CAMAC TRIGGER GENERATOR for INO, INO_CTG

Contents:

CAMAC TRIGGER GENERATOR for INO, INO_CTG Specifications

The CAMAC Trigger Generator, INO_CTG has been designed to generate final trigger based on Predefined Coincidence Criterion on a large number of input signals from the prototype Neutrino detector. Additionally it has SCALARS for the FINAL_TRIGGER and for a number of intermediate coincidence logic signals. The SCALARS can be read on CAMAC. The valid signals are latched with final trigger and can also be read on CAMAC. Inputs can be individually deselected from taking part in the trigger generation.
It has been implemented as a two width CAMAC module and most of the logic has been implemented in Altera 10K50 field programmable gate array device.

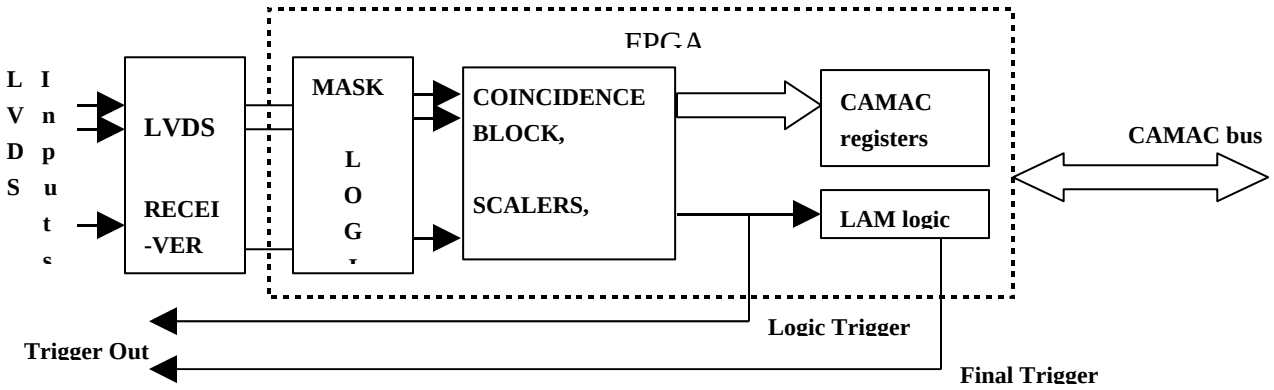


Fig 1: Block Schematic of the INO_CTG

This module accepts first level trigger information derived from the individual layers of the detectors, namely in terms of one, two, three and four fold signals. Further COINCIDENCE LOGIC is implemented in the INO_CTG module. There are a total of 112 signals coming on LVDS lines to the CAMAC module.

Trigger signals generated from the detector assembly:

Input - One, two, three and four fold signals each for 14 X and 14 Y layers
Total 28*4 = 112 from Level 1

Signals - Low Voltage Differential Signaling (LVDS) with 400mV differential signals

C**onnectors -** 4x60 pin FRC standard connectors with 0.1” pitch
28 signals on each connector
Inputs to be connected on twisted pair flat cable.

Output - **LTO** - Logic trigger out, fanout of 1 as NIM output

FTO - Final trigger out = LTO & not LAM
fanout of 3 as NIM output

LTO & **FTO** width in multiples of 50 ns (1 to 7 clocks, default 50 ns)

LAM – CAMAC Look At Me generated with FTO

Multiplicity Register – All 112 inputs latched on FTO, to be read on CAMAC

Scalers – Eight scalers on intermediate triggers, one on LAM width

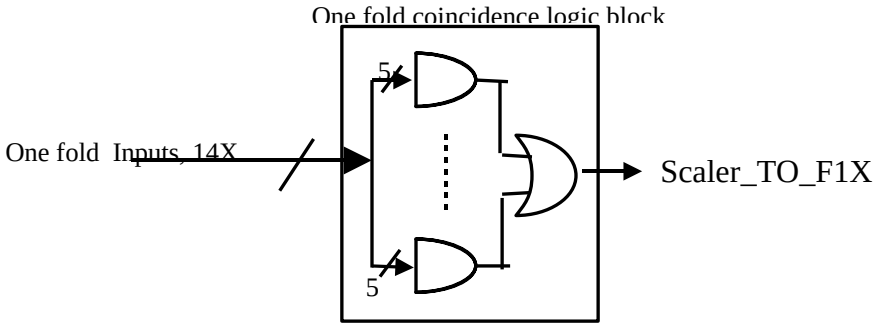
CAMAC TRIGGER GENERATOR for INO, INO_CTG User_manual

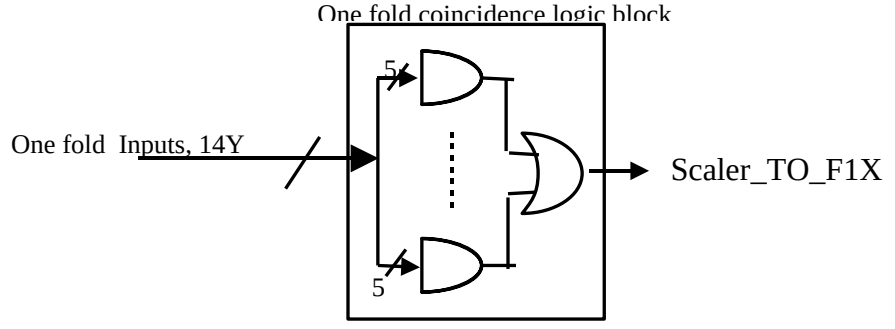
This module has been implemented as a two width CAMAC module and most of the logic has been implemented in Altera 10K50 field programmable gate array device. Low Voltage Differential Signaling (LVDS) has been used as input interface signal standard. Output signals are generated as NIM logic level. It has been designed to generate final trigger based on Predefined Coincidence Criterion on a large number of input signals from the prototype Neutrino detector. Additionally it has SCALARS for the FINAL_TRIGGER and for a number of intermediate coincidence logic signals. The SCALARS can be read on CAMAC. The valid signals are latched with final trigger and can also be read on CAMAC. Inputs can be individually deselected from taking part in the trigger generation.

