OPERATOR'S MANUAL

Model BE-64

Bus Emulator/Word Generator

Manual Revision 05/01/03
Manual Part Number: BEOM100
Instrument Part Number: BE-64

talon
INSTRUMENTS
150 East Arrow Hwy San Dimas, Ca. 91773 909-599-0690 Fax 909-599-6529 www.taloninst.com info@taloninst.com

CERTIFICATION

Talon Instruments certifies that this product met its published specifications at the time of shipment from the factory.

WARRANTY

Talon Instruments products are warranted against defects in materials and workmanship as follows:

(a) One year for the BE-64.

(b) Ninety days for pod cables.

During the warranty period, Talon Instruments will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to the Talon Instruments factory. Buyer shall prepay shipping charges to the factory and Talon Instruments shall pay shipping charges to return the product to the Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Talon Instruments from another country.

Talon Instruments warrants that its software and firmware designated by Talon for use with its instruments will execute its programming instructions when properly installed on the instrument. Talon Instruments does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. TALON INSTRUMENTS SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. TALON INSTRUMENTS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

SAFETY FIRST

PROTECT YOURSELF AND THE EQUIPMENT.
Follow these precautions:

- Don’t bypass the VXI chassis’ power cord’s ground lead with two-wire extension cords or plug adapters.
- Don't disconnect the green and yellow safety-earth-ground wire that connects the ground lug of the VXI chassis power receptacle to the chassis ground terminal.
- Don't energize the VXI chassis until directed to by the installation instructions.
- Don't repair the instrument unless you are a qualified electronics technician and have instructions from Talon Instruments.
- Pay attention to the **WARNING** statements. They point out situations that can cause injury or death.
- Pay attention to the **CAUTION** statements. They point out situations that can cause equipment damage.
- Use ESD static control procedures when handling the SR192A or any of its modules.

Table of Contents

List of Figures

List of Tables

1 GENERAL

1.1 MODEL BE-64

The Model BE-64 houses the digital resources required in digital test and trouble-shooting applications. The Model BE-64 may be software configured to handle applications ranging from basic word generation to sophisticated microprocessor or bus emulation. See Appendix C for information on bus emulation.

1.1.1 Standard Features

The Talon Model BE-64 is a single slot “C” size, VXI module. The Model BE-64 adheres to the SCPI (Standard Commands for Programmable Instruments) remote programming format version 1991.0.

1.1.2 Signal Overview

- 64 I/O channels x 32K bits/channel (Address and Data fields or Word Generator Data; maximum data rate = 25 MHz)
- 12 Output Programmable Timing and Control Signals; maximum output rate = 50 MHz
- 3 Input Programmable Timing and Control Signals; maximum input rate = 50 MHz
- 6 Input External Field Control Signals; maximum input rate = 25 MHz
- 24 Programmed Input/Output Signals; data rate defined by data transfer rate of VXI controller. 16 input/output, 8 output/counter.
- 30 miscellaneous timing and control signals required for exact simulation of various microprocessors and compatible signals for parallel to serial word generator converter pod.
- 2 Clock Outputs; frequency = 10 MHz, 20 MHz, or 50 MHz
- 1 Clock Input; maximum frequency = 50 MHz
- All signals TTL compatible.

1.1.3 Programming Modes

The Model BE-64 can be programmed to operate in either single cycle or continuous cycle mode. Continuous cycle mode allows for easier trouble-shooting of the Unit Under Test (UUT).

1.1.4 Handshaking

Since most digital buses require some form of input control, the Model BE-64 has the capability to handshake with the UUT. One edge sensitive input and two level sensitive inputs are provided for this function.

1.1.5 Internal/External Control

The Timing Sets of the Model BE-64 can be internally or externally clocked (DC to 50 MHz). The internal clock can be software configured to 10 MHz, 20 MHz, or 50 MHz.

The direction, enable and strobe lines of the 64 I/O channels can also be internally or externally controlled.

1.1.6 Microprocessor Bus Emulation

The Model BE-64 can be programmed to simulate the bus structured interface of any microprocessor incorporating an address and data bus, each bus not exceeding 32 bits in width. All timing and control signals are simulated, generating bus cycles to 50 MHz.

