7D02 LOGIC ANALYZER

7D02 LOGIC ANALYZER

POCKET REFERENCE

Tektronix
COMMITTED TO EXCELLENCE

Word Recognition

Counters

NOT Word Recognition

Sequential Word Recognition

Data Comparison

Block Qualification

Data Qualification

Asynchronous Timing

Synchronous Timing

Table of Contents

Table of Contents (cont)

INTRODUCTION

The 7D02 Pocket Reference Manual was designed to serve as a "memory jogger" from which an experienced user can program a 7D02. The programming examples were developed using a 6802 System-Under-Test, but are applicable to any microprocessor or bus-oriented system.

The manual is split into three main sections: General Programming, Qualification, and Timing Option examples. The examples build in complexity from the beginning of each section to the end. Each example is independent of the other examples.

CONVENTIONS USED WITHIN THE MANUAL

Each example contains a title, a description of the condition that the program is trying to find, a list of keystrokes used to program the example, and a description of how the program works.

The keys actually pressed are printed in **bold** type in the keystrokes column. Beneath each keystroke is a list of required numeric fields or menu selections that also must be entered.

NOTE

If a selection for a numeric field or menu selection is not listed under the keystroke, it is assumed to be the POWER UP default (which may be different from what the user entered in a previous example).

The program, as shown, may not always fit on the 7D02 screen at one time, but is being shown complete for the sake of clarity.

7D02 PROGRAMMING CONCEPTS

The keys used to enter a program can be broken into four groups: event keys, command keys, structure keys, and the "[ ]", "OR" and "NOT" keys. Some keys may overlap into more than one group (for example, the COUNTER key is both an event key and a command key). The keys in each group are usually located together on the keyboard. The QUALIFY key is not grouped with the command keys, but may be used as a command. Pressing one of these keys in a valid context will generate one or more lines in the program.

EVENTS

An event is a condition, detectable by the 7D02, that is either TRUE or FALSE. A simple event in the 7D02 consists of either a word recognition (using one of the four word recognizers) or a counter reaching a user-defined value. A word recognizer allows the user to detect the occurrence of a specified value on the system under test bus. A counter allows the user to detect a user-defined length of time or to detect a user-defined number of counts.

COMMANDS

A command is an action to be taken by the 7D02 when a particular event becomes TRUE. These actions perform the tasks of: acquisition completion (the TRIGGER command), data storage (the QUALIFY command), program sequencing (the GOTO command) and counter control (the COUNTER command). These commands are used to acquire the data needed to localize the problem or to transfer program execution to the next part of the sequence of conditions used to localize the problem.

COMPOUND EVENTS

A compound event consists of one or more simple events enclosed within brackets. If more than one simple event is included

within the brackets, each simple event must be separated by an "OR" keystroke. The compound event is enclosed within brackets by pushing the "[ ]" key below an "IF" prompt, an "OR IF" prompt, or a "STORE ON" prompt and by pushing the "[ ]" key again when all of the desired simple events have been entered.

A compound event is either TRUE or FALSE. It is TRUE whenever one or more of the simple events within it is TRUE. It is FALSE only when all of the simple events within it are FALSE. Compound events may not contain other compound events (i.e., they may not be nested).

COMPLEMENTED EVENTS

A simple event within a compound event may be complemented by preceding the event with the NOT key. The complemented event will be TRUE whenever the simple event is FALSE. The NOT key may be used with either a word recognizer or a counter. A compound event cannot be complemented with the NOT key; i.e., the NOT may not appear outside of a bracket.

If the NOT key precedes a word recognizer, the complemented event will become TRUE when any one or more of the individual fields within the word recognizer becomes FALSE. If the NOT key precedes a counter, the complemented event is TRUE until the counter reaches its user-defined value.

COMPOUND COMMANDS

A compound command consists of one or more commands enclosed within brackets. All commands within a compound command are executed together (simultaneously) whenever the event associated with them becomes TRUE. Compound commands can not contain other compound commands (i.e., they can not be nested).

IF-THEN-ELSE

The 7D02 user language uses an IF-THEN-ELSE syntax to condition the execution of a command on the occurrence of an event. Events (the term event can refer to either a simple event or a compound event) are entered under an "IF" or an "OR IF" prompt. Commands (simple or compound) are entered under a "THEN DO" prompt or under an "ELSE DO" prompt.

The "IF", "THEN DO", and "OR IF" prompts are supplied automatically by the 7D02 whenever events or commands are entered (following the syntax shown in the example above). If an else clause is desired it may be entered by pressing the "ELSE" key whenever the cursor is resting immediately below an "IF" or an "OR IF" prompt. This will cause the "IF" or "OR IF" prompt to be replaced by the "ELSE DO" prompt. The command associated with the ELSE clause can then be entered.

Whenever the event associated with an "IF" prompt or an "OR IF" prompt becomes TRUE (on a given state clock) the command under the following "THEN DO" prompt is executed. If more than one event is TRUE on any particular state clock, all of the commands associated with those events are executed simultaneously. If none of the events are TRUE on a state cycle, then the command in the ELSE clause is executed (if one exists). The user can have as many "OR IF"—"THEN DO" clause pairs as necessary (subject to memory limitations). Only one ELSE clause may exist per TEST.

TESTS

When entering a program, a TEST is a structure than encloses the IF-THEN-ELSE syntax described above. The TEST allows a program to contain up to four separate IF-THEN-ELSE setups. When the program is run, each TEST will be either enabled or disabled (all events and commands within a disabled TEST are ig-

nored). Only one TEST can be active at any one time.

A TEST is automatically created when a valid programming keystroke is entered into an empty program, after an "END TEST" keystroke or an "END QUALIFY" keystroke. A TEST is completed when an "END" keystroke is entered.

The number following the "TEST" prompt and the number in the left column of the screen indicate the TEST number. There can be up to four different TESTs in a 7D02 program. These TESTs are automatically numbered from 1 to 4 as they are used. Each TEST can contain completely different combinations of events and commands (subject to resource limitations). Only one TEST can be active at a time when a program is run. This means that if TEST 1 is active and an event in TEST 2 becomes TRUE, then the command associated with the event in TEST 2 will not be executed. A TEST becomes active by executing a GOTO command to that TEST (this also deactivates the TEST executing the GOTO command).

It is important to remember that all events within a TEST are evaluated simultaneously. Therefore, if a user wishes to detect a sequence of events (for example the execution of subroutine A followed by the execution of subroutine B) two TESTs must be used. The first TEST will detect the occurrence of subroutine A. The second TEST will detect the occurrence of subroutine B and then perform the desired function (e.g., count or trigger).

If a TEST containing no GOTOs is activated, that TEST will remain active until the 7D02 stops acquiring data.

A TEST may be empty (i.e., it may have no events or commands defined within it). If an empty test is ever activated, the 7D02 will do nothing except continue to acquire data.

A TEST may also contain only an ELSE clause. In this case the command in the ELSE clause will be executed on every state clock in which that TEST is active.

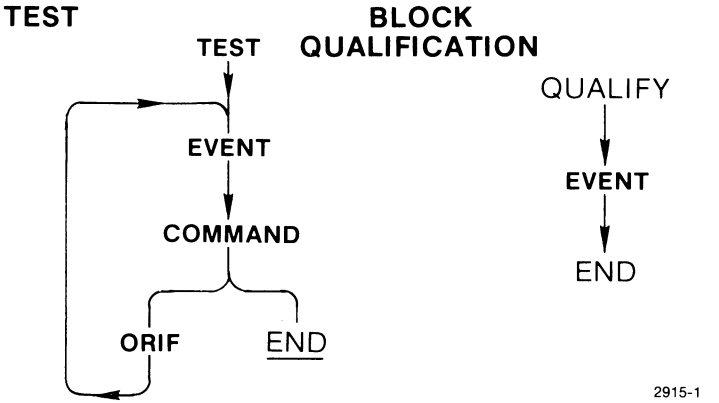
DATA QUALIFICATION

A program may contain a data qualification block. This can be entered by pressing the QUALIFY key in an empty program or after an "END TEST" prompt. Only one QUALIFY block is allowed in a program. The event within the QUALIFY block may be a compound event and may contain one or more complemented events.

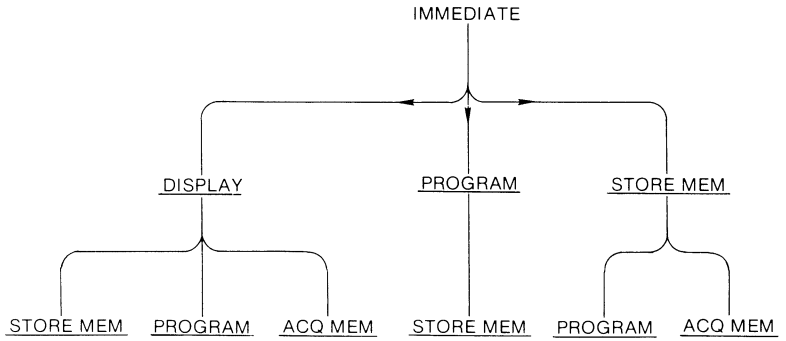
The qualify block controls the storage of data into the Main Acquisition Memory (and into the Timing Option Memory if the Timing Option is used synchronously). Data is only stored in these memories on state clocks in which the event in the QUALIFY block is TRUE. The QUALIFY block is independent of the tests in a program (i.e., the QUALIFY block acts the same regardless of which TEST is currently active).

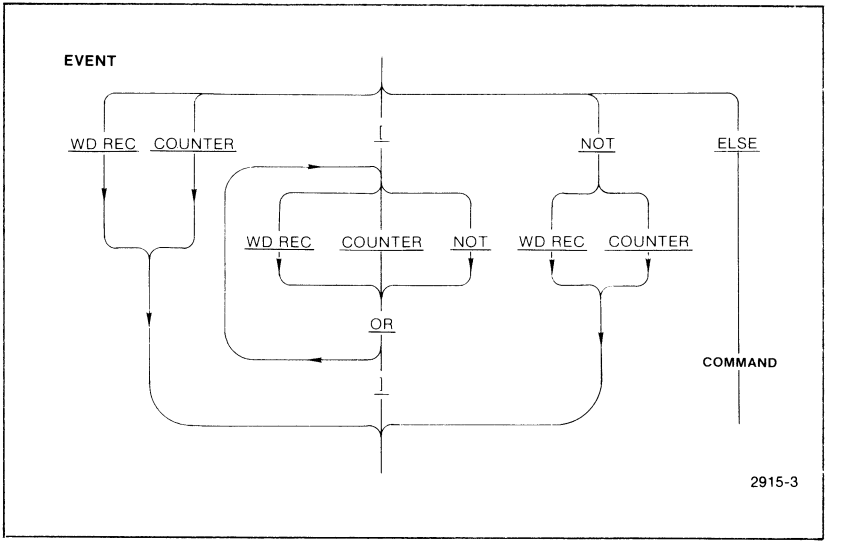
Data qualification can also be accomplished by using the QUALIFY key as a command within a TEST. In this case the data qualification is performed only when the TEST is active and the QUALIFY command is executed. If no QUALIFY block is specified and no QUALIFY commands are used, the 7D02 stores the data present on all state clocks. If either the QUALIFY block or the QUALIFY command is used, the 7D02 stores only when either the QUALIFY command is executed or the event within the QUALIFY block becomes TRUE.

