_Chl_C2
STANDARD
GROUP A: TIMER DONE, A 4, TmrDone_Chl_A4
OR
GROUP B: BLUE LOOP, C 2, Blue_Chl_C2 AND BLUE LOOP, F 1, Blue_Chl_F1
[✓ OUTPUT CHANNEL]
Q=(TmrDone_Chl_A4)+(Blue_Chl_C2)*(Blue_Chl_F1)

Boolean Logic Examples

DOOR OPEN/CLOSE WITH LOCK 160 CH CPU

In this application, there is an air-operated door controlled by a solenoid actuated shuttle valve. The shuttle valve has two control coils. One coil positions the valve to open the door and the other positions it to close the door. The coils do not need to be kept energized to hold the door in position. The designer would like to use a single pushbutton on the exterior of the door and another on the interior of the door.

Since the solenoid coils do not need to be energized continuously and pushbuttons will be used as inputs to initiate the action, it was determined that the PMC outputs would need to be held ON just long enough to ensure that the valve was properly positioned and then turned OFF. In this example, we will use a timer to keep each of the PMC outputs on long enough to perform the mechanical operation, then shut them OFF.

When either pushbutton is pressed the door will open or close, with one exception. When the interior button is used to close the door, the exterior pushbutton will be disabled, in effect locking the door. When the interior button is used to open the door, the exterior button is enabled and may be used to close and open the door.

Since this application is not a high security application, it was deemed necessary to have a means of opening the door from the outside. If the door is closed from the inside, it can be opened from the outside by pushing and holding the exterior button for 3 seconds, after which the door will open. A PMC timer will be used to establish this time. The time can be set to any value that is comfortable.

The programming of this function is a bit complex, but can be accomplished easily by following the steps outlined below. *If assistance is needed please feel free to call Intellitec for help.*

CHANNEL

A1 = DOOR OPEN OUTPUT	**Q6 = VIRTUAL OUTPUT CHANNEL**
A2 = DOOR CLOSE OUTPUT	**Q6 = VIRTUAL OUTPUT CHANNEL**
C1 = INSIDE SWITCH INPUT	**Q8 = VIRTUAL OUTPUT CHANNEL**
C2 = OUTSIDE SWITCH INPUT	**Q9 = VIRTUAL OUTPUT CHANNEL**
Q1 = VIRTUAL OUTPUT CHANNEL	**Q10 = VIRTUAL OUTPUT CHANNEL**
Q5 = VIRTUAL OUTPUT CHANNEL	**R1 = THE ENABLE TO START TIMER 1**
	R3 = THE ENABLE TO START TIMER 3

Using the "Select Channel" screen, select each of these channels and enter the following Boolean statement for each channel.

Channel	**Boolean Logic Statement**
A1 =	**Q8*T1 + S3**
A2 =	**!Q8*T1**
Q1 =	**(A2*C1) + (Q1*!A1)**
Q5 =	**(C1*!Q6) + (C2*!Q6*!Q1)**
Q6 =	**C1 + (C2*!Q1)**
Q8 =	**(Q5*!Q8) + (!Q5*Q8)**
Q9 =	**Q10**
Q10 =	**C2**
R1 =	**Q5 + T1**
R3 =	**C2**

Don't forget to go to PMC setup and select "Timer Set Up". Enter the delay time in seconds for timer 1 and timer 3 here.

You may use different channels to accomplish this, however; it is important to consider the following. Channels Q10, Q9, and Q8 produce an edge detected pulse when the switch is operated. It is important that if different channels are selected that they are written in this order. You may for example have used Q4, Q3, and Q2 instead. In this case substituting in each equation Q4 for Q10, Q3 for Q9 and Q2 for Q8.

If virtual module Q is not available you may use any unused module address as a virtual module.

One Shot Timer

A ***one shot timer*** turns an output ON when an event first occurs. The output will stay ON for a predetermined time. To cycle again, the input must be removed and then re-applied.