1.1.7 Bus Structured Interface Simulation

Bus structured interfaces are simulated exactly like the bus structured interface of a microprocessor. The architecture of the BE-64 has enabled Talon customers to simulate well over 1000 different digital interfaces. These interfaces range from a serial communication interface to a high speed VME bus structured interface. Talon offers “set up” configurations for several bus structured interfaces including the ones listed below. Contact the factory for availability of other configurations not listed..

VME bus AT bus VXI bus

1.1.8 Word Generator Operation

The word generator operation of the Model BE-64 incorporates the following features:

- 64 I/O channels x 32K bits/channel; 25 MHz data rates; Tristate Output control
- 32 I/O channels x 64K bits/channel; 50MHz data rates (external adapter required).
- The 32K memories are software allocated into tables. Up to 256 tables may be defined.
- Table length is programmed from 1 word to 32K words.
- Each table may be looped from 1 to 64K times, or run in continuous mode.
- Up to 16 table / timing entries can be sequenced together, generating zero clocks between tables.
- Continuous operation of an “idle” cycle which can run while downloading new data tables and/or sequences.
- Each table can be input or output.

1.1.9 Non-Bus Structured Interface Simulation

The Model BE-64 word generator mode allows the simulation of almost any user defined digital interface.

1.1.10 Stimulus/Response Testing

For test applications where test vectors are available, the Model BE-64 can be programmed to transmit or receive 64 channels by 32K bits per channel at frequencies to 25 MHz.

1.1.11 Logic Recorder

The BE-64 can be programmed to record up to 64 channels by 32K/channel. When used in conjunction with Talon’s serial word generator card, 2 megabits of serial data can be recorded.

1.2 SPECIFICATIONS

1.2.1 Field I/O Channels

General

Number of Channels: .....64 Bidirectional
Memory Depth: .....32K bits per channel

Timing

Data Rate .....0-25 MHz
Falltime .....3ns typical
Skew I/O Pins .....<3ns typical
Rise time .....3ns typical

Output Levels

High, open .....3.4V min.
Low, open .....0.4V max.
High, sourcing .....12ma @ 3.4V
Low, sinking .....24ma @ 0.4V

Input Levels

High .....>2.0 V
Low .....<0.8V

External Tri-state Control

Input Levels
- Disabled, High .....>2.0 V
- Enabled, Low .....<0.8V

1.2.2 Timing

Timebase

Subcycle/Cell Period
- Min. .....20 ns
- Max .....100ns x 215 or external
- Resolution .....20 ns min.

Cycle Duration
- Min. .....40ns or 2 Cells
- Max. .....256 x 215 Cells

Number timing cycles .....16 active
External Clock 0-50MHz
Internal Clock .....10, 20, & 50MHz

External Clock

Max. Speed .....50 MHz
Min. Pulse Width 8ns
Input Levels
- High .....>2.0 V
- Low .....<0.8V

Control Timing Generator

Control Outputs 8
Resolution 20ns
Min. Pulse Width 20ns
Typical Rise time 4ns
Typical Falltime 4ns
Typical Skew .....<2ns (same module)

Trigger Timing

Min. Pulse Width
- Level Trigger 20ns
- Edge Trigger 10ns

Min. between Trigs .....20 ns

Delay Timing

Prog. Cell Delay .....0-32K clocks

1.3 GENERAL

1.3.1 SCPI Programming

Conforms to SCPI Version 1991.0 and IEEE-488.2 standard mandated commands. Root level commands include:

TABLe	TIMing	EXECute
OUTPut	CALCulate	STATus
SYSTem		

1.3.2 Power on/*RST defaults

The following table lists the power on/*RST defaults of the BE-64.

The *TST? command sets the BE-64 to the power on defaults after completion of the self test.

SUBSYSTEM	DEFAULT
TABLe	No tables defined.
TIMing	Pre-defined timing sets set to two cells. All user defined timing sets deleted.
SETup:CLOCk	10
SETup:DELay	0
SETup:CTIMeout	0
FCONtrol:DIRection	OUTPut
FCONtrol:OREGister	OFF
FCONtrol:OCONtrol	INTernal
CELL	All cells set to #Hffff.
TEST	All test cells reset.