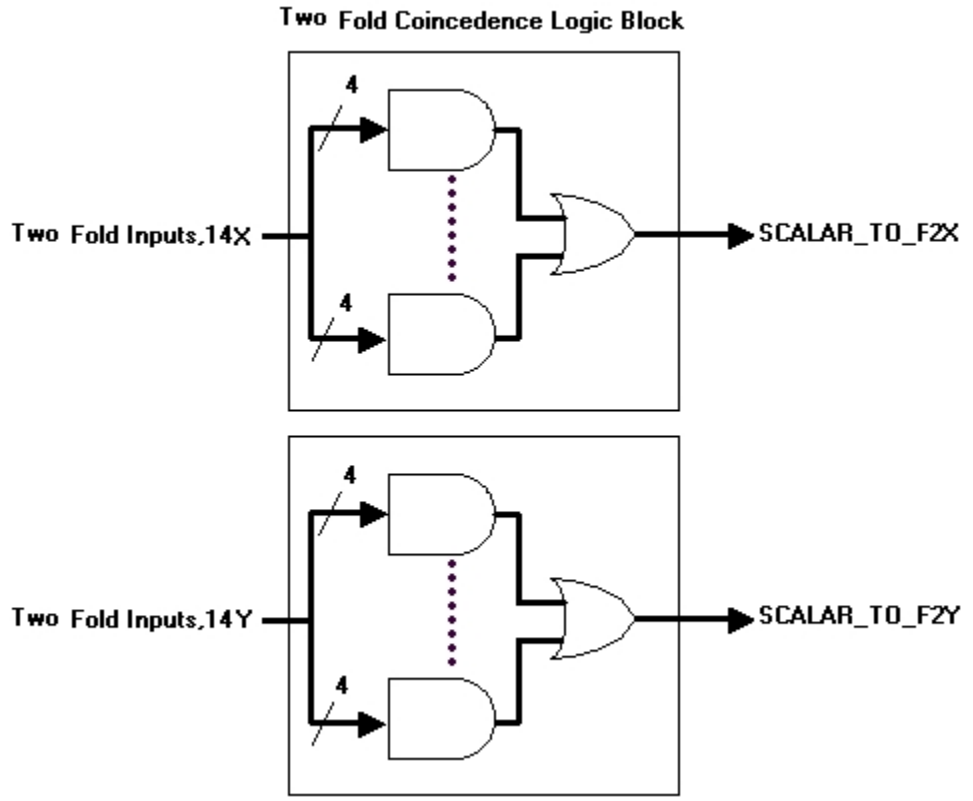
The individual blocks of the module are described in details.

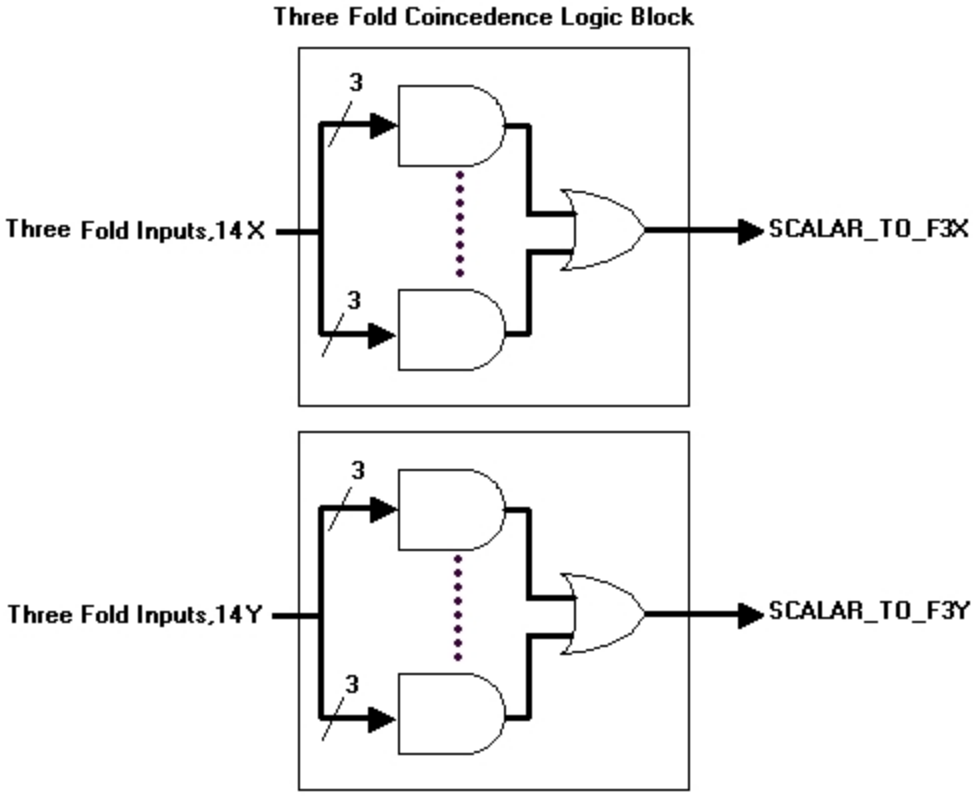
Trigger Logic:

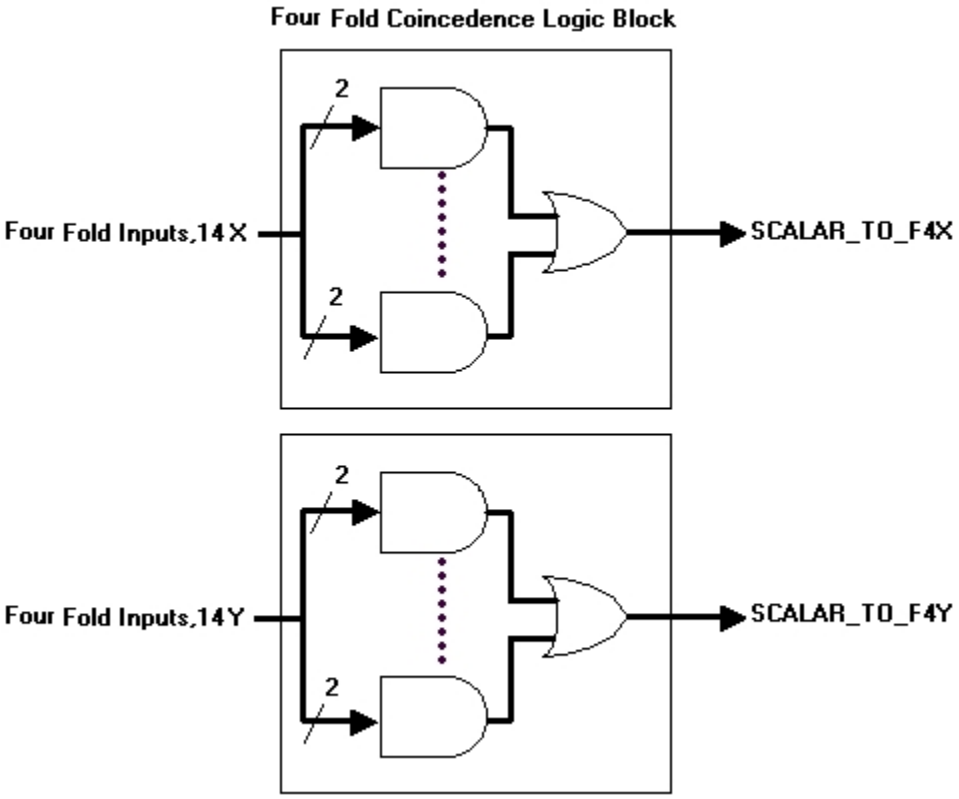
The primary functionality of the module is to generate a trigger based on pre-defined trigger pattern on 112 inputs. It can be described with the help of a block diagram as follows:






Two Fold Coincedence Logic Block
Two Fold Inputs,14X
4
4
SCALAR_TO_F2X
Two Fold Inputs,14Y
4
4
SCALAR_TO_F2Y


Three Fold Coincedence Logic Block
Three Fold Inputs,14X
3
3
SCALAR_TO_F3X
Three Fold Inputs,14Y
3
3
SCALAR_TO_F3Y



Trigger Logic in Boolean equations:

TO_F1X = F1LX1 & F1LX2 & F1LX3 & F1LX4 & F1LX5
+ F1LX2 & F1LX3 & F1LX4 & F1LX5 & F1LX6
+ F1LX3 & F1LX4 & F1LX5 & F1LX6 & F1LX7
:
:
+ F1LX10 & F1LX11 & F1LX12 & F1LX13 & F1LX14

TO_F1Y = F1LY1 & F1LY2 & F1LY3 & F1LY4 & F1LY5
+ F1LY2 & F1LY3 & F1LY4 & F1LY5 & F1LY6
+ F1LY3 & F1LY4 & F1LY5 & F1LY6 & F1LY7
:
:
+ F1LY5 & F1LY11 & F1LY12 & F1LY13 & F1LY14

TO_F2X = F2LX1 & F2LX2 & F2LX3 & F2LX4

```
                +   F2LX2 & F2LX3 & F2LX4  & F2LX5
                +   F2LX3 & F2LX4 & F2LX5  & F2LX6
                    :
                    :
                +   F2LX11 & LX12 & F2LX13 & F2LX14

TO_F2Y      =       F2LY1 & F2LY2 & F2LY3  & F2LY4
                +   F2LY2 & F2LY3 & F2LY4 & F2LY5
                +   F2LY3 & F2LY4 & F2LY5  & F2LY6
                    :
                    :
                +   F2LY11 & F2LY12 & F2LY13 & F2LY14

TO_F3X      =       F3LX1 & F3LX2  & F3LX3
                +   F3LX2 & F3LX3  & F3LX4
                +   F3LX3 & F3LX4 & F3LX5
                    :
                    :
                +    F3LX12 & X13 & F3LX14

TO_F3Y      =       F3LY1 & F3LY2  & F3LY3
                +   F3LY2 & F3LY3  & F3LY4
                +   F3LY3 & F3LY4 & F3LY5
                    :
                    :
                +   F3LY12 & F1LY13 & F3LY14

TO_F4X      =       F4LX1   & F4LX2
                +   F4LX2   & F4LX3
                +   F4LX3  & F4LX4
                    :
                    :
                +    F4LX13 & F4LX14

TO_F4Y      =       F4LY1   & F4LY2
                +   F4LY2   & F4LY3
                +   F4LY3  & F4LY4
```

:
:
+ F4LY13 & F4LY14

LTO = TO_F1X + TO_F1Y
+ TO_F2X + TO_F2Y
+ TO_F3X + TO_F3Y
+ TO_F4X + TO_F4Y

Note: LTO is assigned to one of the front panel NIM outputs on Lemo connector

LAM_ENABLE.SET = N & A (0) & F (26)
LAM_ENABLE.RESET = N & A (0) & F (24) +Z S2

LAM_REQ.D = 1
LAM_REQ.CLK = /FTO
LAM_REQ.RESET = N & A (0) & F (10) + ZS2

LAM = LAM_REQ & LAM_ENABLE

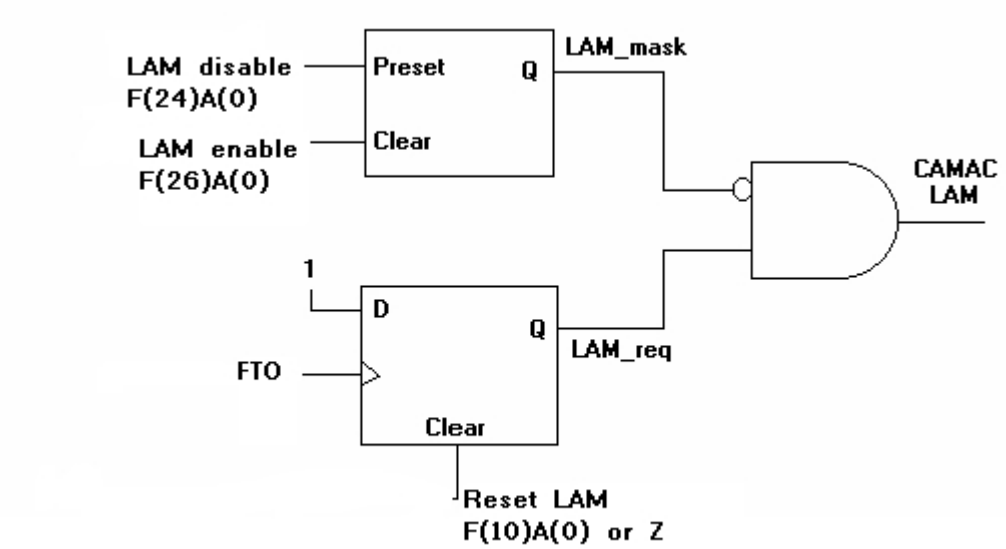
FTO = LTO & /LAM

Note: FTO is assigned to three of the front panel NIM outputs on Lemo connectors

Q = LAM (CAMAC command N & A (0) & F (8))

LAM LOGIC:

Final LAM signal is sent on CAMAC bus when FTO is generated and LAM is enabled through CAMAC command N & A(0) & F(26). LAM can be disabled through CAMAC command N & A(0) & F(24) and reset through CAMAC command N & A 0) & F(10). Once LAM is generated and till it is cleared, another event will generate only LTO and not FTO.