In this example we will have the event sensed on input channel BE1.

Channel BC1 is the output channel.
Channel VA1 is a virtual channel on which we will create an *edge detected pulse* that will occur when input BE1 first appears. We will use Timer A3.
For this example the timer will be set for 5 seconds.

Example shown uses the 320 channel CPU and WinPMC II

BE1=input
BC1=output
VA1=virtual channel with edge pulse created by BE1
T enbl A3= the timer enable channel
T Run A3= the timer running channel
T Done A3= the timer done pulse

CREATING THE EDGE PULSE ON VA1
<u>VA1=!prBE1*BE1</u>

Creates a leading edge .040 second on pulse when input BE1 occurs.

ENABLING AND STARTING THE TIMER
<u>T enbl A3=VA1 + T Run A3</u>

The pulse on VA1 enables the timer.
It remains enabled as long as the timer is running.

TURNING THE OUTPUT *ON*
<u>BC1=VA1+BC1 * !T done A3</u>

When the pulse occurs on VA1, BC1 will turn ON and then latch ON because it is equal to itself.
BC1 will stay latched ON until the timer done pulse occurs. When the pulse occurs, BC1 turns OFF.

Boolean Logic Examples

1-3 INPUT, MIRROR HEAT,
DEFROST TIMER,
ONE SHOT TIMER
160 CH CPU

In this application, the PMC user would like to have an output turn a heating element ON and then OFF after a period of time, ***if*** 3 inputs are present. In the event that one or more of the inputs is removed, the output will stay ON until the timer times out.

If all three inputs are present when the timer times out, the output will turn OFF. To start another cycle one of the inputs must be removed and then all 3 inputs must be re-applied.

The Boolean equations to accomplish this are as follows:

A1 = THE HEATER OUTPUT
B1 = INPUT (MASTER SWITCH)
B2 = INPUT (DEFROST FLAG)
B3 = INPUT (DEST. SIGN THERMO SWITCH)
R1 = TIMER 1 ENABLE
S1 = TIMER 1 DONE OUTPUT
T1 = TIMER 1 RUN OUTPUT
Q1 AND Q2 ARE VIRTUAL CHANNELS

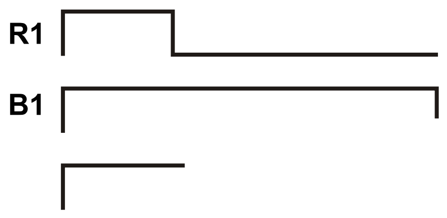
Q1 = S1 + (Q1*Q2)
Q2 = (B1*B2*B3)
R1 = (B1*B2*B3*!Q1) + T1 REDUCING TERMS R1=(Q2*!Q1)+T1
A1 = T1 + (A1*!Q1)
T = SET TIME

Set the time interval by clicking on PMC Setup, then "Timer Setup". Enter the time in seconds for timer R1.

If you would like to create a **one shot timed output** that is triggered from a single input, enter the following Booleans. *This example <u>should not be used for an interior light delay</u> as the light would turn off, even if the switch was still on. See page 9-9 for an interior light delay.*

A1 = THE OUTPUT
B1 = INPUT SWITCH
R1 = TIMER 1 ENABLE
S1 = TIMER 1 DONE OUTPUT
T1 = TIMER 1 RUN OUTPUT
Q1 = VIRTUAL CHANNEL

R1 = (B1*!Q1) + T1
Q1 = S1 + (Q1*B1)
A1 = T1 + (A1*!Q1)
T = SET TIME



Boolean Logic Examples

5 STEP SEQUENCER 160 CH CPU

This example creates a 5 step sequence using a timer and ten virtual channels. The example is written using real channels so that the outputs can be displayed on the status monitor. Any group of 10 free channels, virtual or real, may be used and any available timer may be used.

CHANNEL DEFINITION

A3 = MASTER SWITCH
A3 triggers the sequence and could be any channel you wish to use.