SUBSYSTEM	DEFAULT
EXECute	
MODE	RESet
FIELd	OUT,0
OUTPut	
FIELd:MODE PIO1/PIO2	INPut
FIELd:MODE OUT	OUTPut
TTLTrg<n>:STATe	OFF
TTLTrg<n>:SOURce	TTLTrg0 - TRIGGER, TTLTrg4 - PFLD1
TTLTrg<a | b>:STATe	OFF
TTLTrg<a | b>:SOURce	TTLTrg0
BENable:STATe	OFF
BENable:DATA	All codes set to zero.
BAENable	OFF
RESet	HIGH

The Event status and Service Request enable registers are set to zero on power-up only.

1.3.3 VXI Interface

Message based, 1K byte input buffer. Supports the following subsets/protocols:

- A24 A16 D16 Slave;
- VXIbus IEEE-488.2 Instrument Protocol (I4);
- VXIbus Event Generation Protocol (EG);
- VXIbus Interrupter.
- SC (Static Configuration) Device;
- DC (Dynamic Configuration) Device;

1.3.4 Environmental

Temperature Range

Operating:.....0°C to 50°C (25°C æ10°C for specified operation)
Storage: -40°C to +71°C (RH not controlled.)

Altitude:

Operating:.....Sea level to 10,000 ft.
Storage: Sea level to 40,000 ft.

Relative Humidity (non-condensing)

0°C to +10°C:.....not controlled.
+11°C to +30°C:.....95 æ5% RH max.
+31°C to +40°C:.....75 æ5% RH max.
+41°C to +50°C:.....45 æ5% RH max.

1.3.5 Size

Dimensions: .....Single slot, “C” size VXI module. (30 x 260 x 350 mm).
Weight: .....<1.90 kg (4.19 lb.).

1.3.6 Power

Total:<60 Watts

Voltage	Peak Current	Dynamic Current
+5 Vdc	11.5A	1.5A

1.3.7 Cooling Requirements

For 10 C degree rise 5.0 l/s, 0.5 mm H20

1.4 Technical Specifications

This section describes the timing specifications for the Model BE-64. Unless otherwise noted, these timing specifications are valid in all cells of the timing set.

These specifications were obtained from a test fixture at the end of one foot ~100 ohm ribbon cables. If cables of a different length are used, then that difference must be taken into account. The rule of thumb for signal delay in cable is 1.7ns per foot.

1.4.1 BE-64 Timing Set Specifications

The timing set specifications are in reference to the timing set cells. The timing set cells are defined by the TSOUT(1-8) signals measured on the UUT. See section 1.4.1.1.

1.4.1.1 Clock Reference Timing

The source of the clock signal for the Model BE-64 consists of either an internal clock (TSCLK-A) or a user supplied external clock (EXCLK-, 50 MHz max.). Figure 1-1 illustrates the timing relationship between the two clocks when EXCLK- is used.