MASK LOGIC

Mask registers are provided to individually enable / disable inputs from taking part in the coincidence. After power-on all channels are masked. According to the bits in the mask corresponding inputs will be masked (bit =0) or unmasked (bit = 1). The Mask registers are 8 bit wide and the bits assignment is as follows:

F1X8 – F1X1 mask register
N & A(0) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F1LX8	F1LX7	F1LX6	F1LX5	F1LX4	F1LX3	F1LX2	F1LX1

F1Y2– F1X9 mask register
N & A(1) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F1LY2	F1LY1	F1LX14	F1LX13	F1LX12	F1LX11	F1LX10	F1LX9

F1Y10– F1Y3 mask register

N & A(2) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F1LY10	F1LY9	F1LY8	F1LY7	F1LY6	F1LY5	F1LY4	F1LY3

F2X4 – F1Y11 mask register

N & A(3) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F2LX4	F2LX3	F2LX2	F2LX1	F1LY14	F1LY13	F1LY12	F1LY11

F2X12 - F2X5 mask register

N & A(4) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F2LX12	F2LX11	F2LX10	F2LX9	F2LX8	F2LX7	F2LX6	F2LX5

F2Y6 – F2X13 mask register

N & A(5) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F2LY6	F2LY5	F2LY4	F2LY3	F2LY2	F2LY1	F2LX14	F2LX13

F2LY14 – F2LY7 mask register

N & A(6) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F2LY14	F2LY13	F2LY12	F2LY11	F2LY10	F2LY9	F2LY8	F2LY7

F3X8 – F3X1 mask register

N & A(7) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F3LX8	F3LX7	F3LX6	F3LX5	F3LX4	F3LX3	F3LX2	F3LX1

F3Y2– F3X9 mask register

N & A(8) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F3LY2	F3LY1	F3LX14	F3LX13	F3LX12	F3LX11	F3LX10	F3LX9

F3Y10– F3Y3 mask register
N & A(9) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F3LY10	F3LY9	F3LY8	F3LY7	F3LY6	F3LY5	F3LY4	F3LY3

F4X4 – F3Y11 mask register
N & A(10) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F4LX4	F4LX3	F4LX2	F4LX1	F3LY14	F3LY13	F3LY12	F3LY11

F4X12 - F4X5 mask register
N & A(11) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F4LX12	F4LX11	F4LX10	F4LX9	F4LX8	F4LX7	F4LX6	F4LX5

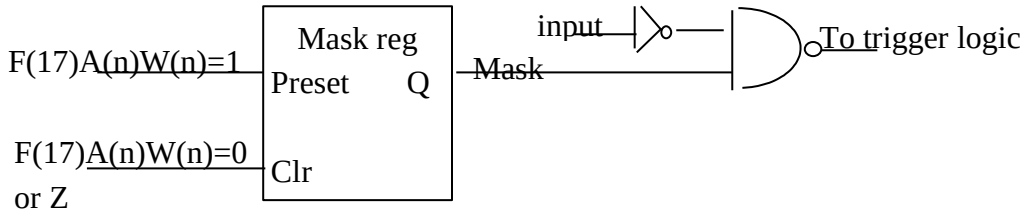
F4Y6 – F4X13 mask register
N & A(12) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F4LY6	F4LY5	F4LY4	F4LY3	F4LY2	F4LY1	F4LX14	F4LX13

F4LY14 – F4LY7 mask register
N & A(13) & F(17) with data bits as shown:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F4LY14	F4LY13	F4LY12	F4LY11	F4LY10	F4LY9	F4LY8	F4LY7

The blocking of respective input works as follows:

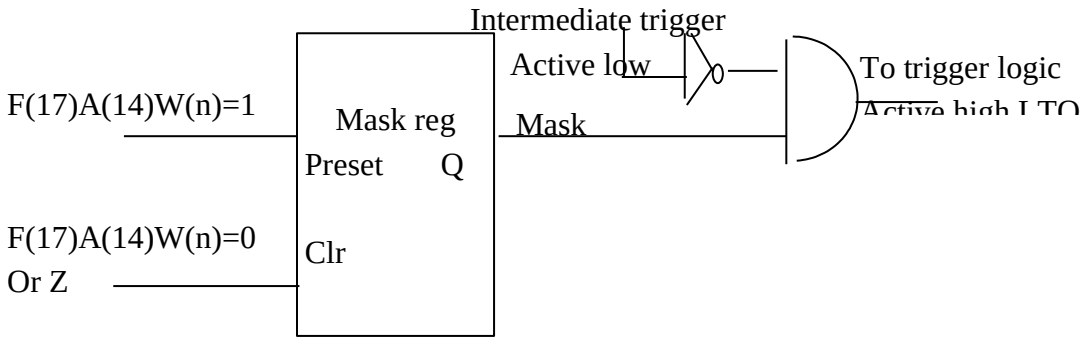


If mask is set, then the respective input does not take part in the trigger logic, but allows the other inputs to decide the coincidence. Once mask is set the respective input is always treated as 1. On POWER and Z, all masks are set but no trigger will be generated. All inputs to be unmasked.Only ones which are no working should be masked.

Mask for intermediate trigger
N & A(14) & F(17) with data bits as shown:

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
F4Y	**F3Y**	**F2Y**	**F1Y**	**F4X**	**F3X**	**F2X**	**F1X**

The blocking of respective intermediate trigger works as follows:



If mask is set, then the respective intermediate trigger does not take part in further trigger logic, but allows the other intermediate trigger inputs to decide the coincidence. Once mask is set the respective input is always treated as 0. On POWER and Z, all masks are set but no trigger will be generated.All intermediate triggers should be unmasked. Masking any one will not block the trigger logic.

Pulse width on front panel LTO and FTO:

N & A(15) & F(17) W(4-1)

Width in term of multiples of clock (50ns) specified by W in range 1 to 7. Default value is 6. The leading edge of the LTO and FTO signals starts with the leading edge of coincidence overlap (after the propagation delay involved in the trigger logic) and the width will vary from set to one additional clock jitter.