C5 through C1 are virtual outputs that will sequence ON, one at a time, when A3 is turned ON. They will sequence OFF, one at a time after A3 is turned OFF. The first channel to come ON after A3 is turned ON is C5 followed by C4 etc. Set the timer using the timer set up menu, to set the interval between steps.

R1 is the Timer enable for timer 1. The timer determines the time between sequence steps. You may of course use any available timer.

S1 = TIMER DONE PULSE
This should correspond to the timer # that you are using. In the "PMC set up", "Timer set up" screen, set the time you would like between events.

C6-10 are virtual channels that allow C1-C5 to cycle OFF after they have been turned ON.

C5 is the first virtual load to come ON after switch A3 is turned ON and the last to turn OFF after A3 is turned OFF.

By “AND”ing C1 through C5 with another switch, loads can be sequenced ON and OFF.
For example a boolean that stated **"Light Bar = C1 AND light bar switch"** would cause the light bar to turn ON when the Master (A3) is on and only after items “AND”ed with C5-C4 came ON. In other words the light bar would be the last to turn ON in the sequence and the first to turn OFF.

Boolean expressions to create the sequencer are as follows.

```
  R1 = (A3 * !C1) + (!A3*C5)
  C1 = (A3*S1*C2) + (C1*!C6)
  C2 = (A3*S1*C3) + (C2*!C7)
  C3 = (A3*S1*C4) + (C3*!C8)
  C4 = (A3*S1*C5) + (C4*!C9)
  C5 = (A3*S1)+(C5*!C10)
  C6 = !A3*S1
  C7 = !A3*S1*!C1
  C8 = !A3*S1*!C2
  C9 = !A3*S1*!C3
 C10 = !A3*S1*!C4
```

Boolean Logic Example

7 STEP SEQUENCER WITH ADJUSTABLE ON INTERVAL AND ADJUSTABLE OFF INTERVAL 320 CH CPU

This example allows you to program loads to come on in a sequence and turn off in a sequence. The time interval between steps can be set to whatever period is desired. This example uses virtual modules O and P, Timers A1 and A2, and Blue channel F1. You can use any virtual or real modules. If you use real channels, instead of virtual channels, you can use a status monitor to observe what is happening as the sequence progresses.

The only restriction is that the order of the booleans for Virtual Module O and Virtual Module P be as shown in this example. In other words, Step 1 must start with channel 10, step 2, channel 9, step 3, channel 8, and so on. The reason for this is the order in which PMC processes booleans. Booleans are processed from Channel A1 through P10 in order. After P10 is processed it starts again at A1. These cycles repeat every .04 seconds. When done in reverse channel order, channel 10 gets processed in one boolean cycle, channel 9 in the next and so on. In this manner we can have each subsequent step evaluate what happened in the previous step. In other words, channel 9 turns ON only if channel 10 is ON, 8 only if 9 is ON, and so forth.

When the booleans for the virtual channels and timers have been written, any output in the PMC system can be programmed to come on with a sequence step by ANDing it with a virtual step in virtual module P.

FOR EXAMPLE

A fire truck has individual switches for each load that are AND'ed with a Master Switch. This would mean that if all the individual switches were in the ON position, all of the loads would turn ON and OFF simultaneously when the Master Switch is turned ON or OFF. Depending upon the load, this may be undesirable. The sequencer can be used to prevent this. In this example, the first step in the sequence is channel VP10, the second VP9, the third VP8, the fourth VP7, the fifth VP6, the sixth VP5, and the seventh VP4.

After having written the booleans on the following two pages, simply AND the step with the switch that operates the load. This can be done for as many outputs as necessary.