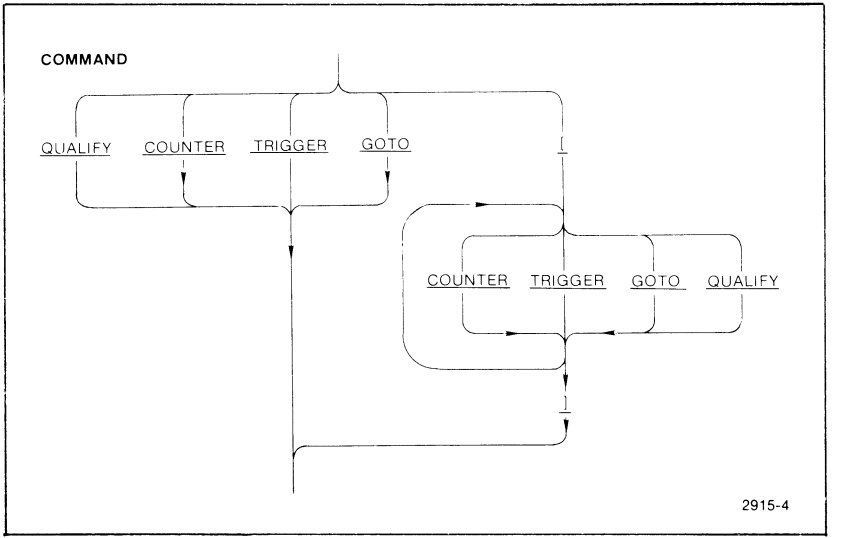
7D02 User Language Syntax



IMMEDIATE MODE




EVENT
WD REC
COUNTER
[
NOT
ELSE
WD REC
COUNTER
NOT
WD REC
COUNTER
OR
]
COMMAND
2915-3


COMMAND
QUALIFY
COUNTER
TRIGGER
GOTO
COUNTER
TRIGGER
GOTO
QUALIFY
2915-4

PROBLEM SOLVING

The ability to recognize events simultaneously and to execute various commands when these events become TRUE is a powerful debugging tool. The simultaneous recognition and execution does not lend itself to flow charts, however. The sequence of conditions leading up to the detection of a problem can usually be best expressed as a state transition diagram. The state transition diagram can then be translated directly into a 7D02 program.

A state transition diagram is used to describe a problem as a series of decision-making nodes or states. In each state the set of possible inputs is the same. The actions based on those inputs may vary from one state to another. Each state is independent of all of the other states in regard to the actions it may take based on the inputs. One of the actions taken may be to transition to another state or to the same state.

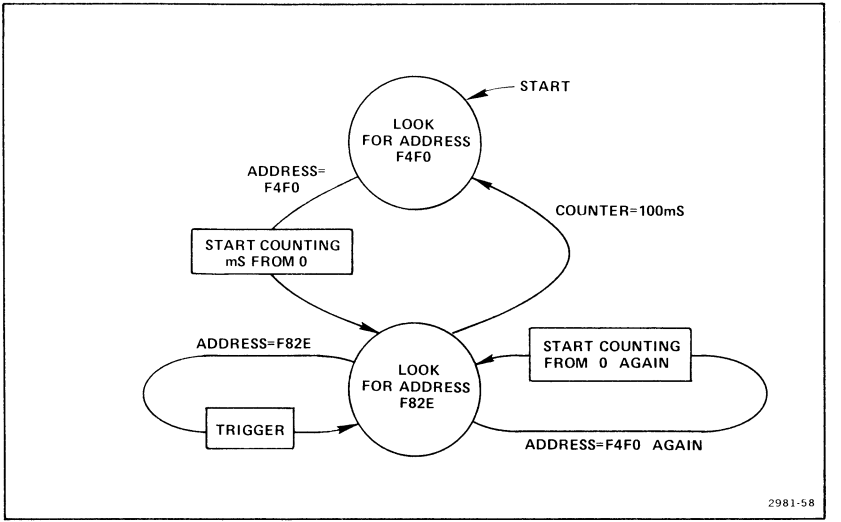
Each state is represented in the diagram as a bubble. Transitions between states are shown as arrows from the old state to the new; each arrow is labeled with the input that causes the transition. Actions associated with the input can be indicated by having the arrow pass through a labeled box on its way to its new state. All inputs that are not explicitly indicated are assumed to result in no action and no state change.

For example, it might be desired to trigger the Main Acquisition memory whenever address F90E is detected within 100 milliseconds after address F8D6 is detected.

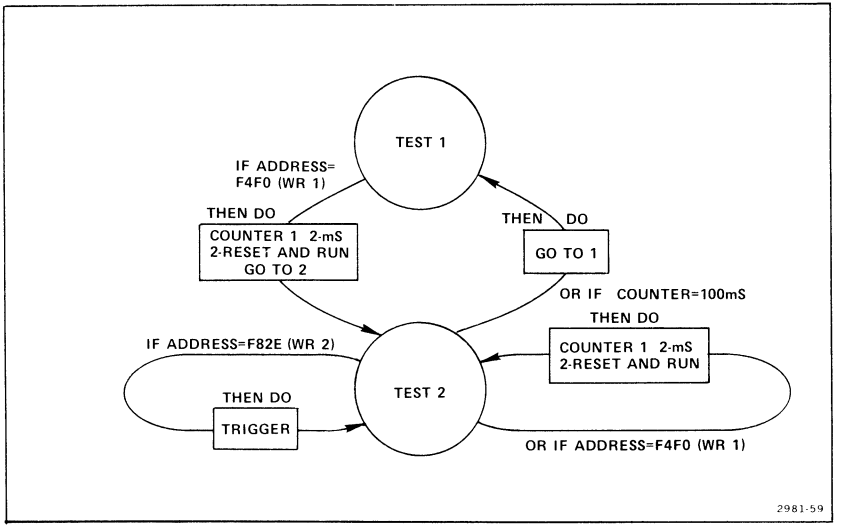
The first step would be to define a state in which the occurrence of F8D6 on the address bus is looked for. When this address is found, the counter would start to count the 100-millisecond time window.

The next step would be to look for address F90E. This must be done as part of a separate state since it is only valid as a triggering event when F90E is found **after** F8D6 is found (if they were both looked for in the same state, the trigger would occur any time F90E was found). If 100 milliseconds pass before address F90E is found, it is necessary to go back to state 1 and start the sequence over again. If F8D6 is detected a second time while counting the 100 millisecond time window, the window count must be restarted.

The labels on the arrows are the events to be looked for; the boxes are the actions to be taken. When an arrow leaves one state and enters a new one, a GOTO is necessary.


START
LOOK
FOR ADDRESS
F4F0
ADDRESS=
F4F0
COUNTER=100mS
START COUNTING
mS FROM 0
ADDRESS=F82E
LOOK
FOR ADDRESS
F82E
START COUNTING
FROM 0 AGAIN
TRIGGER
ADDRESS=F4F0 AGAIN
2981-58

Translating the labels into the 7D02 syntax gives:



The program can then be virtually read off the diagram to appear as follows:

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F4F0
1 /NMI=X /IRQ=X FETCH=X R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1
1 COUNTER # 1 2-MS
1    2-RESET AND RUN
1 GOTO 2
1
 END TEST 1
 TEST 2
2IF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F82E
2 /NMI=X /IRQ=X FETCH=X R/W=X
2 BA=X INVAL OP=X  EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2 TRIGGER 0-MAIN
2    0-BEFORE DATA
2    0-SYSTEM UNDER TEST CONT.
2    0-STANDARD CLOCK QUAL.
2OR IF
2 WORD RECOGNIZER # 1
2 DATA=XX
2 ADDRESS=F4F0
2 /NMI=X /IRQ=X FETCH-X R/W=X
2 BA=X INVAL OP=X  EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2 COUNTER # 1 2-MS
2    2-RESET AND RUN
2OR IF
2 COUNTER # 1 = 00100 2-MS
2THEN DO
2 GOTO 2
 END TEST 2
```

If the counter reaches the 100 ms time limit, go back to TEST 1 and start the sequence over.

Beginning address of sequence.

When the starting address is detected, start counting the 100 ms time period and go to state 2 to look for the second address in the sequence.

This is the second address in the sequence. If this is found before the counter reaches 100 ms, trigger the Main Acquisition Memory.

If the beginning address of the sequence is detected, reset the counter to begin the 100 ms count again.

2915-10

MICROPROCESSOR CONTROL LINES

In addition to data bus and address bus lines, microprocessors have control lines that are monitored by the 7D02. These control lines appear as part of the word recognizer display and vary according to the Personality Module attached to the 7D02.

Control lines can: (1) allow the microprocessor to output signals that control memory or other peripherals, (2) allow peripheral units to control the microprocessor (e.g., halt it or request an interrupt), or (3) identify the information on the address and data buses.

Most instructions in a microprocessor's instruction set require several cycles to execute. Usually each cycle corresponds to one bus transaction, one transfer of information into or out of the CPU. The "type" of each cycle is indicated by the values on the control lines. The first cycle in an instruction's execution is always a Fetch cycle (FETCH=1), i.e., the retrieval of an opcode for decoding and execution. In the 6802, many instructions contain more than one byte—the first is the opcode and the following are the operands. Additional read cycles (R/W=1) are needed to get the operand bytes into the CPU. Finally, Read (R/W=1) and Write (R/W=0) cycles are needed to carry out the instruction when its effect is to write data to and/or read data from memory.

SIMPLE WORD RECOGNITION

PROGRAM DESCRIPTION

TRIGGER when specified subroutine begins execution.

PROGRAM NOTES

The word recognizer becomes TRUE when an opcode Fetch is performed from address F8D6. The Main Acquisition Memory is then triggered. After 240 more words are acquired, the 7D02 will switch to Display mode.

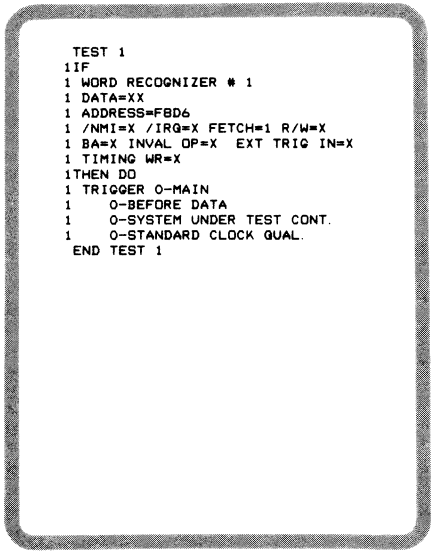
KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
Move cursor to address field. Enter "F8D6"

Move cursor to Fetch field. Enter "1".

TRIGGER

END


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=FBD6
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 TRIGGER O-MAIN
1 O-BEFORE DATA
1 O-SYSTEM UNDER TEST CONT.
1 O-STANDARD CLOCK QUAL.
END TEST 1

SIMPLE RANGE RECOGNITION USING DON'T CARES

PROGRAM DESCRIPTION

Trigger the 7D02 when the processor executes an instruction in the range 0-FFF (out of its normal program range).

PROGRAM NOTES

If the processor executes an instruction Fetch in an address between 0000 and 0FFF (this is out of the normal ROM space), the word recognizer becomes TRUE and the Main Acquisition Memory will trigger. 15 locations will be stored after the trigger (Trigger after data) and then the S.U.T. will halt.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
AD = 0XXX
Fetch = 1

TRIGGER
2 - After Data

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=0XXX
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 TRIGGER 0-MAIN
1     2-AFTER DATA
1     0-SYSTEM UNDER TEST CONT.
1     0-STANDARD CLOCK QUAL.
 END TEST 1
```