CAMAC_OUTPUT

Multiplicity logic: All the inputs to be latched with LTO. Will be cleared with All multiplicity register clear .CLEAR (N & A(1) F(10))

The input pattern can be read as follows:

Read Command N & A (0) & F (0) with data bits as shown

Bit 15 **F1LY2**	**Bit 14** **F1LY1**	**Bit 13** **F1LX14**	**Bit 12** **F1LX13**	**Bit 11** **F1LX12**	**Bit 10** **F1LX11**	**Bit 9** **F1LX10**	**Bit 8** **F1LX9**

Bit 7 **F1LX8**	**Bit 6** **F1LX7**	**Bit 5** **F1LX6**	**Bit 4** **F1LX5**	**Bit 3** **F1LX4**	**Bit 2** **F1LX3**	**Bit 1** **F1LX2**	**Bit 0** **F1LX1**

Read Command N & A (1) & F (0) with data bits as shown

Bit 15 **F2LX4**	**Bit 14** **F2LX3**	**Bit 13** **F2LX2**	**Bit 12** **F2LX1**	**Bit 11** **F1LY14**	**Bit 10** **F1LY13**	**Bit 9** **F1LY12**	**Bit 8** **F1LY11**

Bit 7 **F1LY10**	**Bit 6** **F1LY9**	**Bit 5** **F1LY8**	**Bit 4** **F1LY7**	**Bit 3** **F1LY6**	**Bit 2** **F1LY5**	**Bit 1** **F1LY4**	**Bit 0** **F1LY3**

Read Command N & A (2) & F (0) with data bits as shown

Bit 15 **F2LY6**	**Bit 14** **F2LY5**	**Bit 13** **F2LY4**	**Bit 12** **F2LY3**	**Bit 11** **F2LY2**	**Bit 10** **F2LY1**	**Bit 9** **F2LX14**	**Bit 8** **F2LX13**

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**

F2LX12	**F2LX11**	**F2LX10**	**F2LX9**	**F2LX8**	**F2LX7**	**F2LX6**	**F2LX5**

Read Command N & A (3) & F (0) with data bits as shown

Bit 15 **F3LX8**	**Bit 14** **F3LX7**	**Bit 13** **F3LX6**	**Bit 12** **F3LX5**	**Bit 11** **F3LX4**	**Bit 10** **F3LX3**	**Bit 9** **F3LX2**	**Bit 8** **F3LX1**

Bit 7 **F2LY14**	**Bit 6** **F2LY13**	**Bit 5** **F2LY12**	**Bit 4** **F2LY11**	**Bit 3** **F2LY10**	**Bit 2** **F2LY9**	**Bit 1** **F2LY8**	**Bit 0** **F2LY7**

Read Command N & A (4) & F (0) with data bits as shown

Bit 15 **F3LY10**	**Bit 14** **F3LY9**	**Bit 13** **F3LY8**	**Bit 12** **F3LY7**	**Bit 11** **F3LY6**	**Bit 10** **F3LY5**	**Bit 9** **F3LY4**	**Bit 8** **F3LY3**

Bit 7 **F3LY2**	**Bit 6** **F3LY1**	**Bit 5** **F3LX14**	**Bit 4** **F3LX13**	**Bit 3** **F2LX12**	**Bit 2** **F3LX11**	**Bit 1** **F3LX10**	**Bit 0** **F3LX9**

Read Command N & A (5) & F (0) with data bits as shown

Bit 15 **F4LX12**	**Bit 14** **F4LX11**	**Bit 13** **F4LX10**	**Bit 12** **F4LX9**	**Bit 11** **F4LX8**	**Bit 10** **F4LX7**	**Bit 9** **F4LX6**	**Bit 8** **F4LX5**

Bit 7 **F4LX4**	**Bit 6** **F4LX3**	**Bit 5** **F4LX2**	**Bit 4** **F4LX1**	**Bit 3** **F3LY14**	**Bit 2** **F3LY13**	**Bit 1** **F3LY12**	**Bit 0** **F3LY11**

Read Command N & A (6) & F (0) with data bits as shown

Bit 15 **F4LY14**	**Bit 14** **F4LY13**	**Bit 13** **F4LY12**	**Bit 12** **F4LY11**	**Bit 11** **F4LY10**	**Bit 10** **F4LY9**	**Bit 9** **F4LY8**	**Bit 8** **F4LY7**

Bit 7 **F4LY6**	**Bit 6** **F4LY5**	**Bit 5** **F4LY4**	**Bit 4** **F4LY3**	**Bit 3** **F4LY2**	**Bit 2** **F4LY1**	**Bit 1** **F4LX14**	**Bit 0** **F4LX13**

Read Command N & A (7) & F (0) with data bits as shown

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
0	0	0	0	0	0	0	0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TO_F4Y	TO_F3Y	TO_F2Y	TO_F1Y	TO_F4X	TO_F3X	TO_F2X	TO_F1X

Scalars: LTO, TO_F1X, TO_F2X, TO_F3X, TO_F4X, TO_F1Y, TO_F2Y, TO_F3Y, TO_F4Y. These are 16 bit counters (except LTO which is 24 bit counter) and will enable on CAMAC command N & A (1) & F (26) and will disable on CAMAC command N & A (1) & F (24) readable on CAMAC bus using CAMAC commands as follows:

Read Scalar_LTO_low -> N & A(14) & F(2)
Read Scalar_LTO_high -> N & A(15) & F(2) (bits 8 - 15 are 0)

Read Scalar_Lam_width_low -> N & A(12) & F(2)
Read Scalar_LAM_width_high -> N & A(13) & F(2) (bits 8 - 15 are 0)

Read Scalar_TO_F1X -> N & A(0) & F(2)
Read Scalar_TO_F2X -> N & A(1) & F(2)
Read Scalar_TO_F3X -> N & A(2) & F(2)
Read Scalar_TO_F4X -> N & A(3) & F(2)
Read Scalar_TO_F1Y -> N & A(4) & F(2)
Read Scalar_TO_F2Y -> N & A(5) & F(2)
Read Scalar_TO_F3Y -> N & A(6) & F(2)
Read Scalar_TO_F4Y -> N & A(7) & F(2)

These scalars can be cleared by individual commands or a common command as follows:

Reset Scalar_LTO -> N & A(14) & F(11)
Read Scalar_Lam_width-> N & A(12) & F(2)

Reset Scalar_TO_F1X -> N & A(0) & F(11)
Reset Scalar_TO_F2X -> N & A(1) & F(11)
Reset Scalar_TO_F3X -> N & A(2) & F(11)

Reset Scalar_TO_F4X -> N & A(3) & F(11)
Reset Scalar_TO_F1Y -> N & A(4) & F(11)
Reset Scalar_TO_F2Y -> N & A(5) & F(11)
Reset Scalar_TO_F3Y -> N & A(6) & F(11)
Reset Scalar_TO_F4Y -> N & A(7) & F(11)

Reset ScalarsAll -> N & A(15) & F(11)

Total of 10 scalars. Scalers on LTO and LAM_width are 24bit. Others are 16bits.