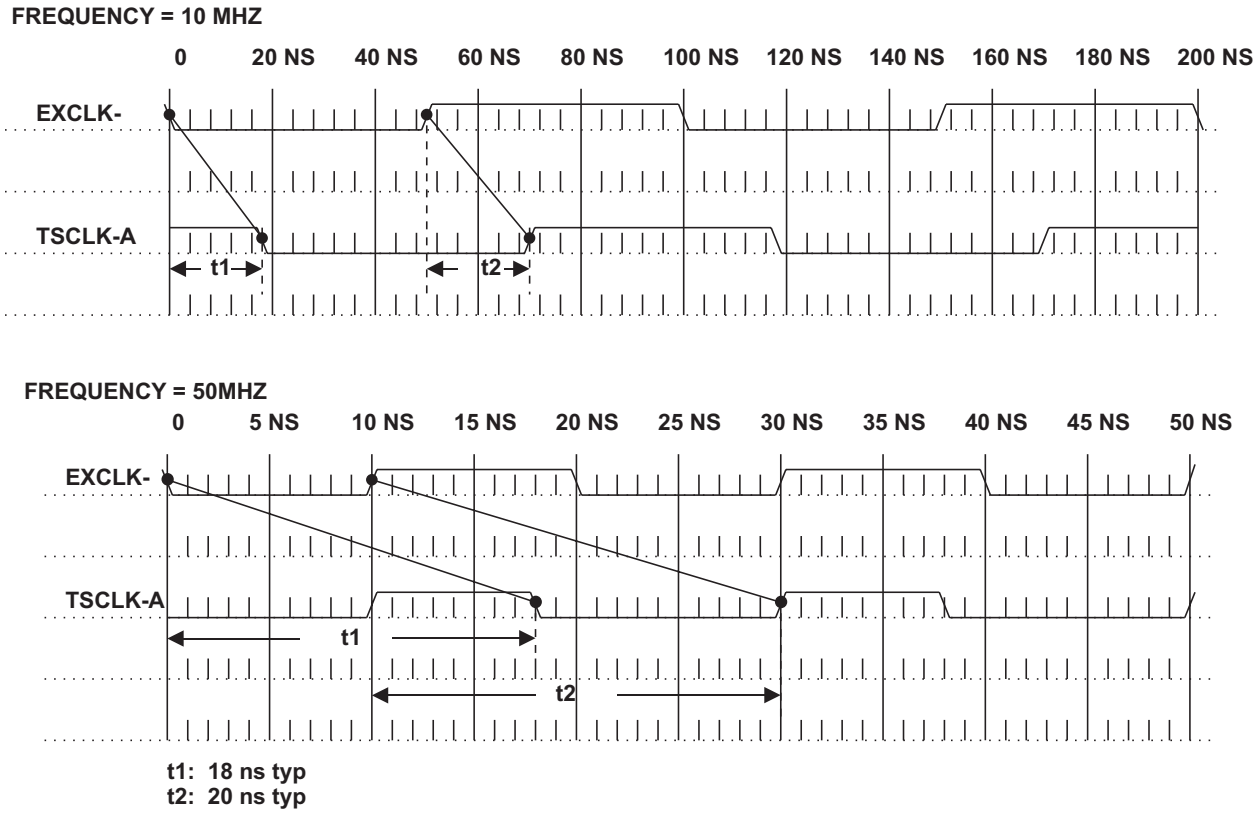


FIGURE 1-1 TSCLK-A IN REFERENCE TO EXCLK-

The timing set cells are defined by the TSOUT(1-8) signals measured on the UUT. Figure 1-3 illustrates

the TSOUT(1-8) signals with respect to TSCLK-A. Also, the relationship between the TSOUT signals is illustrated.

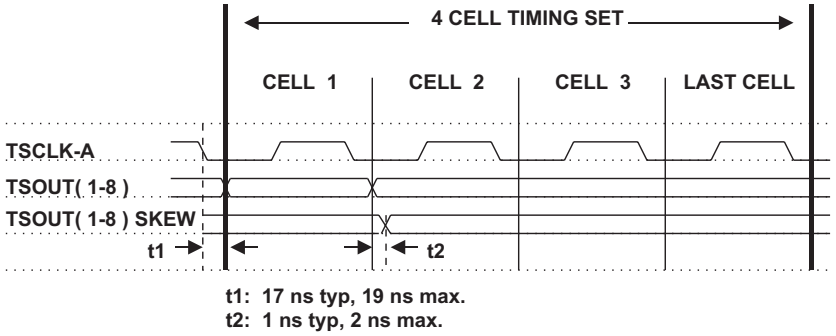


FIGURE 1-3 TSOUT(1-8) SIGNALS WITH RESPECT TO TSCLK-A

1.4.1.2 Input Timing Set Signals

Figure 1-2 illustrates the timing specifications for the test input (handshake) signals. When using TSINPUT1 or TSINPUT2 the BE-64 will not enter a WAIT state if the respective signal is asserted t1 or greater prior to the beginning of the test cell and is not removed t2 or less prior to the beginning of the test cell.

When using TSSTROBE the BE-64 will not enter a WAIT state if the transition occurs t3 or greater prior to the beginning of the test cell. TSSTROBE test is invalid during the last cell.

For PCB REV. N/C the last cell cannot be used as a TSINPUT test cell. For PCB REV. A the last cell can be used as a TSINPUT test cell.