Light Bar Red = light b red sw AND VP10 (turns ON with the first step)
Light Bar White = light b white sw AND VP9 (turns ON with the second step)
Scene Light = scene light sw AND VP8 (turns ON with the third step)

NOTE *that the Master Switch is <u>not used</u> in the boolean for the output.*

Don't forget to set the time period for timer A1 and A2. Practical values to prevent load dump, or a sudden load application would be 0.2 - 0.5 seconds. Timer A1 sets the sequence up time and timer A2 sets the down time.

See the following two pages for booleans needed to produce the sequencer.

Boolean Logic Example

7 STEP SEQUENCER *(Continued from previous page)*

CHANNEL	**DESCRIPTION (B= Blue Loop, V= Virtual Loop, != Not, pr= Prior State, *= AND, += OR)**
BF1	Master Switch This can be any maintained contact or latched push button switch input channel you choose**. Since this is an input <u>no</u> boolean should be written.**
VO1	Leading edge pulse created using Master Switch BF1**.** *See edge detection for more detail* **Vo1 = !prBF1*BF1**
VP1	Trailing edge pulse created using Master Switch BF1**.** *See edge detection for more detail* **Vp1 = prBF1*!BF1**
T enable A1	Timer A1 begins to run when the pulse occurs on VO1 and latches on until channel VO4 comes ON. This timer sets the interval between ON steps. Set the time period in "PMC setup / Timer setup". **T enable A1 = (VO1)+(T enable A1*!VO4)**
T enable A2	Timer A2 begins to run when the pulse occurs on VP1 and latches on until channel VO1 comes on. This timer sets the interval between off steps. Set the time period in "PMC setup / Timer setup". **T enable A2 = (VP1)+(T enable A2*!VO1)**

<u>THE FOLLOWING CHANNEL ORDER IS CRITICAL</u>

VP10	Turns ON when Master Switch BF1 is ON. AND this channel with every output that should come ON with step 1 **VP10=BF1**
VP9	Latches ON if Master Switch BF1 and Timer done A1 and VP10 is ON**.** Unlatches when VO9 comes ON. AND this channel with every output that should come ON with step 2. **VP9=(BF1)*(TmrDone A1)*(VP10)+(VP9)*(!VO9)**
VP8	Latches ON if Master Switch BF1 and timer done A1, and VP9 is ON. Unlatches when VO8 comes ON. AND this channel with every output that should come ON with step 3. **VP8=(BF1)*(TmrDone A1)*(VP9)+(VP8)*(!VO8)**
VP7	Latches ON if Master Switch BF1 and timer done A1, and VP8 is on, Unlatches when VO7 comes ON. AND this channel with every output that should come ON with step 4. **VP7=(BF1)*(TmrDone A1)*(VP8)+(VP7)*(!VO7)**
VP6	Latches ON if Master Switch BF1 and timer done A1, and VP7 is on, Unlatches when VO6 comes ON. AND this channel with every output that should come ON with step 5. **VP6=(BF1)*(TmrDone A1)*(VP7)+(VP6)*(!VO6)**
VP5	Latches ON if Master Switch BF1 and timer done A1, and VP6 is on, Unlatches when VO5 comes ON. AND this channel with every output that should come ON with step 6. **VP5=(BF1)*(TmrDone A1)*(VP6)+(VP5)*(!VO5)**
VP4	Latches ON if Master Switch BF1 and timer done A1, and VP5 is ON, Unlatches when Vo4 comes ON. AND this channel with every output that should come ON with step 6. **VP4=(BF1)*(TmrDone A1)*(VP5)+(VP4)*(!VO4)**