TRIGGER ON THE OCCURRENCE OF EITHER WORD A OR WORD B (USING BRACKETS)

PROGRAM DESCRIPTION

Trigger the 7D02 main acquisition section when either word A or word B is recognized.

PROGRAM NOTES

If the processor executes an instruction Fetch from either of two address locations (F872 or F89A) then one of the word recognizers will become TRUE and the 7D02 will trigger. 240 locations will be stored after the trigger (Trigger before data) and then the 7D02 will switch from Acquisition mode to Display mode.

KEYSTROKES *(IN SEQUENCE)*

[ ]

WORD RECOGNIZER #1
ADD = F872
FETCH = 1

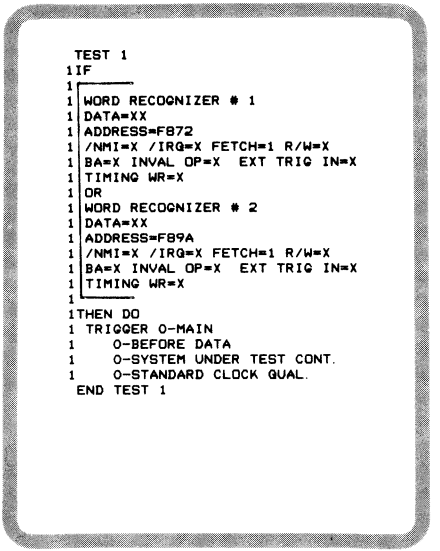
OR

WORD RECOGNIZER #2
ADD = F89A
FETCH = 1

[ ]

TRIGGER
Before Data
S.U.T. continue

END


TEST 1
IF
WORD RECOGNIZER # 1
DATA=XX
ADDRESS=F872
/NMI=X /IRQ=X FETCH=1 R/W=X
BA=X INVAL OP=X EXT TRIG IN=X
TIMING WR=X
OR
WORD RECOGNIZER # 2
DATA=XX
ADDRESS=F89A
/NMI=X /IRQ=X FETCH=1 R/W=X
BA=X INVAL OP=X EXT TRIG IN=X
TIMING WR=X
THEN DO
TRIGGER O-MAIN
O-BEFORE DATA
O-SYSTEM UNDER TEST CONT.
O-STANDARD CLOCK QUAL.
END TEST 1

TRIGGERING ON THE OCCURRENCE OF EITHER WORD A OR WORD B (USING OR IF)

PROGRAM DESCRIPTION

Trigger the 7D02 when either Word A or Word B begins execution (another way to do Example 3).

PROGRAM NOTES

If an instruction is fetched from address F872 the Main Acquisition Memory is triggered.

If an instruction is fetched from address F89A the Main Acquisition Memory is triggered.

Both addresses are looked for **simultaneously**.

KEYSTROKES *(IN SEQUENCE)*

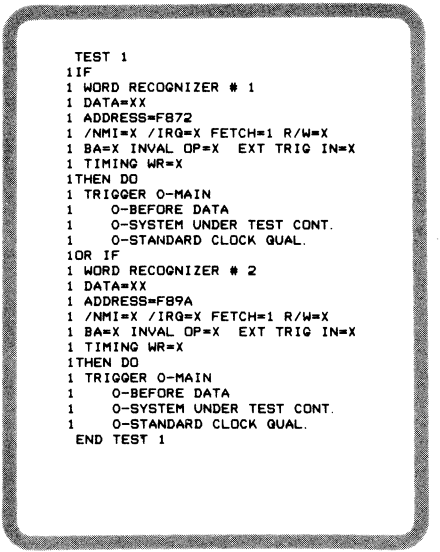
WORD RECOGNIZER #1
AD = F872
FETCH = 1

TRIGGER

WORD RECOGNIZER #2
AD = F89A
FETCH = 1

TRIGGER

END


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F872
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT.
1 0-STANDARD CLOCK QUAL.
1OR IF
1 WORD RECOGNIZER # 2
1 DATA=XX
1 ADDRESS=F89A
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT.
1 0-STANDARD CLOCK QUAL.
END TEST 1

COUNTING EVENTS

PROGRAM DESCRIPTION

Count the number of passes through a program loop until the program crashes.

PROGRAM NOTES

Any time that an opcode is fetched from address F89A (the beginning address of a loop) the counter is incremented. When an opcode is fetched from address F8D3 (this location is the crash point) the 7D02 will trigger.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
AD = F89A
FETCH = 1

COUNTER

WORD RECOGNIZER #2
AD = F8D3
FETCH = 1

TRIGGER

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F89A
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 COUNTER # 1 O-EVENTS
1     O-INCREMENT
1OR IF
1 WORD RECOGNIZER # 2
1 DATA=XX
1 ADDRESS=F8D3
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 TRIGGER O-MAIN
1     O-BEFORE DATA
1     O-SYSTEM UNDER TEST CONT.
1     O-STANDARD CLOCK QUAL.
 END TEST 1
```

TRIGGERING ON THE FIFTH OCCURRENCE OF A WORD

PROGRAM DESCRIPTION

Trigger on the fifth call to a specified program routine.

PROGRAM NOTES

Increment the counter whenever the first location (F934) of the routine is executed. On the cycle after the counter reaches 5, trigger the Main Acquisition Memory.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
AD = F934
FETCH = 1

COUNTER #1

COUNTER #1
00005

TRIGGER
2 - After Data

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F934
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 COUNTER # 1 0-EVENTS
1     0-INCREMENT
1OR IF
1 COUNTER # 1 = 00005 0-EVENTS
1THEN DO
1 TRIGGER 0-MAIN
1     2-AFTER DATA
1     0-SYSTEM UNDER TEST CONT.
1     0-STANDARD CLOCK QUAL.
 END TEST 1
```

MEASURING ELAPSED TIME BETWEEN TWO EVENTS

PROGRAM DESCRIPTION

Start a timer on the first occurrence of a word; stop the timer and trigger on the first occurrence of a different word.

PROGRAM NOTES

Counter #1 acts as a timer, counting microseconds. It is started when Word Recognizer #1 detects an instruction fetch from F95A (the start of the wait loop). When Word Recognizer #2 detects an instruction fetch from F984 (the end of the wait loop), the counter is stopped and the 7D02 is triggered. Note that a command has been included to simultaneously transfer to Test 2, an empty test. This is to make sure that if Word Recognizer #1 comes TRUE again after the trigger (while the final 240 words of data are being stored) it will not reset the counter.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER # 1
AD = F95A
FETCH = 1

COUNTER # 1
1 - μS
2 - Reset and Run

WORD RECOGNIZER # 2
AD = F984
FETCH = 1

[ ]

COUNTER # 1
1 - Stop

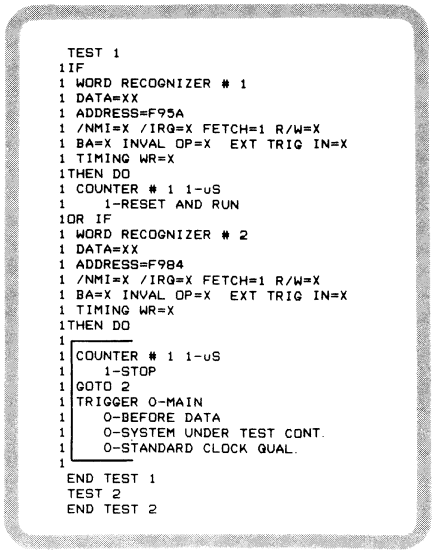
GOTO

TRIGGER

[ ]

END Test 1

END Test 2


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F95A
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 COUNTER # 1 1-uS
1 1-RESET AND RUN
1OR IF
1 WORD RECOGNIZER # 2
1 DATA=XX
1 ADDRESS=F984
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1
1 COUNTER # 1 1-uS
1 1-STOP
1 GOTO 2
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT.
1 0-STANDARD CLOCK QUAL.
1
END TEST 1
TEST 2
END TEST 2

TRIGGER AFTER SPECIFIED TIME

PROGRAM DESCRIPTION

Trigger the 7D02 after a specific amount of time has elapsed.

PROGRAM NOTES

Counter #1 will start on the first Qualified Clock. The counter will run, and the 7D02 will acquire data, until the counter reaches 5000 ms (5 seconds). Then the 7D02 will trigger.

KEYSTROKES *(IN SEQUENCE)*

COUNTER #1
05000 2 - MS

TRIGGER

ELSE

COUNTER

END

```
 TEST 1
1IF
1 COUNTER # 1 = 00005 2-MS
1THEN DO
1 TRIGGER O-MAIN
1     O-BEFORE DATA
1     O-SYSTEM UNDER TEST CONT.
1     O-STANDARD CLOCK QUAL.
1ELSE DO
1 COUNTER # 1 2-MS
1     O-RUN
 END TEST 1
```

NOT WORD RECOGNITION

PROGRAM DESCRIPTION

Trigger when the data written to a particular location is not what is expected.