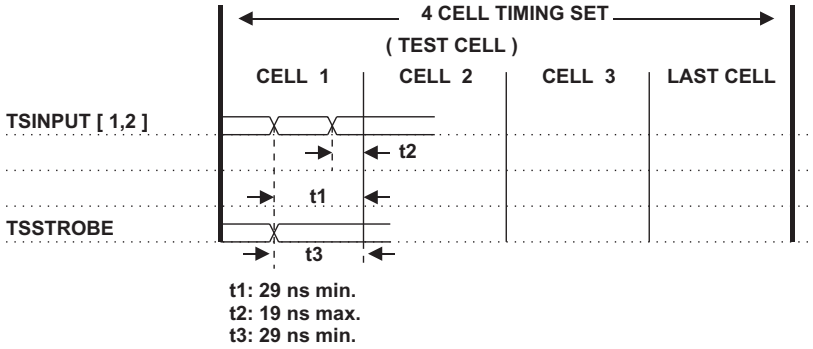


FIGURE 1-2 INPUT TIMING SET SIGNALS

1.4.2 BE-64 Field Channel Specifications

The field channel specifications are in reference to the timing set cells. See section 1.4.1.1. Both fields are identical and independent. Only the address in the 32K memory is common between them.

Each field can have the direction set to input, output, or be controlled externally. When the direction is set to input, the input strobe can be controlled either internally or externally. When the direction is set to output, the output can be either registered or non-registered and the enable and strobe can be controlled either internally or externally.

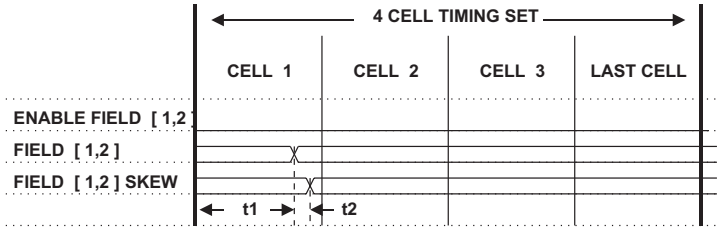
1.4.2.1 Non-registered Output Timing

In this section the direction of the field is set to output and is non-registered.

Figure 1-5 illustrates the time that data becomes valid when the output is continuously enabled. The

enable control may be either internal or external. Also, the relationship between the data channels is illustrated.

SELECTION / DIRECTION	OUTPUT REGISTER		OUTPUT CONTROL	INPUT STROBE
INPUT	N/A		N/A	INTERNAL / EXTERNAL
OUTPUT	YES	NO	INTERNAL / EXTERNAL	N/A
EXTERNAL	YES / NO		INTERNAL / EXTERNAL	INTERNAL / EXTERNAL

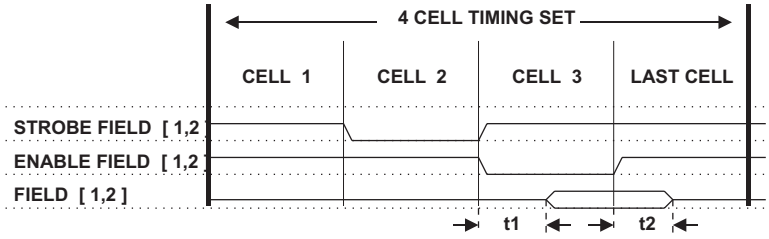


t1: 15 ns typ, 18 ns max.
t2: 1.5 ns typ, 3 ns max.

FIGURE 1-5 DATA VALID, CONTINUOUSLY ENABLED

Figure 1-4 illustrates the time that data is valid when the enable is controlled internally. Due to signal delays, this timing specification does not apply to cell 1. Also, the time for F[1,2]ENBLD- to be asserted is illustrated.

SELECTION / DIRECTION	OUTPUT REGISTER		OUTPUT CONTROL	INPUT STROBE
INPUT	N/A		N/A	INTERNAL / EXTERNAL
OUTPUT	YES	NO	INTERNAL EXTERNAL	N/A
EXTERNAL	YES / NO		INTERNAL / EXTERNAL	INTERNAL / EXTERNAL



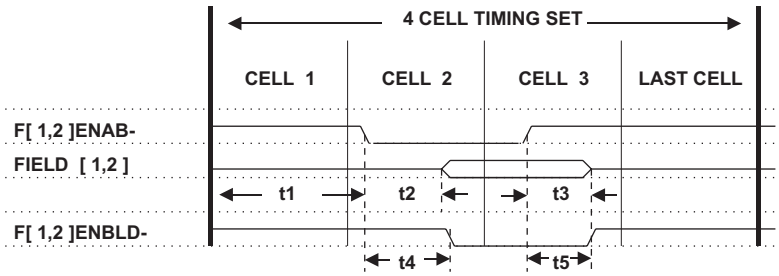
t1: 11 ns typ, 13 ns max. (not applicable for cell 1 and cell 2)
t2: 15 ns typ, 14 ns min.