Boolean Logic Example

7 STEP SEQUENCER *(Continued from previous page)*

CHANNEL	**DESCRIPTION**
VO10	This channel latches ON with the trailing edge pulse ON VP1 and when virtual VP10 is OFF. VO10 will unlatch when the Master Switch is turned OFF and VO9 is not ON. You will not use this channel in any booleans, except for Virtual P module. **VO10= (VP1)*(!VP10)+(VO10)*(!BF1)*(!VO9)**
VO9	This channel latches ON if BF1 is OFF, Tmr doneA2 is ON and VO10 is ON, it unlatches if VO8 or VP10 turns ON. You will not use this channel in any booleans, except for Virtual P module. VO9 is used to unlatch VP9 in sequence when the Master Switch is turned OFF. **VO9=(!BF1)*(TmrDone A2)*(VO10)+(VO9)*(!VO8)*(!VP10)**
VO8	This channel latches ON if BF1 is OFF, Tmr doneA2 is ON and VO9 is ON, it unlatches if either VO7 or VP10 turns ON. You will not use this channel in any booleans, except for Virtual P module. VO8 is used to unlatch VP8 in sequence when the Master Switch is turned OFF. **VO8=(!BF1)*(TmrDone A2)*(VO9)+(VO8)*(!VO7)*(!VP10)**
VO7	This channel latches ON if BF1 is OFF, Tmr doneA2 is ON and VO8 is ON, it unlatches if either VO6 or VP10 turns ON. You will not use this channel in any booleans, except for Virtual P module. VO7 is used to unlatch VP7 in sequence when the Master Switch is turned OFF. **VO7=(!BF1)*(TmrDoneA2)*(VO8)+(VO7)*(!VO6)*(!VP10)**
VO6	This channel latches ON if BF1 is OFF, Tmr doneA2 is ON and VO7 is ON, it unlatches if Either VO5 or VP10 turns ON. You will not use this channel in any booleans, except for Virtual P module. VO6 is used to unlatch VP6 in sequence when the Master Switch is turned OFF. **VO6=(!BF1)*(TmrDoneA2)*(VO7)+(VO6)*(!VO5)*(!VP10)**
VO5	This channel latches ON if BF1 is OFF, Tmr doneA2 is ON and VO6 is ON, it unlatches if either VO4 or VP10 turns ON. You will not use this channel in any booleans, except for Virtual P module. VO5 is used to unlatch VP5 in sequence when the Master Switch is turned OFF. **VO5=(!BF1)*(TmrDoneA2)*(VO6)+(VO5)*(!VO4)*(!VP10)**
VO4	This channel latches ON if BF1 is OFF, Tmr doneA2 is ON and VO5 is ON, it unlatches when VP10 turns ON. You will not use this channel in any booleans, except for Virtual P Module. VO4 is used to unlatch VP4 in sequence when the Master Switch is turned OFF. **VO4=(!BF1)*(TmrDoneA2)*(VO5)+(VO4)*(!VP10)**

This completes the booleans needed to create the 7 step sequencer.

Boolean Logic Examples

MOMENTARY PUSH AND HOLD ON, PUSH AND HOLD OFF
160 Channel CPU

In this application, a momentary push button is used to turn an output on when the button has been held for a predetermined period of time. Once the output is on, pushing and holding the button will turn the output off after the same time period.

A1 = PUSHBUTTON ***(<u>DO NOT</u> CHECK MOMENTARY SWITCH LATCH BOX)***

R1= TIMER 1 ENABLE

S1= TIMER DONE PULSE

B1= OUTPUT

Write the Booleans as follows:
R1=A1
B1= (!B1*S1)+(B1*!S1)

In timer set up, adjust the time in timer 1 for the button hold time you would like.

OPERATION

When A1 is pressed and held, the timer R1 runs and a pulse is produced on S1 at the end of the time period. If A1 is released prior to the end of the time period, the timer stops running and no pulse appears on S1.

When the S1 pulse occurs, B1 will turn on because !B1*S1 is true. The next time the boolean is evaluated, 40ms later, the pulse S1 will be gone and the statement B1 *!S1 will be true. *This causes B1 to latch on.*

When the button is pressed and held again, a pulse will appear on S1 when the timer is done. At this time !B1*S1 is false, and B1*!S1 is false. *This causes B1 to unlatch.*

You may notice that this application is the same as a flasher. If the button is held continuously the output will turn on and off and on and off.