PROGRAM NOTES

This program will trigger when any value other than 03 is written to address 0004.

"NOT Word Recognizer" #1 is true when a write to location 0004 is **not** taking place. Word Recognizer #2 is true when the value on the data bus is 03. The compound (bracketed) event is, therefore, true when 0004 is not being written to, or the value being written is a 03.

Since there is only 1 test in the program, the "GOTO 1" command associated with the event is essentially a "no-operation".

Since the "trigger" command is associated with the "Else", it will be executed when the compound event is **not** true; i.e., when a write data is done to 0004 and the value on the data bus is **not** 03.

Note: This example takes advantage of DeMorgan's theorem:

$A \wedge B = (A) \vee (B)$ i.e.,

(Write to 0004) and (data $\neq$ 3) = (Write to 4) or (data = 3)

KEYSTROKES *(IN SEQUENCE)*

NOT

WORD RECOGNIZER #1
AD = 0004
R/W = 0

OR

WORD RECOGNIZER #2
DATA = 03

[ ]

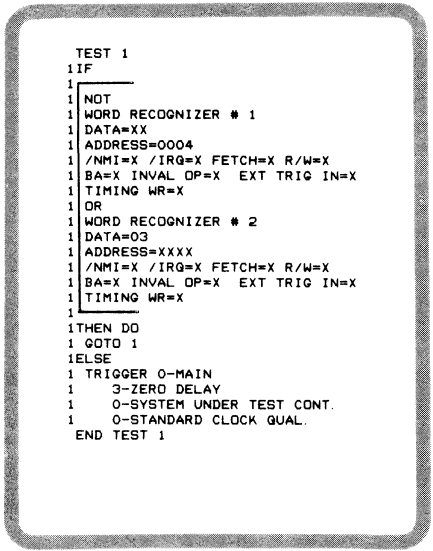
GOTO

1

ELSE

TRIGGER
3 - Zero Delay

END


TEST 1
1IF
1
1 NOT
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=0004
1 /NMI=X /IRQ=X FETCH=X R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1 OR
1 WORD RECOGNIZER # 2
1 DATA=03
1 ADDRESS=XXXX
1 /NMI=X /IRQ=X FETCH=X R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1
1THEN DO
1 GOTO 1
1ELSE
1 TRIGGER 0-MAIN
1 3-ZERO DELAY
1 0-SYSTEM UNDER TEST CONT.
1 0-STANDARD CLOCK QUAL.
END TEST 1

2-WORD SEQUENTIAL WORD RECOGNITION

PROGRAM DESCRIPTION

Trigger when a particular word is detected any time following another word.

PROGRAM NOTES

Word Recognizer 1 comes true when the first instruction (address F805) is executed. It causes Test 2, in which the second word in the sequence is sought, to be activated.

The 7D02 then triggers when Word Recognizer 2 detects the execution of a transfer to address F81C.

240 words of data are stored after the trigger. (We trigger "before data".)

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
AD = F805
FETCH = 1

WORD RECOGNIZER #2
AD = F81C
FETCH = 1

TRIGGER

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F805
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 GOTO 2
 END TEST 1
 TEST 2
2IF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F81C
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X  EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2 TRIGGER 0-MAIN
2    0-BEFORE DATA
2    0-SYSTEM UNDER TEST CONT.
2    0-STANDARD CLOCK QUAL.
 END TEST
```

SEQUENTIAL WORD RECOGNITION WITH COUNTERS

PROGRAM DESCRIPTION

Triggering on a specific iteration of a nested loop.

PROGRAM NOTES

Counter 1 is incremented whenever the instruction at location F966 is executed. This instruction is in the outer loop.

After 123 executions, the "GOTO" command causes Test 2 to be entered, to look for the next word in the sequence.

Counter 2 is incremented every time Word Recognizer #2 detects an instruction fetch from F96B. This instruction is in the nested loop.

After 69 executions of the nested loop, the 7D02 is triggered.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
AD - F966
FETCH - 1

COUNTER #1
0 - Increment

COUNTER #1
000123 0 - Events

GOTO

END

WORD RECOGNIZER #2
AD - F96B
FETCH - 1

COUNTER #2
0 - Increment

COUNTER #2
00069 0 - Events

TRIGGER

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F966
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 COUNTER # 1 0-EVENTS
1OR IF
1 COUNTER # 1 = 00123 0-EVENTS
1THEN DO
1 GOTO 2
 END TEST 1
 TEST 2
2IF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F96B
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X  EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2 COUNTER # 2 0-EVENTS
2OR IF
2 COUNTER # 2 = 00069 0-EVENTS
2THEN DO
2 TRIGGER 0-MAIN
```

```
2    0-BEFORE DATA
2    0-SYSTEM UNDER TEST CONT.
2    0-STANDARD CLOCK QUAL.
 END TEST 2
```

ESTABLISHING A TEMPORAL "WINDOW" FOR TRIGGERING

PROGRAM DESCRIPTION

Trigger when the time spent in a subroutine reaches a specified value.

PROGRAM NOTES

Test 1 looks for the beginning of execution of a subroutine. (An instruction fetch from F8D6.) It starts a timer (Counter #1 in millisecond mode) when the Word Recognizer comes true.

Test 2 triggers (and goes to Test 3) if 17 MS elapse. If the end of the subroutine is reached (fetch from F90E) before the 17 MS elapses, it returns to Test 1. (Note: There is no trigger command in Test 1—this effectively "closes" the triggering window.)

Note that if the Counter reaches 17 MS on the same cycle that the word recognizer comes true in Test 2, the "GOTO 3" will be executed instead of the "GOTO 1" because the commands are contradictory and the GOTO 3 appears first in the program.

Test 3 does nothing. This assures us the Counter will not be reset in Test 1 after the trigger occurs, and data is still being acquired. (This is not strictly necessary in this example.)

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
AD = F8D6
FETCH = 1

[ ]

COUNTER #1
2 MS
2 - Reset and Run

GOTO

[ ]

END

COUNTER #1
00017 2 MS

[ ]

TRIGGER

GOTO

[ ]

WORD RECOGNIZER #2
AD = F90E
FETCH = 1

GOTO 1

END

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F8D6
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 ┌──────
1 │COUNTER # 1 2-MS
1 │   2-RESET AND RUN
1 │GOTO 2
1 └──────
 END TEST 1
 TEST 2
2IF
2 COUNTER # 1 = 00017 2-MS
2THEN DO
2 ┌──────
2 │TRIGGER 0-MAIN
2 │   0-BEFORE DATA
2 │   0-SYSTEM UNDER TEST CONT.
2 │   0-STANDARD CLOCK QUAL.
2 │GOTO 3
2 └──────
2ORIF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F90E
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X  EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2 GOTO 1
 END TEST 2
 TEST 3
 END TEST 3
```

TRIGGER WHEN A TIME PERIOD IS LESS THAN EXPECTED

PROGRAM DESCRIPTION

Trigger if a particular subroutine finishes in less than a specified period of time.

PROGRAM NOTES

In this program, Test 1 detects address F8D6, starts the counter and enters Test 2. Test 2 looks for address F90E, stops the counter and enters Test 3. In Test 3, if the counter is not equal to 200 MS (the normal time period for the subroutine to execute) the Main Acquisition Memory is triggered. Otherwise, Test 1 is entered and the process starts over again.

NOTE: The same condition would be detected by removing the **NOT** before the counter and switching the GOTO and trigger commands in Test 3.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
Address = F8D6
Fetch = 1

[ ]

COUNTER #1
2 MS
2 - Reset and Run

GOTO

[ ]

END

WORD RECOGNIZER #2
Address = F90E
Fetch = 1

[ ]

COUNTER #1

GOTO

[ ]

END

NOT

COUNTER #1
00200

[ ]

TRIGGER

ELSE

GOTO 1

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F8D6
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1
1 COUNTER # 1 2-MS
1    2-RESET AND RUN
1 GOTO 2
1
 END TEST 1
 TEST 2
2IF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F90E
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X  EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2
2 COUNTER # 1 2-MS
2    1-STOP
2 GOTO 3
2
 END TEST 2
```

```
 TEST 3
3IF
3
3 NOT
3 COUNTER # 1 = 00200 2-MS
3
3THEN DO
3 TRIGGER 0-MAIN
3     0-BEFORE DATA
3     0-SYSTEM UNDER TEST CONT.
3     0-STANDARD CLOCK QUAL.
3ELSE DO
3 GOTO 1
 END TEST 3
```

FIRMWARE PERFORMANCE ANALYSIS

PROGRAM DESCRIPTION

Measure the time spent in a subroutine in units of time **and** as a percentage of the total time spent in the calling routine.

PROGRAM NOTES

Address F872 is the start of a routine. When this address is detected, start Counter #1 and go to Test 2.

Address F95A is the start of the routine which performs a wait loop. Word Recognizer #2 will cause Counter 2 to start counting time in the loop and then go to Test 3 to look for the end of the loop.

Address F984 is the end of the wait loop. When this is detected, Counter 2 is stopped, giving the elapsed time in the wait loop. A goto to Test 4 is executed to look for the end of the first routine.