Status of control signal: The status of control signals like **I, LAM, Q, X can** be read on

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	I	LAM	Q	X

CAMAC command N & A(1) & F(1). The bit pattern is as follows,

General: Upon POWER_ON and command Z, the module will be in clear state i.e.
LAM_REQ will be cleared, LAM_ENABLE will be reset, scalers will be cleared
Multiplicity Latched input registers will be zero.
All input masks are set.

Command accepted signal X is generated on each implemented CAMAC command as listed in this manual.

Commands C and I have not been implemented and have no effect on the module.

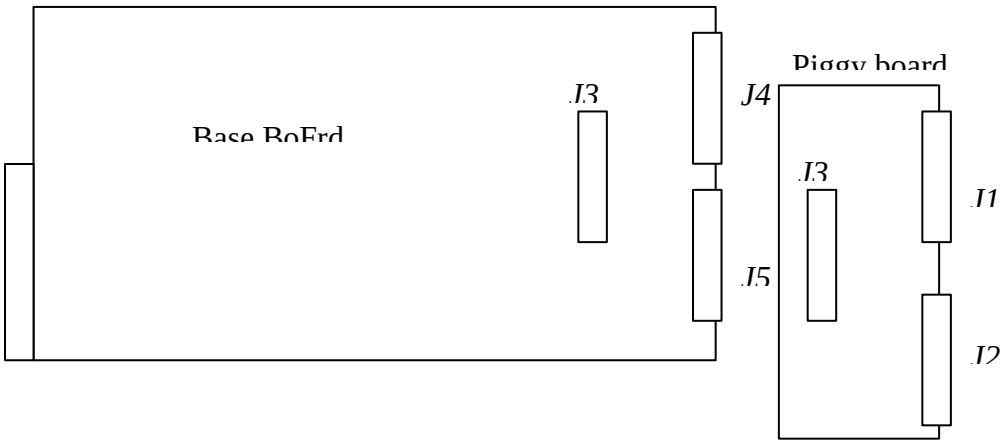
CAMAC TRIGGER GENERATOR for INO, INO_CTG Application_note

This note gives a guide on the use of the module. It describes the connector details, input interconnections and CAMAC commands. It will serve as a ready reference.

Front panel diagram:

Connector details

There are two boards, one base board which has the CFMFC PC edge connector and one piggy board which connects to the base board via a 60 pin FRC cable. Connectors J1 and J2 on the piggy board, connectors J4 and J5 are on the base-board and connectors J3 on both the boards for interconnection.



J1		J2		J4		J5	
J1.31	F1LY2+	J2.31	F2LY2+	J4.31	F3LY2+	J5.31	F4LY2+
J1.32	F1LY2-	J2.32	F2LY2-	J4.32	F3LY2-	J5.32	F4LY2-
J1.33	F1LY3+	J2.33	F2LY3+	J4.33	F3LY3+	J5.33	F4LY3+
J1.34	F1LY3-	J2.34	F2LY3-	J4.34	F3LY3-	J5.34	F4LY3-
J1.35	F1LY4+	J2.35	F2LY4+	J4.35	F3LY4+	J5.35	F4LY4+
J1.36	F1LY4-	J2.36	F2LY4-	J4.36	F3LY4-	J5.36	F4LY4-
J1.37	F1LY5+	J2.37	F2LY5+	J4.37	F3LY5+	J5.37	F4LY5+
J1.38	F1LY5-	J2.38	F2LY5-	J4.38	F3LY5-	J5.38	F4LY5-
J1.39	F1LY6+	J2.39	F2LY6+	J4.39	F3LY6+	J5.39	F4LY6+
J1.40	F1LY6-	J2.40	F2LY6-	J4.40	F3LY6-	J5.40	F4LY6-
J1.41	F1LY7+	J2.41	F2LY7+	J4.41	F3LY7+	J5.41	F4LY7+
J1.42	F1LY7-	J2.42	F2LY7-	J4.42	F3LY7-	J5.42	F4LY7-
J1.43	F1LY8+	J2.43	F2LY8+	J4.43	F3LY8+	J5.43	F4LY8+
J1.44	F1LY8-	J2.44	F2LY8-	J4.44	F3LY8-	J5.44	F4LY8-
J1.45	F1LY9+	J2.45	F2LY9+	J4.45	F3LY9+	J5.45	F4LY9+
J1.46	F1LY9-	J2.46	F2LY9-	J4.46	F3LY9-	J5.46	F4LY9-
J1.47	F1LY10+	J2.47	F2LY10+	J4.47	F3LY10+	J5.47	F4LY10+
J1.48	F1LY10-	J2.48	F2LY10-	J4.48	F3LY10-	J5.48	F4LY10-
J1.49	F1LY11+	J2.49	F2LY11+	J4.49	F3LY11+	J5.49	F4LY11+
J1.50	F1LY11-	J2.50	F2LY11-	J4.50	F3LY11-	J5.50	F4LY11-
J1.51	F1LY12+	J2.51	F2LY12+	J4.51	F3LY12+	J5.51	F4LY12+
J1.52	F1LY12-	J2.52	F2LY12-	J4.52	F3LY12-	J5.52	F4LY12-
J1.53	F1LY13+	J2.53	F2LY13+	J4.53	F3LY13+	J5.53	F4LY13+
J1.54	F1LY13-	J2.54	F2LY13-	J4.54	F3LY13-	J5.54	F4LY13-
J1.55	F1LY14+	J2.55	F2LY14+	J4.55	F3LY14+	J5.55	F4LY14+
J1.56	F1LY14-	J2.56	F2LY14-	J4.56	F3LY14-	J5.56	F4LY14-
J1.57	GND	J2.57	GND	J4.57	GND	J5.57	GND
J1.58	GND	J2.58	GND	J4.58	GND	J5.58	GND
J1.59	GND	J2.59	GND	J4.59	GND	J5.59	GND
J1.60	GND	J2.60	GND	J4.60	GND	J5.60	GND