Boolean Logic Examples

EDGE DETECTING A SWITCH
160 CH CPU
See Chapter 5 for edge detection with the 320 CH CPU

Normally when a switch is pressed, the PMC system will react to the switch as long as it is held. In some applications you may wish the PMC system to ***react <u>only</u> at the moment the switch is first pressed or released.*** In these special instances, it is not important to the application that the switch is on, only that the switch changed from off to on, or on to off. Edge Detection Booleans must be used to detect the ***transition*** of a switch, either to the on, or off state.

In order to accomplish this, we take advantage of the fact that the PMC CPU processes one Boolean at a time, in order, from A1 to Q10 and then starts at A1 again. Therefore, if we have a switch at F1, we can use two other virtual channels to create edge detection. ***Any two channels can be used as long as the order of the channels is maintained,*** however; for simplicity, in this example we will use F1, F2, & F3.

Boolean Expression (detects when a switch is first turned on)

F1 will be the switch input and will not require any Booleans.
(A Switch will physically have to be wired to Module F Input 1)

F3 = F1 In this Boolean an output F3 will be simply and directly affected by the F1 switch.
(Although this channel is only used as a virtual channel for edge detection, if a test light were connected to an F3 output, the light would turn on.)

Now a Boolean is placed in between these two channels.
F2 = F1 * !F3

As soon as the switch (F1) is pressed F2 will be on because F1 is on and F3 is not on. The following channel F3, has not yet had a chance to process so that F3 will still be off. Immediately following F2, the CPU will process and calculate F3. F3 will turn on because F3 equals the switch F1. ***The next time thru*** the Boolean calculation loop, F2 will turn back off because of the statement not F3. In this example F2 will only be on for a single Boolean processor loop (approximately .040 seconds).

Boolean Expression (detects when a switch is first turned off)

By changing nothing else except the Boolean for F2, PMC can detect when a switch is released:
F2 = !F1 * F3

Boolean Expressions (detects when a switch is first turned on or off)
In this example any change in the switch position will be detected.

By combining the two Booleans for F2 above, PMC can detect either transition of a switch:
F2 = (F1 * !F3) + (!F1 * F3)

SUMMARY

Boolean Statements - Leading edge detection
F1=input
F2= (edge detection pulse occurs here)
F3=virtual channel

F3=F1
F2= F1*!F3 (F2 will turn on for 0.040 seconds when input F1 first turns on)

Trailing edge detection
F3=F1
F2= !F1*F3 (F2 will turn on for 0.040 seconds when input F1 turns off)

Leading or trailing edge detection
F3=F1
F2 = (F1 * !F3) + (!F1 * F3) (F2 will turn on for 0.040 seconds when input F1 turns on or off)

Boolean Logic Examples

USING EDGE DETECTION

MASTER ON/OFF SWITCH

This is a practical application, which requires a single momentary Master Switch to turn many outputs off and then back to their original settings, what will be accomplished is that if the Master Switch is pressed and held for 2 seconds, all lights will turn off. If the Master Switch is pressed again the lights will come on. While the Master is on, individual lights can be turned on and off.

(Note this example uses “Edge Detection”, “Latching”, “Virtual Channels” and “Timers”. It is recommended that these functions be reviewed. A working knowledge of these applications is necessary before proceeding.)

CHANNEL DESCRIPTIONS

F1 = MASTER SWITCH (MOMENTARY)
A1 = MASTER VIRTUAL CHANNEL
A10 = DELAY VIRTUAL CHANNEL
R1 = TIMER ENABLE (TIMER1 SET FOR 2 SECONDS)
S1 = TIMER DONE PULSE

F1 Channel to which the physical Master switch is wired.

A1 Master virtual channel, which will be used in other Booleans to keep track of master on/off state. Pressing the Master switch F1 turns A1 on. Pressing and holding F1 for two seconds will turn A1 off. A1 can then be used in any load Boolean to turn that load off and then return it to its previous state.