Address F899 is the address of the end of the first routine. When this address is detected, Counter 1 is stopped (giving the total elapsed time) and the acquisition memory is triggered.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
AD = F872
FETCH = 1

[ ]

COUNTER #1
1 μS
0 - Run

GOTO

[ ]

END

WORD RECOGNIZER #2
AD = F95A
FETCH = 1

[ ]

COUNTER #2
1 μS
0 - Run

GOTO

[ ]

END

WORD RECOGNIZER #3
AD = F984
FETCH = 1

[ ]

COUNTER #2
1 μS
1 Stop

GOTO

[ ]

END

WORD RECOGNIZER #4
AD = F899
FETCH = 1

[ ]

COUNTER #1
1 μS
1 - Stop

TRIGGER

[ ]

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F872
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1┌──────
1│COUNTER # 1 1-uS
1│   0-RUN
1│GOTO 2
1└──────
 END TEST 1
 TEST 2
2IF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F95A
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X  EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2┌──────
2│COUNTER # 2 1-uS
2│   0-RUN
2│GOTO 2
2└──────
 END TEST 2
```

```
 TEST 3
3IF
3 WORD RECOGNIZER # 3
3 DATA=XX
3 ADDRESS=F984
3 /NMI=X /IRQ=X FETCH=1 R/W=X
3 BA=X INVAL OP=X  EXT TRIG IN=X
3 TIMING WR=X
3THEN DO
3┌──────
3│COUNTER # 2 1uS
3│   1-STOP
3│GOTO 4
3└──────
 END TEST 3
 TEST 4
4IF
4 WORD RECOGNIZER # 4
4 DATA=XX
4 ADDRESS=F899
4 /NMI=X /IRQ=X FETCH=1 R/W=X
4 BA=X INVAL OP=X  EXT TRIG IN=X
4 TIMING WR=X
4THEN DO
4┌──────
4│COUNTER # 1 1-uS
4│   1-STOP
4│TRIGGER 0-MAIN
4│   0-BEFORE DATA
4│   0-SYSTEM UNDER TEST CONT.
4│   0-STANDARD CLOCK QUAL.
4└──────
 END TEST 4
```

LOCATING THE SOURCE OF AN UNKNOWN JUMP TO A SUBROUTINE

PROGRAM DESCRIPTION

Detect all fetches of instructions that cannot validly jump to a subroutine. Trigger when one of them is followed by a fetch from the first instruction in a subroutine.

PROGRAM NOTES

This program will trigger the Main Acquisition Memory whenever the subroutine at address F90F is called from any address other than F903. When Test 1 detects an Opcode fetch from address F903, it transitions to State 2. This makes use of the fact that the next fetch cycle after F903 will always be the beginning of the subroutine. When the next fetch is detected, Test 2 transitions back to Test 1. If the subroutine is called from anywhere but F903, Test 1 will be active, will detect the call to the subroutine in Word Recognizer #1, and trigger. Note that Word Recognizer #1 could be used instead of Word Recognizer #3 in Test 2.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
Address = F90F
Fetch = 1

TRIGGER

WORD RECOGNIZER #2
Address = F903
Fetch = 1

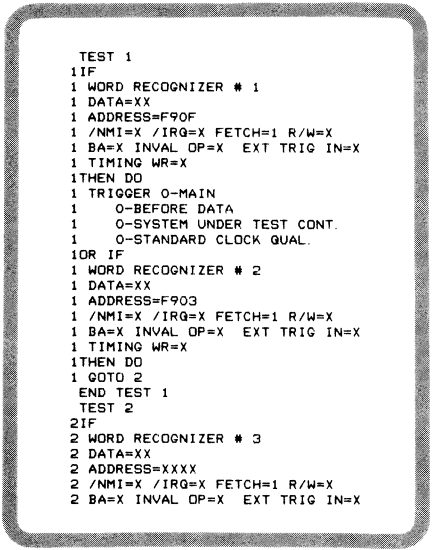
GOTO

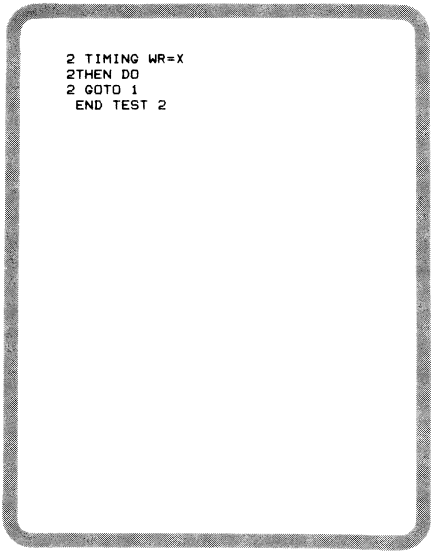
END

WORD RECOGNIZER #3
Fetch = 1

GOTO
1

END


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F90F
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT.
1 0-STANDARD CLOCK QUAL.
1OR IF
1 WORD RECOGNIZER # 2
1 DATA=XX
1 ADDRESS=F903
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 GOTO 2
END TEST 1
TEST 2
2IF
2 WORD RECOGNIZER # 3
2 DATA=XX
2 ADDRESS=XXXX
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X EXT TRIG IN=X


2 TIMING WR=X
2THEN DO
2 GOTO 1
END TEST 2

STORAGE MEMORY COMPARISON FOR MAIN ACQUISITION DATA

PROGRAM DESCRIPTION

Trigger on a write to an address. Store the acquired data in the storage memory and compare that data with a new acquisition.

PROGRAM NOTES

The Main Acquisition Memory is triggered when a write is performed to address C000. When the 7D02 returns to display mode the keystroke sequence **IMMEDIATE STORE MEM ACQ MEM →** will store the contents of the Acquisition Memory into the Storage Memory. By entering format mode the user can enable memory difference highlighting. By pressing **START** new data will be acquired into the Acquisition Memory and any differences will be highlighted on the display.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
ADDRESS = C000
R/W = 0

TRIGGER

END

START 7D02

IMMEDIATE STORE MEM ACQ MEM →

FORMAT
Select "yes" for highlight memory differences

FORMAT

START 7D02

```
 TEST 1
1IF
1 WORD RECOGNZER # 1
1 DATA=XX
1 ADDRESS=0000
1 /NMI=X /IRQ=X FETCH=X R/W=0
1 BA=X INVAL OP=X  EXT TRIG IN=X
1THEN DO
1 TRIGGER 0-MAIN
1     0-BEFORE DATA
1     0-SYSTEM UNDER TEST CONT.
1     0-STANDARD CLOCK QUAL.
 END TEST 1
```

BLOCK QUALIFICATION OF DATA—1

PROGRAM DESCRIPTION

Trigger when C000 is written to. Store **only** those cycles on which C000 is written to.

PROGRAM NOTES

Test 1 triggers as soon as a write to C000 takes place.

Inclusion of a Qualify block causes data to be stored into Acquisition Memory **only** when the event specified in the block comes true.

We reused Word Recognizer #1, but any word recognizer could have been used in the Qualification Block.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
AD = C000
R/W = 0

TRIGGER

END

QUALIFY

WORD RECOGNIZER #1
(use WR #1 again)

END

FORMAT
Set Highlight Memory Differences to 1 - NO
Set Data Display field to 3 - ASCII

FORMAT

START