J1		J2		J4		J5	
J1.1	F1LX1+	J2.1	F2LX1+	J4.1	F3LX1+	J5.1	F4LX1+
J1.2	F1LX1-	J2.2	F2LX1-	J4.2	F3LX1-	J5.2	F4LX1-
J1.3	F1LX2+	J2.3	F2LX2+	J4.3	F3LX2+	J5.3	F4LX2+
J1.4	F1LX2-	J2.4	F2LX2-	J4.4	F3LX2-	J5.4	F4LX2-
J1.5	F1LX3+	J2.5	F2LX3+	J4.5	F3LX3+	J5.5	F4LX3+
J1.6	F1LX3-	J2.6	F2LX3-	J4.6	F3LX3-	J5.6	F4LX3-
J1.7	F1LX4+	J2.7	F2LX4+	J4.7	F3LX4+	J5.7	F4LX4+
J1.8	F1LX4-	J2.8	F2LX4-	J4.8	F3LX4-	J5.8	F4LX4-
J1.9	F1LX5+	J2.9	F2LX5+	J4.9	F3LX5+	J5.9	F4LX5+
J1.10	F1LX5-	J2.10	F2LX5-	J4.10	F3LX5-	J5.10	F4LX5-
J1.11	F1LX6+	J2.11	F2LX6+	J4.11	F3LX6+	J5.11	F4LX6+
J1.12	F1LX6-	J2.12	F2LX6-	J4.12	F3LX6-	J5.12	F4LX6-
J1.13	F1LX7+	J2.13	F2LX7+	J4.13	F3LX7+	J5.13	F4LX7+
J1.14	F1LX7-	J2.14	F2LX7-	J4.14	F3LX7-	J5.14	F4LX7-
J1.15	F1LX8+	J2.15	F2LX8+	J4.15	F3LX8+	J5.15	F4LX8+
J1.16	F1LX8-	J2.16	F2LX8-	J4.16	F3LX8-	J5.16	F4LX8-
J1.17	F1LX9+	J2.17	F2LX9+	J4.17	F3LX9+	J5.17	F4LX9+
J1.18	F1LX9-	J2.18	F2LX9-	J4.18	F3LX9-	J5.18	F4LX9-
J1.19	F1LX10+	J2.19	F2LX10+	J4.19	F3LX10+	J5.19	F4LX10+
J1.20	F1LX10-	J2.20	F2LX10-	J4.20	F3LX10-	J5.20	F4LX10-
J1.21	F1LX11+	J2.21	F2LX11+	J4.21	F3LX11+	J5.21	F4LX11+
J1.22	F1LX11-	J2.22	F2LX11-	J4.22	F3LX11-	J5.22	F4LX11-
J1.23	F1LX12+	J2.23	F2LX12+	J4.23	F3LX12+	J5.23	F4LX12+
J1.24	F1LX12-	J2.24	F2LX12-	J4.24	F3LX12-	J5.24	F4LX12-
J1.25	F1LX13+	J2.25	F2LX13+	J4.25	F3LX13+	J5.25	F4LX13+
J1.26	F1LX13-	J2.26	F2LX13-	J4.26	F3LX13-	J5.26	F4LX13-
J1.27	F1LX14+	J2.27	F2LX14+	J4.27	F3LX14+	J5.27	F4LX14+
J1.28	F1LX14-	J2.28	F2LX14-	J4.28	F3LX14-	J5.28	F4LX14-
J1.29	F1LY1+	J2.29	F2LY1+	J4.29	F3LY1+	J5.29	F4LY1+
J1.30	F1LY1-	J2.30	F2LY1-	J4.30	F3LY1-	J5.30	F4LY1-

CAMAC commands:

For INO_TEST module

Z,C reset module
F(26)A(0) Disable O/P
F(24)A(0) enable O/P
F(16)A(0) enable or Disable coincidence pattern out of 5 O/P
Example
COMMAND F(24)A(0)
Write DATA 1F
COMMAND F(16)A(0) will enable all 5 O/P

For INO module

Initialize, Z: Set LAM mask, clear LAM, clear multiplicity register, clear scalers

F(0)A(0-7): Read multiplicity registers

F(1)A(1): Read status

F(2)A(0-7): read scalers on intermediate coincidences

F(2)A(14-15): read LTO scaler

F(2)A(12-13): read LAM width scaler

F(8)A(0): Test LAM; Q=0 if LAM set

F(10)A(0): Clear LAM

F(11)A(0-15): clear scalers

F(17)A(0-13): set/reset input mask; 1 = set; 0 = reset

F(17)A(14): set/reset mask register for intermediate coincidence, 1 = set; 0 = reset

F(17)A(15): set LTO / FTO width

F(24)A(0): set LAM mask, disable LAM

F(26)A(0): reset LAM mask, enable LAM

Recommended sequence of commands:

After POWER_ON or any other time to reinitialize the module:

Initialize, Z
Set input mask registers
Enable LAM

After receiving LAM:

Read multiplicity registers
Clear multiplicity registers
Clear LAM

To estimate the system dead time

read LAM width scaler
Clear LAM width scaler

To estimate the input event rate

Read trigger scalers
Clear trigger scalers

Sample programs for the CAMAC spy program

1. Enable all inputs, reset masks: for details of bit map, see user manual

Write data = 255
NAF = N 0 17
Write data = 255
NAF = N 1 17
Write data = 255
NAF = N 2 17
Write data = 255
NAF = N 3 17
write data = 255
NAF = N 4 17

Write data = 255
NAF = N 5 17
write data = 255
NAF = N 6 17
Write data = 255
NAF = N 7 17
Write data = 255
NAF = N 8 17
Write data = 255
NAF = N 9 17
Write data = 255
NAF = N 10 17
Write data = 255
NAF = N 11 17
Write data = 255
NAF = N 12 17
Write data = 255
NAF = N 13 17
Write data = 255
NAF = N 14 17
Write data = 255

2. Disable all inputs, reset masks: for details of bit map see user manual

Write data = 0
NAF = N 0 17
Write data = 0
NAF = N 1 17 0
Write data = 0
NAF = N 2 17 0
Write data = 0
NAF = N 3 17 0
Write data = 0
NAF = N 4 17 0
Write data = 0
NAF = N 5 17 0

Write data = 0
NAF = N 6 17 0
Write data = 0
NAF = N 7 17 0
Write data = 0
NAF = N 8 17 0
Write data = 0
NAF = N 9 17 0
Write data = 0
NAF = N 10 17 0
Write data = 0
NAF = N 11 17 0
Write data = 0
NAF = N 12 17 0
Write data = 0
NAF = N 13 17 0
Write data = 0
NAF = N 14 17 0