A10 Will be used in conjunction with A1 to help sense when the Master Switch is first pressed. ***NOTE*** *Although any channels can be used for this function, the address relationship between A10, A1, and F1 must be maintained.* In other words, the CPU processes channel Booleans A1 – Q10 and then back to A1 again in circular loop. The Master channel (A1) must be processed *after* the Master Switch (F1), but *before* the Delay channel (A10).

R1 Timer1 enable input, which should be set as two-second timer.

S1 Timer1 done output signal.

BOOLEAN’S – MASTER ON/OFF

R1= F1 (timer runs whenever F1 Master Switch is on)
S1 will provide a pulse if the Master Switch has been pressed for 2 seconds

A10 = F1 (A10 is a delayed signal that F1 has turned on, it will be used in A1 to detect when the switch is first pressed)

LET’S BUILD THE A1 BOOLEAN IN STEPS:

First, we want to sense when the Master Switch F1 is first turned on (Edge Detection).
A1 = (F1 * !A10)
When F1 is first pressed, A1 Boolean will be true, because in the order of the Boolean processor, A10 has not yet been processed. A1 will pulse quickly anytime the Master switch is pressed.
Second, let’s now latch the Master channel on, when we sense the pulse
A1 = (F1 * !A10) + A1
Once the first part of the equation pulses A1 on, the second half of the equation will latch it on.
Now since we do not wish to latch A1 on forever, let’s turn off the Master channel, in the event the switch has been held for two seconds.
A1 = (F1 * !A10) + (A1 * !S1)
Now the Master channel will stay latched on as long as we do NOT have a pulse from Timer 1 and this is our final Boolean.

USING EDGE DETECTION *(Continued from previous page)*

LET'S BUILD THE A1 BOOLEAN IN STEPS: *(Continued)*
Now A1 can be used in other simple Booleans.
D1 (Light) = A1 (Master Virtual Channel) * H2 (Light Switch)
This Boolean will turn on the Light only if the Master Virtual Channel AND Light Switch are on.

It is important to remember that in this application the switch must be momentary. Do not check the momentary switch latch box in the Boolean editor screen. If you do check the box, the switch will appear to only operate every other key press and the lights will turn on, and then off again in two seconds.

BACKLIGHTING

This is a practical application, which requires a single momentary Master Switch to turn switch panel backlighting on and off. What will be accomplished is that each press of the Master Switch will toggle switch panel backlighting on and off. Note this backlighting example can be combined with the previous Master On/Off example to have one switch perform both functions.

(Note this example uses "Edge Detection", "Latching", "Toggle" and "Virtual Channels". It is recommended that these functions be reviewed. A working knowledge of these applications is necessary before proceeding.)

CHANNEL DESCRIPTIONS

F1 = MASTER SWITCH
A2 = BACKLIGHT VIRTUAL CHANNEL
A9 = EDGE DETECT VIRTUAL CHANNEL
A10 = DELAY VIRTUAL CHANNEL

F1 Channel to which the physical Master switch is wired.
A2 Backlight virtual channel, which will be used in switch panel Booleans to turn on and off backlighting. Pressing the Master switch F1 toggles A2 on and off
A9 Will be used to sense when the Master Switch is first pressed.
A10 is a delayed signal that F1 has turned on; it will be used in A9 to detect when the switch is first pressed ***NOTE*** *Although any channels can be used for this function, the address relationship between A10, A9, and F1 must be maintained.* In other words, the CPU processes channel Booleans A1 – Q10 and then back to A1 again in circular loop. The Edge channel (A9) must be processed after the Master Switch (F1), but before the Delay Channel (A10).