```
 TEST 1
1IF
1 WORD RECOGNZER # 1
1 DATA=XX
1 ADDRESS=C000
1 /NMI=X /IRQ=X FETCH=X R/W=0
1 BA=X INVAL OP=X  EXT TRIG IN=X
1THEN DO
1 TRIGGER 0-MAIN
1     0-BEFORE DATA
1     0-SYSTEM UNDER TEST CONT.
1     0-STANDARD CLOCK QUAL.
 END TEST 1
 QUALIFY
QSTORE ON
Q WORD RECOGNIZER # 1
Q DATA=XX
Q ADDRESS=C000
Q /NMI=X /IRQ=X FETCH=X R/W=0
Q BA=X INVAL OP=X  EXT TRIG IN=X
Q TIMING WR=X
 END QUALIFY
```

BLOCK QUALIFICATION OF DATA—2

PROGRAM DESCRIPTION

Trigger on a write to C000; store data **only** when C000 is written to **and** bits 6—5 of the data are 10 or 01.

PROGRAM NOTES

Test 1 triggers the 7D02 as soon as a write to C000 takes place.

Inclusion of the Qualify Block causes data to be stored **only** when the compound event in this block is true. Word Recognizer #1 is true when data with bits 6-5 = 10 is written to C000. Word Recognizer #2 is true when data with bits 6-5 = 01 is written to C000. The event is true when either Word Recognizer #2 or Word Recognizer #3 is true.

KEYSTROKES *(IN SEQUENCE)*

FORMAT

WORD RECOGNIZER #1
AD = C000
R/W = 0

TRIGGER

END

QUALIFY

[ ]

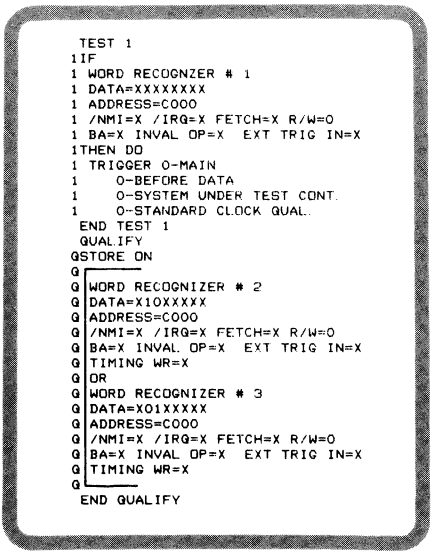
WORD RECOGNIZER #2
DATA = X10XXX-XX
AD = C000
R/W = 0

OR

WORD RECOGNIZER #3
DATA = X01XXX-XX
AD = C000
R/W = 0

[ ]

END


TEST 1
1IF
1 WORD RECOGNZER # 1
1 DATA=XXXXXXXX
1 ADDRESS=C000
1 /NMI=X /IRQ=X FETCH=X R/W=0
1 BA=X INVAL OP=X EXT TRIG IN=X
1THEN DO
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT.
1 0-STANDARD CLOCK QUAL.
END TEST 1
QUALIFY
QSTORE ON
Q
Q WORD RECOGNIZER # 2
Q DATA=X10XXXXX
Q ADDRESS=C000
Q /NMI=X /IRQ=X FETCH=X R/W=0
Q BA=X INVAL OP=X EXT TRIG IN=X
Q TIMING WR=X
Q OR
Q WORD RECOGNIZER # 3
Q DATA=X01XXXXX
Q ADDRESS=C000
Q /NMI=X /IRQ=X FETCH=X R/W=0
Q BA=X INVAL OP=X EXT TRIG IN=X
Q TIMING WR=X
Q
END QUALIFY

BLOCK QUALIFICATION OF DATA—3

PROGRAM DESCRIPTION

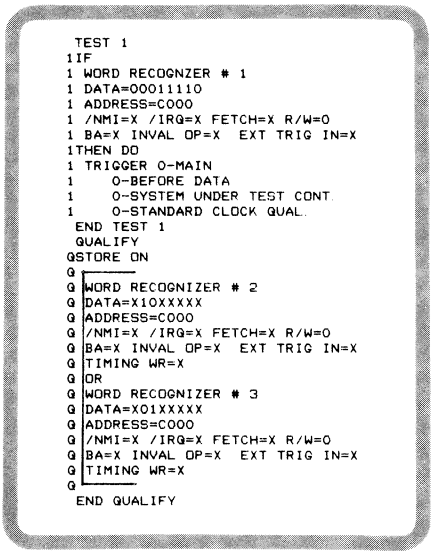
Trigger when 00011110 (binary is written to C000). Store **only** those cycles on which C000 is written to and bits 6-5 of the data are 10 or 01.

PROGRAM NOTES

This program is identical to that in the previous example, except that the trigger occurs when a synchronizing control character (1E) is seen, indicating the beginning of the message.

KEYSTROKES *(IN SEQUENCE)*

Use the same procedure as in the previous example, except set the Data Field of Word Recognizer #1 to 00011110.


TEST 1
1IF
1 WORD RECOGNZER # 1
1 DATA=00011110
1 ADDRESS=C000
1 /NMI=X /IRQ=X FETCH=X R/W=0
1 BA=X INVAL OP=X EXT TRIG IN=X
1THEN DO
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT
1 0-STANDARD CLOCK QUAL
END TEST 1
QUALIFY
QSTORE ON
Q
Q WORD RECOGNIZER # 2
Q DATA=X10XXXXX
Q ADDRESS=C000
Q /NMI=X /IRQ=X FETCH=X R/W=0
Q BA=X INVAL OP=X EXT TRIG IN=X
Q TIMING WR=X
Q OR
Q WORD RECOGNIZER # 3
Q DATA=X01XXXXX
Q ADDRESS=C000
Q /NMI=X /IRQ=X FETCH=X R/W=0
Q BA=X INVAL OP=X EXT TRIG IN=X
Q TIMING WR=X
Q
END QUALIFY

3-WORD SEQUENCE WORD-RECOGNITION WITH QUALIFY

PROGRAM DESCRIPTION

Trigger on a 3-word sequential word recognition and store the occurrence of each of the word recognitions.

PROGRAM NOTES

In this example, a sequence of three words is being detected. The first word, an instruction fetch from address F82B, is the beginning of a routine. When it is found, the word is qualified so that it can be seen in the acquired data. Test 2 is then entered to look for the occurrence of the second word, an instruction fetch from address F89A. This is the beginning of a loop. This is also qualified. Test 3 is then entered to look for the third word, an instruction fetch from address F8D3. This is the "crash point" in the program. This is also qualified, the Main Acquisition Memory is triggered and Test 4 is entered. Test 4 stores all cycles until the 7D02 enters display mode.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
ADDRESS = F82B
FETCH = 1

[ ]

QUALIFY

GOTO

[ ]

END

WORD RECOGNIZER #2
ADDRESS = F89A
FETCH = 1

[ ]

QUALIFY

GOTO

[ ]

END

WORD RECOGNIZER #3
ADDRESS = F8D3
FETCH = 1

[ ]

QUALIFY

TRIGGER

GOTO

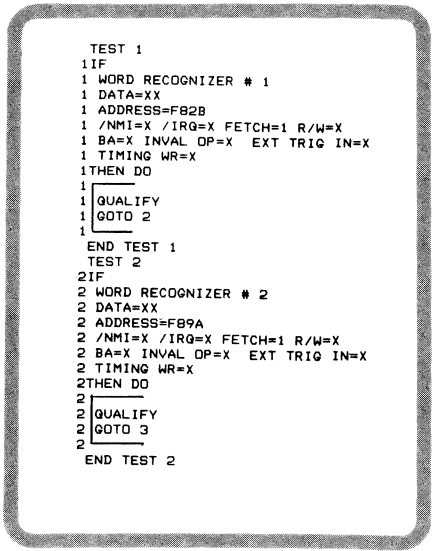
[ ]

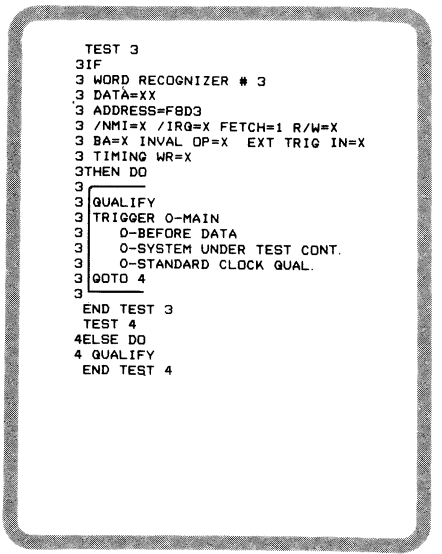
END

ELSE

QUALIFY

END


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F82B
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1
1 QUALIFY
1 GOTO 2
1
END TEST 1
TEST 2
2IF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F89A
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2
2 QUALIFY
2 GOTO 3
2
END TEST 2


TEST 3
3IF
3 WORD RECOGNIZER # 3
3 DATA=XX
3 ADDRESS=F8D3
3 /NMI=X /IRQ=X FETCH=1 R/W=X
3 BA=X INVAL OP=X EXT TRIG IN=X
3 TIMING WR=X
3THEN DO
3
3 QUALIFY
3 TRIGGER 0-MAIN
3 0-BEFORE DATA
3 0-SYSTEM UNDER TEST CONT.
3 0-STANDARD CLOCK QUAL.
3 GOTO 4
3
END TEST 3
TEST 4
4ELSE DO
4 QUALIFY
END TEST 4

DATA QUALIFICATION USING A COUNTER

PROGRAM DESCRIPTION

Qualify data acquired after a counter reaches a specific value.

PROGRAM NOTES

Address F849 is a location within a subroutine. When it is detected, the data is qualified (causing each fetch from address F849 to be stored in the Acquisition Memory) and the counter is incremented. When the counter reaches 5, the Main Acquisition Memory is triggered and all data (until storage is complete) is qualified. This happens because once the counter reaches 5, it becomes true and stays true until the 7D02 stops acquiring data or the counter is reset (which does not happen in this program).

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
ADDRESS = F849
FETCH = 1

[ ]

COUNTER #1

QUALIFY

[ ]

COUNTER #1
00005

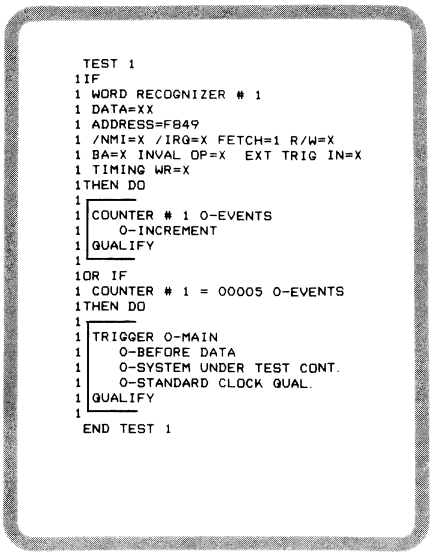
[ ]

TRIGGER

QUALIFY

[ ]

END


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F849
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1
1 COUNTER # 1 0-EVENTS
1 0-INCREMENT
1 QUALIFY
1
1OR IF
1 COUNTER # 1 = 00005 0-EVENTS
1THEN DO
1
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT.
1 0-STANDARD CLOCK QUAL.
1 QUALIFY
1
END TEST 1

QUALIFY BETWEEN OCCURRENCES OF 2 WORDS

PROGRAM DESCRIPTION

Store data only when a particular section of the program is being executed.

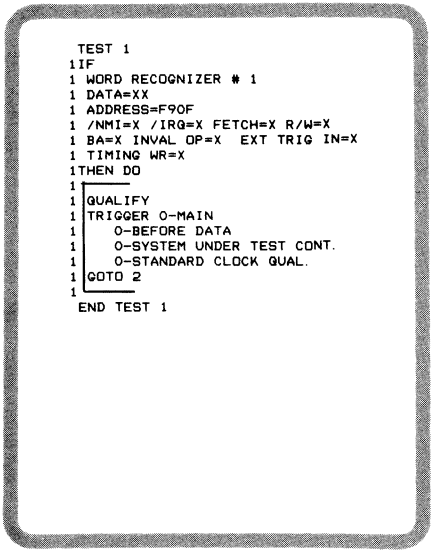
PROGRAM NOTES

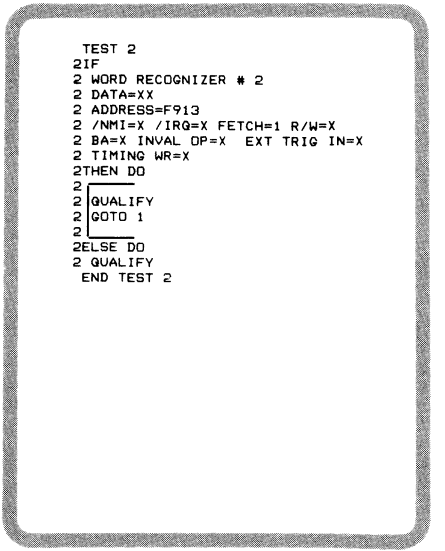
This program will store data between the execution of the beginning of the routine (address F90F) and the execution of the last address of the routine (address F913). Word Recognizer #1 detects the beginning of the routine, qualifies the first address, triggers the Main Acquisition Memory and then enters Test 2. Test 2 detects the end of the routine. If it is found, it is qualified and then Test 1 is entered (starting the process over). In Test 2 all cycles are qualified by the else clause until the word recognizer becomes true. This process is repeated after the trigger is executed until 240 acquisition memory locations have been filled. Note that the 7D02 will be in Test 1 whenever the display routine is not being executed and will be in Test 2 whenever it is being executed.

Note also that the 7D02 can only be triggered once. Thus the second and subsequent executions of the Trigger command in Test 1 will be ignored, but the associated Qualify and Goto execute normally.

KEYSTROKES *(IN SEQUENCE)*