3. Disabling a specific input: for ex. Input F3LX1

Write data = 254
NAF = N 7 17

This enables all the other inputs in this group

4. Enabling a specific input: for ex. Input F3LX1

Write data = 1
NAF = N 7 17

This disables all the other inputs in this group

If it is required to enable or disable a specific input without disturbing others in the group, then the previous value of the group must be preserved and only that bit should be modified. In the above example, if the group mask is defined as MASK8 then it should be modified as

MASK8 = MASK8 (bit wise AND) 0xFE to enable

and
MASK8 = MASK8 (bit wise OR) 0x01 to disable

5. Reading multiplicity registers

NAF = N 0 26 ; enable LAM
Wait for LAM
NAF = N 0 0
Read data
NAF = N 1 0
Read data
NAF = N 2 0
Read data
NAF = N 3 0
Read data
NAF = N 4 0
Read data
NAF = N 5 0
Read data
NAF = N 6 0
Read data
NAF = N 7 0
Read data
NAF = N 0 10; reset LAM

6. Reading scalers:

NAF = N 0 2
Read data
NAF = N 1 2
Read data
NAF = N 2 2
Read data
NAF = N 3 2
Read data
NAF = N 4 2

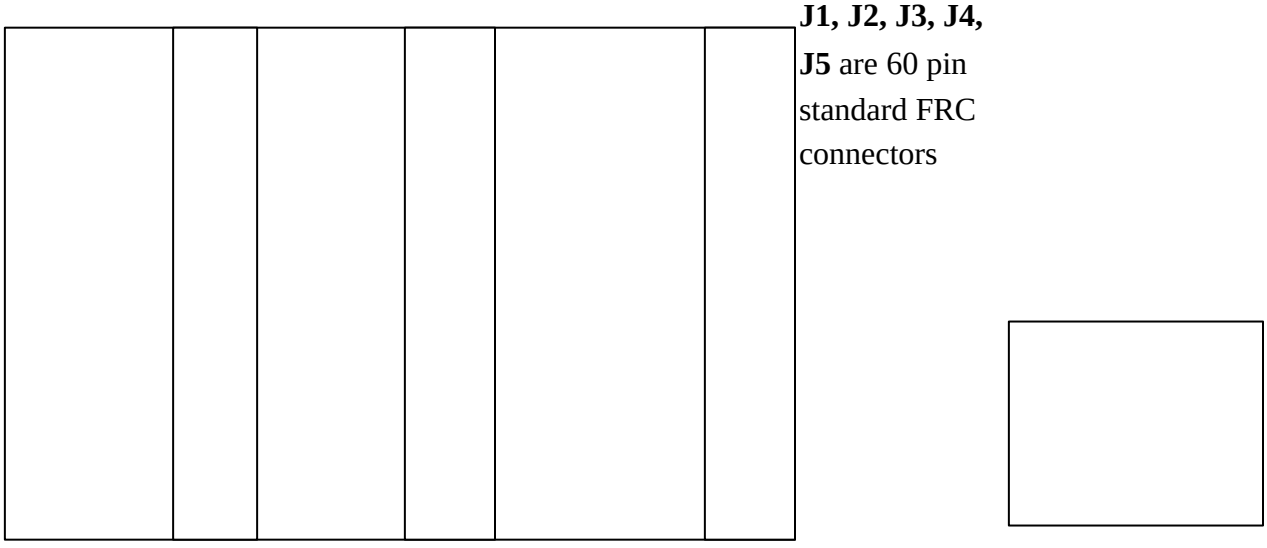
Read data
NAF = N 5 2
Read data
NAF = N 6 2
Read data
NAF = N 7 2
Read data
NAF = N 12 2; read LAM width scaler low
Read data
NAF = N 13 2; read LAM width scaler high
Read data
NAF = N 14 2; read LTO scaler low
Read data
NAF = N 15 2; read LTO scaler high
Read data

CAMAC TRIGGER GENERATOR for INO, INO_CTG
Test setup and observations

Test setup:

As the inputs to the INO_CTG module are LVDS logic signal which are not available from any standard logic source, a special CAMAC module was developed to generate the required test signal. The highest fold of coincidence is 5 on 1fold signal from the first layer, thus this test module generates 5 separate signals. These five signals can be individually masked so as to facilitate the coincidence logic of the INO_CTG module. These five signals are in two groups of two and three signals. Two free running signal sources are built-in in the tset module with the width as well as overlap of these two signals groups settable using front panel pots. This allows the testing of the minimum overlap required to generate final trigger. The five signals can be individually connected to any of the 112 inputs using a specially made cable with 10 pin FRC connector on one end and five individual pairs on the other.

A block schematic of the test setup is as shown:



Step by step procedure for field testing of the module using CAMAC test module:

1. Adjust the widths and overlaps of the output signals of the test module as per required test stimulus.
2. Connect the five outputs to the required inputs of the INO_CTG module.

3. Enable the test module by enabling and unmasking its five output
4. Run test programs to test various features of the INO_CTG module as detailed in the application note.
5. Checking multiplicity logic:
    a. To test two fold coincidence, first connect only one input to one of the four-fold input and check that no coincidence is generated.
    b. With this connection itself, mask-off the neighboring input so that this will now form a coincidence condition. Check if trigger is generated. LTO, FTO and LAM should be generated. Also the multiplicity registers read should be correct.
    c. In similar way, check the two fold for other four-fold inputs.
    d. Following similar steps check three, four and five fold coincidences.

6. In all of the above tests check if LAM enable, disable and reset works correctly. When LAM is set on trigger and is disabled, it will be removed from the CAMAC bus. But will be reasserted when it is enabled. Whereas resetting LAM will clear the LAM request register and enabling or disabling it will have no effect on the LAM on CAMAC bus.

7. In all of the above tests, check if the respective scalers are read correctly.

8. Also check the LAM width scaler. It indicates the system dead-time. To test this feature, connect inputs so that trigger will happen. Enable the respective inputs. Wait for LAM, then put a varying delay before clearing LAM. Now read the LAM width scaler. With different values for the delay this should give appropriate value.

    Test program for checking LAM width scaler:

    Wait for LAM
    Delay (time value)
    NAF = N 0 10 ; reset LAM
    NAF = N 12 2
    Read data
    NAF = N 13 2
    Read data

9. Set different values for **LTO / FTO** widths and check the output

Observations:

The module has been tested for all its functionality as listed as below and is found to be working as required.

Minimum overlap to detect coincidence

Propagation delay from input to output

Feedback notes