BOOLEAN'S – BACKLIGHTING

A10 = F1 (A10 is a delayed signal that F1 has turned on, it will be used in A9 to detect when the switch is first pressed)
A9 = F1 * !A10 (Edge detect when switch is first pressed, and produce a short pulse)
A2 = (A9 * !A2) + (!A9 * A2), A2 toggles on and off with each press of switch F1

If you wish to combine Master On/Off function with Backlighting then the A1 channel developed on the previous page needs to be added to the A2 Boolean above to turn backlighting off if the Master Switch is held for two seconds. The new Boolean would be: **A2 = (A9 * !A2) + (!A9 * A2 * A1)**

It is important to remember that in this application the switch must be momentary. Do not check the momentary switch latch box in the Boolean editor screen. If you do, the switch will appear to only operate every other key press.

(***NOTE*** if switch F1 is not being used for any other Boolean Function, such as Master On/off, then this entire Backlighting function can be accomplished by checking the momentary switch latch box in the Boolean editor screen for F1, and then setting A2 = F1)

Boolean Logic Examples

FAN SPEED CONTROL FUNCTION

To make a three speed fan control, a four step function is needed; three speeds and OFF. One way to do this is to first create a pulse from a switch imput. Then, using this pulse, trigger a series of three channels to make a divide by four.

This is done in the following way:

The pulse is created on Channel B1 in this example. The three channels involved with the counter are C1, C2, and C3.

The Booleans are as follows:

C1 = B1 And!C2 Or !B1 And C1

C2 = B1 And C3 Or !B1 And C2

C3 = B1 And C1 Or!B1 And C3

The truth table for this group is:

COUNT	C1	C2	C3
0	0	0	0
1	1	0	1
2	1	1	1
3	0	1	0
4	0	0	0

Then, each of the states is decoded with a three input “And” Boolean to produce three separate outputs:

C4 = C1 And !C2 And C3

C5 = C1 And C2 And C3

C6 = !C1 And C2 And !C3

Boolean Logic Examples

FOUR STEP PUSH BUTTON SWITCH FUNCTION
OFF-LOW-MEDIUM-HIGH-OFF FAN SPEED CONTROL

With this example, Virtual channel VN8, VN9 and VN5 turn on in sequence each time pushbutton BA7 is pressed. You can of course use any module to create this effect. If you would like to perform this function with a 160 channel CPU, you would use virtual module Q, or an unused real module.
This example is shown using a virtual module.

! = NOT function
+ = OR function
*** = AND function**
B = Blue Loop
V = Virtual Loop

BA7 = Push Button Switch
BC1 = Fan low speed Winding
BC2 = Fan Medium speed Winding
BC3 = Fan High speed Winding

VN8 = Virtual Low
VN9 = Virtual Medium
VN5 = Virtual High

VN1 = (BA7 * !VN7) + (VN1*!VN7)	Turn on first step when switch is pressed, but only if not in the middle of the cycle, latch it with VN1 itself, unlatch with VN7 .
VN2 = (VN1 * !BA7) + (VN2 * !VN7)	Sense when switch is released, latch condition, and unlatch with VN7**.**
VN3 = (VN2 * BA7) + (VN3 * !VN7)	Sense when switch is pressed second time, latch condition, and unlatch with VN7**.**
VN4 = (VN3 * !BA7) + (VN4 * !VN7)	Sense when switch is released, latch condition, and unlatch with VN7.
VN5 = (VN4 * BA7) + (VN5 * !VN7)	Sense when switch is pressed third, latch condition, and unlatch with VN7.
VN6 = (VN5 * !BA7) + (VN6 * !VN7)	Sense when switch is released, latch condition, and unlatch with VN7.
VN7 = (VN6 * BA7) + (VN7 * BA7)	Sense when switch is pressed fourth time, maintain condition only as long as switch is pressed.
VN8 = (VN1 * !VN3)	Low speed occurs when low is requested but not medium.
VN9 = (VN3 * !VN5)	Medium speed occurs when medium is requested but not high.

BC1 = VN8 Low speed winding
BC2 = VN9 Medium speed winding
BC3 = VN5 High speed winding.