```
WORD RECOGNIZER #1
    ADDRESS = F90F
    FETCH = 1
[ ]
QUALIFY
TRIGGER
GOTO
[ ]
END
WORD RECOGNIZER #2
    ADDRESS = F913
    FETCH = 1
[ ]
QUALIFY
GOTO
    1
[ ]
ELSE
QUALIFY
END
```


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F90F
1 /NMI=X /IRQ=X FETCH=X R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1
1 QUALIFY
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT.
1 0-STANDARD CLOCK QUAL.
1 GOTO 2
1
END TEST 1


TEST 2
2IF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F913
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2
2 QUALIFY
2 GOTO 1
2
2ELSE DO
2 QUALIFY
END TEST 2

QUALIFY OUTSIDE OF OCCURRENCES OF 2 WORDS

PROGRAM DESCRIPTION

Always store data **except** when a particular section of the program is being executed.

PROGRAM NOTES

This program waits for the beginning of a routine (address F805) to be executed. When this is detected (Word Recognizer #1 becomes true) the Main Acquisition Memory is triggered, the first address in the routine is stored and Test 2 is entered. Test 2 looks for the beginning of a loop to be executed (address F8D6). If it is not (meaning the processor is not executing that loop) then the data is stored by the qualify command in the else clause. If the beginning of the loop is executed, Test 3 is entered. Test 3 waits for the end of the loop to be executed. No data is stored while the processor is executing the loop. When the end is detected, Test 2 is entered so that data may again be qualified.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
ADDRESS = F805
FETCH = 1

[ ]

TRIGGER

QUALIFY

GOTO

[ ]

END

WORD RECOGNIZER #2
ADDRESS = F8D6
FETCH = 1

GOTO

ELSE

QUALIFY

END

WORD RECOGNIZER #3
ADDRESS = F90E
FETCH = 1

GOTO
2

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F805
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 ┌──────
1 │TRIGGER 0-MAIN
1 │   0-BEFORE DATA
1 │   0-SYSTEM UNDER TEST CONT
1 │   0-STANDARD CLOCK QUAL
1 │QUALIFY
1 │GOTO 2
1 └──────
 END TEST 1
 TEST 2
2IF
2 WORD RECOGNIZER # 2
2 DATA=XX
2 ADDRESS=F8D6
2 /NMI=X /IRQ=X FETCH=1 R/W=X
2 BA=X INVAL OP=X  EXT TRIG IN=X
2 TIMING WR=X
2THEN DO
2 GOTO 3
2ELSE DO
2 QUALIFY
 END TEST 2
```

```
 TEST 3
3IF
3 WORD RECOGNIZER # 3
3 DATA=XX
3 ADDRESS=F90E
3 /NMI=X /IRQ=X FETCH=1 R/W=X
3 BA=X INVAL OP=X  EXT TRIG IN=X
3 TIMING WR=X
3THEN DO
3 GOTO 2
 END TEST 3
```

ASYNCHRONOUS WORD RECOGNITION AND TRIGGERING

PROGRAM DESCRIPTION

Arm the Asynchronous Timing Option to perform its own word recognition and triggering.

PROGRAM NOTES

Because no events are specified for the main section, the else clause will be executed on the first 7D02 cycle, causing the Trigger command to be executed. Because the Timing Option is running asynchronously, the Trigger command merely **arms** the Timing Option to trigger on its own word recognizer.

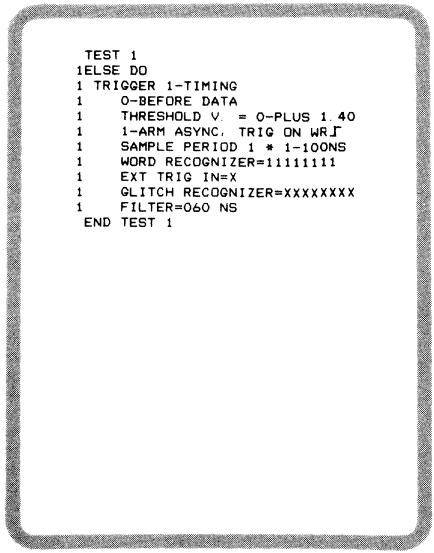
The word recognizer runs asynchronously to the data sampling. The filter value selected here helps to assure that the data values which cause the word recognizer to come true will actually be sampled and stored.

KEYSTROKES *(IN SEQUENCE)*

ELSE

TRIGGER
1 - Timing
Word Recognizer = 11111111
Filter = 60 NS

END


TEST 1
IELSE DO
1 TRIGGER 1-TIMING
1 0-BEFORE DATA
1 THRESHOLD V. = 0-PLUS 1.40
1 1-ARM ASYNC. TRIG ON WR
1 SAMPLE PERIOD 1 * 1-100NS
1 WORD RECOGNIZER=11111111
1 EXT TRIG IN=X
1 GLITCH RECOGNIZER=XXXXXXXX
1 FILTER=060 NS
END TEST 1

ASYNCHRONOUS GLITCH RECOGNITION AND TRIGGERING

PROGRAM DESCRIPTION

Arm the Asynchronous Timing Option to trigger on its own glitch recognition.

PROGRAM NOTES

This program is identical to the previous one, except a different sampling rate has been selected, and triggering is dependent on a glitch being detected on channel 5.

KEYSTROKES *(IN SEQUENCE)*

ELSE

TRIGGER
1 - Timing
Sample Period =
2 μS
Word Recognizer =
XXXXXXXX
Glitch Recognizer
=
XX1XXXXX

END

```
 TEST 1
1ELSE DO
1 TRIGGER 1-TIMING
1     0-BEFORE DATA
1     THRESHOLD V.  = 0-PLUS 1.40
1     1-ARM ASYNC, TRIG ON WR∫
1     SAMPLE PERIOD 1 * 2-1 uS
1     WORD RECOGNIZER=11111111
1     EXT TRIG IN=X
1     GLITCH RECOGNIZER=XX1XXXXX
1     FILTER=060 NS
 END TEST 1
```

MAIN SECTION TRIGGERS ASYNCHRONOUS TIMING OPTION

PROGRAM DESCRIPTION

Trigger the Timing Option after a word recognition in the main section.

PROGRAM NOTES

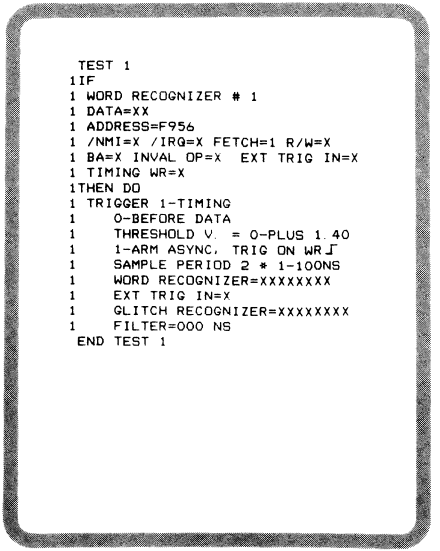
The program arms the timing option when an opcode is fetched from address F956. The timing option is running asynchronously at 200 ns per sample. Since the timing option word recognizer and glitch recognizer are set to all don't cares, the timing option will trigger as soon as it is armed.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
ADDRESS = F956
FETCH = 1

TRIGGER
1 - Timing
Sample Period = 200 NS
Word Recognizer = XXXXXXXX
Glitch Recognizer = XXXXXXXX

END


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F956
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 TRIGGER 1-TIMING
1 0-BEFORE DATA
1 THRESHOLD V. = 0-PLUS 1.40
1 1-ARM ASYNC, TRIG ON WR
1 SAMPLE PERIOD 2 * 1-100NS
1 WORD RECOGNIZER=XXXXXXXX
1 EXT TRIG IN=X
1 GLITCH RECOGNIZER=XXXXXXXX
1 FILTER=000 NS
END TEST 1

ARM THE TIMING OPTION FROM THE MAIN AND TRIGGER ON WORD RECOGNITION

PROGRAM DESCRIPTION

Arm the Timing Option on the occurrence of a main word recognition. Trigger the Timing Option when its word recognizer becomes true.

PROGRAM NOTES

This program will arm the timing option when an opcode is fetched from address F956. The timing option will trigger when all 0's are detected on the timing option probe for a minimum of 60 ns.

F956 is the address of the instruction which clears the counter connected to the timing option probe. The timing option will then trigger when the outputs from the counter become all 0's.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
ADDRESS = F956
FETCH = 1

TRIGGER
1 - Timing
1 - Centered
Sample Period = 200 NS
Word Recognizer = 00000000
Filter = 60 NS

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F956
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1 TRIGGER 1-TIMING
1     1-CENTERED
1     THRESHOLD V.  = 0-PLUS 1.40
1     1-ARM ASYNC, TRIG ON WR┘
1     SAMPLE PERIOD 2 * 1-100NS
1     WORD RECOGNIZER=00000000
1     EXT TRIG IN=X
1     GLITCH RECOGNIZER=XXXXXXXX
1     FILTER=060 NS
 END TEST 1
```

TRIGGER MAIN FROM TIMING

PROGRAM DESCRIPTION

Trigger the Main Acquisition Memory when the Timing Option word recognizer stays true for 300 ns.

PROGRAM NOTES

In this program the timing word recognizer bit in the main word recognizer was set to 1. This "AND's" the output of the timing option word recognizer with the other fields of the main word recognizer (all don't cares in this case). When the timing option word recognizer becomes true (the 0 value must be valid for 300 ns for this to happen), the output of the word recognizer is pulse stretched to the next 7D02 State clock. At this point the main word recognizer becomes true, triggering the Main Acquisition Memory.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1

Timing Word Recognizer = 1
Word Recognizer = 00000000
Filter = 300 NS

TRIGGER

END

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=XXXX
1 /NMI=X /IRQ=X FETCH=X R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=1
1     THRESHOLD V.  = 0-PLUS 1.40
1     1-ASYNC
1     SAMPLE PERIOD 2 * 1-100NS
1     WORD RECOGNIZER=00000000
1     EXT TRIG IN=X
1     GLITCH RECOGNIZER=XXXXXXXX
1     FILTER=300 NS
1THEN DO
1 TRIGGER 0-MAIN
1     0-BEFORE DATA
1     0-SYSTEM UNDER TEST CONT.
1     0-STANDARD CLOCK QUAL.
 END TEST 1
```

TRIGGER MAIN FROM TIMING AND MAIN

PROGRAM DESCRIPTION

Trigger the main section when both a main word recognition **and** an Asynchronous Timing Option word recognition occur.

PROGRAM NOTES

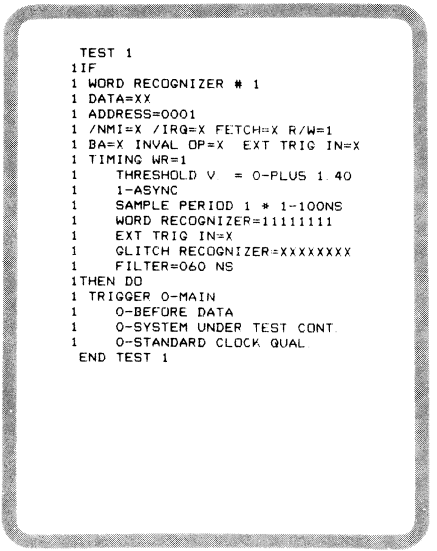
This program will trigger the Main Acquisition Memory when a data value of 11111111 is detected on the timing option probe for 60 ns just prior to a state clock in which address 0001 is detected by the main word recognizer.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
Address = 0001
R/W = 1
Timing Word Recognizer = 1
Word Recognizer = 11111111
Filter = 60 NS

TRIGGER

END


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=0001
1 /NMI=X /IRQ=X FETCH=X R/W=1
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=1
1 THRESHOLD V = 0-PLUS 1.40
1 1-ASYNC
1 SAMPLE PERIOD 1 * 1-100NS
1 WORD RECOGNIZER=11111111
1 EXT TRIG IN=X
1 GLITCH RECOGNIZER=XXXXXXXX
1 FILTER=060 NS
1THEN DO
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT
1 0-STANDARD CLOCK QUAL
END TEST 1

TRIGGER MAIN AND TIMING FROM A MAIN WORD RECOGNIZER

PROGRAM DESCRIPTION

A main word recognizer triggers both the main section and the Timing Option (asynchronous).

PROGRAM NOTES

This program will trigger the Main Acquisition Memory and the Timing Option Acquisition Memory when an IRQ interrupt is requested (the /IRQ lines go low).

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
/IRQ = 0

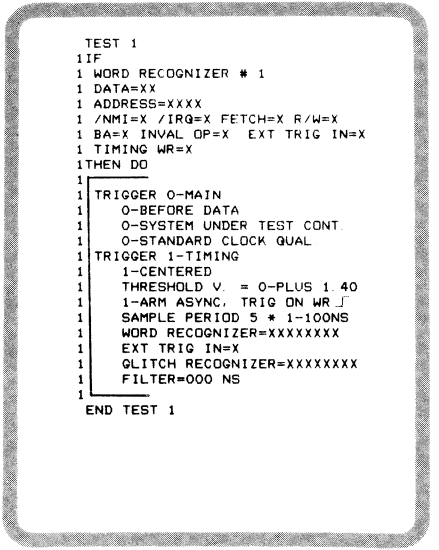
[ ]

TRIGGER

TRIGGER
1 - Timing
1 - Centered
Sample Period =
500 NS

[ ]

END


TEST 1
IF
WORD RECOGNIZER # 1
DATA=XX
ADDRESS=XXXX
/NMI=X /IRQ=X FETCH=X R/W=X
BA=X INVAL OP=X EXT TRIG IN=X
TIMING WR=X
THEN DO
TRIGGER 0-MAIN
0-BEFORE DATA
0-SYSTEM UNDER TEST CONT
0-STANDARD CLOCK QUAL
TRIGGER 1-TIMING
1-CENTERED
THRESHOLD V. = 0-PLUS 1.40
1-ARM ASYNC, TRIG ON WR
SAMPLE PERIOD 5 * 1-100NS
WORD RECOGNIZER=XXXXXXXX
EXT TRIG IN=X
GLITCH RECOGNIZER=XXXXXXXX
FILTER=000 NS
END TEST 1

MAIN WORD RECOGNIZER TRIGGER MAIN AND ARMS TIMING OPTION

PROGRAM DESCRIPTION

A main word recognition TRIGGERS the main section and ARMS the Timing Option (asynchronous).

PROGRAM NOTES

This program will trigger the Main Acquisition Memory and arm the Timing Option Acquisition Memory when an Opcode is fetched from address F909. The timing option will trigger when all channels on the timing option probe remain high for at least 300 ns.

KEYSTROKES *(IN SEQUENCE)*

WORD RECOGNIZER #1
Address = F909
Fetch = 1

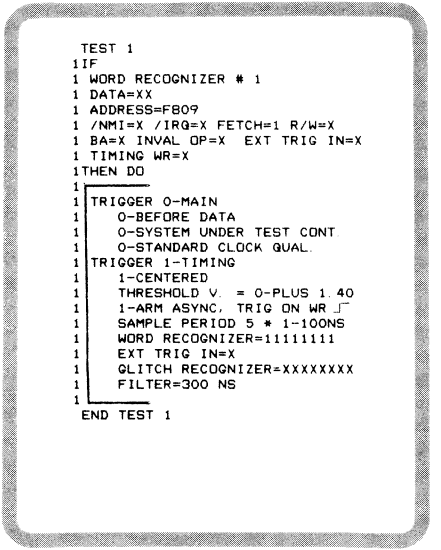
[ ]

TRIGGER

TRIGGER
1 - Timing
1 - Centered
Sample Period = 500 NS
Word Recognizer = 11111111
Filter = 300 NS

[ ]

END


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=F809
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT
1 0-STANDARD CLOCK QUAL
1 TRIGGER 1-TIMING
1 1-CENTERED
1 THRESHOLD V = 0-PLUS 1.40
1 1-ARM ASYNC, TRIG ON WR
1 SAMPLE PERIOD 5 * 1-100NS
1 WORD RECOGNIZER=11111111
1 EXT TRIG IN=X
1 GLITCH RECOGNIZER=XXXXXXXX
1 FILTER=300 NS
1
END TEST 1

SYNCHRONOUS OPERATION OF THE TIMING OPTION

PROGRAM DESCRIPTION

Using the Timing Option synchronously as an 8-bit extension of the main section.

PROGRAM NOTES

In this program the Main Acquisition Memory and the Timing Option Acquisition Memory will both be triggered when an NMI interrupt is requested (this occurs when the /NMI channel into the main word recognizer goes low) AND all channels on the timing option word recognizer go high. This is detected on the edge of a state clock only. Note that when used in synchronous mode, the timing option is not armed, but is triggered immediately. The timing option is also subject to any qualification entered for the Main Memory (only when used synchronously).

KEYSTROKES *(IN SEQUENCE)*

```
WORD RECOGNIZER
#1
    /NMI = 0
    Timing  Word
    Recognizer = 1
    0 - Sync
    Word Recognizer =
      11111111

[ ]

TRIGGER

TRIGGER
    1 - Timing
    1 - Centered

[ ]

END
```

```
 TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=XXXX
1 /NMI=0 /IRQ=X FETCH=X R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=1
1     THRESHOLD V.  = 0-PLUS 1.40
1     0-SYNC
1     WORD RECOGNIZER=11111111
1THEN DO
1┌──────
1│TRIGGER 0-MAIN
1│    0-BEFORE DATA
1│    0-SYSTEM UNDER TEST CONT.
1│    0-STANDARD CLOCK QUAL.
1│TRIGGER 1-TIMING
1│    1-CENTERED
1│    THRESHOLD V.  = 0-PLUS 1.40
1│    0-SYNC, TRIGGER IMMEDIATE
1└──────
 END TEST 1
```

SYNCHRONOUS OPERATION OF THE TIMING OPTION—NO QUALIFICATION

PROGRAM DESCRIPTION

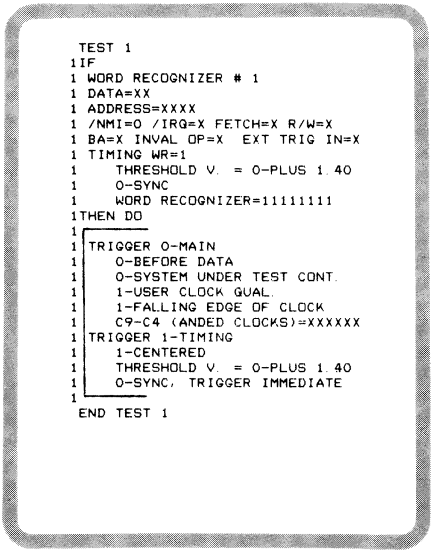
Trigger the Main Acquisition Memory and the Timing Option Memory synchronously with no clock qualification.

PROGRAM NOTES

This program is the same as in the previous example except that the clock qualification has been turned off. (It previously stored only on valid memory address cycles.)

KEYSTROKES *(IN SEQUENCE)*

```
WORD RECOGNIZER
#1
    /NMI = 0
    Timing  Word
    Recognizer =1
    0 - Sync
    Word Recognizer =
      11111111

[]

TRIGGER
    1 - User Clock Qual.
    C6 = X

TRIGGER
    1 - Timing
    1 - Centered

[]

END
```


TEST 1
1IF
1 WORD RECOGNIZER # 1
1 DATA=XX
1 ADDRESS=XXXX
1 /NMI=0 /IRQ=X FETCH=X R/W=X
1 BA=X INVAL OP=X EXT TRIG IN=X
1 TIMING WR=1
1 THRESHOLD V. = 0-PLUS 1.40
1 0-SYNC
1 WORD RECOGNIZER=11111111
1THEN DO
1
1 TRIGGER 0-MAIN
1 0-BEFORE DATA
1 0-SYSTEM UNDER TEST CONT.
1 1-USER CLOCK QUAL.
1 1-FALLING EDGE OF CLOCK
1 C9-C4 (ANDED CLOCKS)=XXXXXX
1 TRIGGER 1-TIMING
1 1-CENTERED
1 THRESHOLD V. = 0-PLUS 1.40
1 0-SYNC, TRIGGER IMMEDIATE
1
END TEST 1

BLOCK QUALIFICATION WITH THE TIMING OPTION

PROGRAM DESCRIPTION

Trigger the Main and Timing Option Memories (running synchronously) from the Main. Qualify using a block qualify.

PROGRAM NOTES

This program will trigger the Main Memory and the Timing Option Memory synchronously when an opcode is fetched from address F953. The data is qualified by a block qualifier such that only cycles in which the /IRQ line is low are stored in both memories.

KEYSTROKES *(IN SEQUENCE)*

QUALIFY

WORD RECOGNIZER #1
/IRQ — 0

END

WORD RECOGNIZER #2
ADDRESS — F953
FETCH — 1

[ ]

TRIGGER

TRIGGER
1 - Timing
0 — Sync

[ ]

END

```
 QUALIFY
GSTORE ON
G WORD RECOGNIZER # 1
G DATA=XX
G ADDRESS=XXXX
G /NMI=X /IRQ=0 FETCH=X R/W=X
G BA=X INVAL OP=X  EXT TRIG IN=X
G TIMING WR=X
 END QUALIFY
 TEST 1
1IF
1 WORD RECOGNIZER #2
1 DATA=XX
1 ADDRESS=F953
1 /NMI=X /IRQ=X FETCH=1 R/W=X
1 BA=X INVAL OP=X  EXT TRIG IN=X
1 TIMING WR=X
1THEN DO
1┌──────
1│TRIGGER 0-MAIN
1│    0-BEFORE DATA
1│    0-SYSTEM UNDER TEST CONT.
1│    0-STANDARD CLOCK QUAL.
1│TRIGGER 1-TIMING
1│    0-BEFORE DATA
1│    THRESHOLD V. = 0-PLUS 1.40
1│    0-SYNC, TRIGGER IMMEDIATE
1└──────
 END TEST 1
```

7D02 LOGIC ANALYZER