FIGURE 1-4 DATA VALID/INVALID, INTERNALLY ENABLED

Figure 1-7 illustrates the time that data is valid when the enable is controlled externally. Due to signal delays, the timing specification for when data becomes valid (t2) only applies if the restriction on the enable signal with respect to the beginning of cell 1 (t1) is met. Should the external enable occur earlier,

then t2 will be reduced. The limit is t1 in figure 1-5. Also, the time for F[1,2]ENBLD- to be asserted is illustrated.

SELECTION / DIRECTION	OUTPUT REGISTER		OUTPUT CONTROL	INPUT STROBE
INPUT	N/A		N/A	INTERNAL / EXTERNAL
OUTPUT	YES	NO	INTERNAL EXTERNAL	N/A
EXTERNAL	YES / NO		INTERNAL / EXTERNAL	INTERNAL / EXTERNAL



t1: 2 ns min.
t2: 23 ns typ, 25 ns max.
t3: 20 ns typ, 19 ns min.
t4: 28 ns typ, 30 max.
t5: 28 ns typ, 30 max.

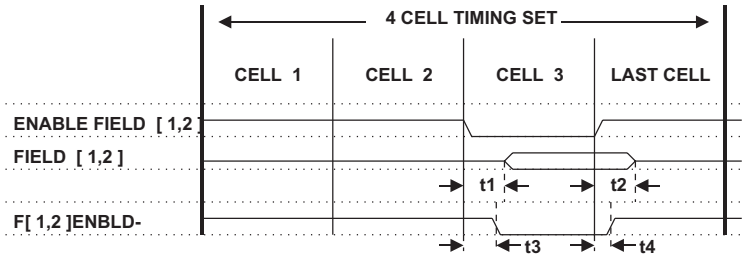
FIGURE 1-7 DATA VALID/INVALID, EXTERNALLY ENABLED

1.4.2.2 Registered Output Timing

In this section the direction of the field is set to output and is registered.

Figure 1-6 illustrates the time that data is valid when the enable is controlled internally and the internal strobe has already occurred. Due to signal delays, the internal strobe must not occur in cell 1.

SELECTION / DIRECTION	OUTPUT REGISTER		OUTPUT CONTROL	INPUT STROBE
INPUT	N/A		N/A	INTERNAL / EXTERNAL
OUTPUT	YES	NO	INTERNAL EXTERNAL	N/A
EXTERNAL	YES / NO		INTERNAL / EXTERNAL	INTERNAL / EXTERNAL



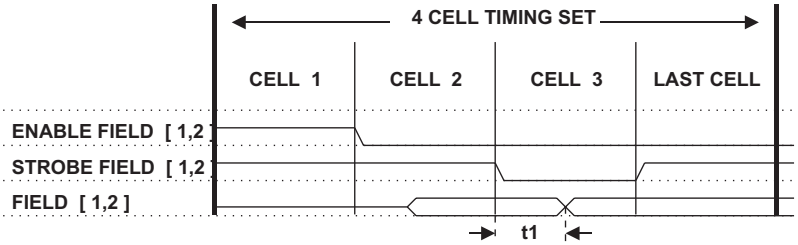
t1: 11 ns typ, 13 ns max. (not applicable for cell 1)
t2: 15 ns typ, 14 ns min.
t3: 9 ns typ, 11 ns max.
t4: 9 ns typ, 11 ns max.

FIGURE 1-6 DATA VALID/INVALID, INTERNALLY ENABLED

Figure 1-9 illustrates the time that data becomes valid after the internally controlled strobe occurs and the

internal enable has already occurred. Due to signal delays, this timing specification does not apply to cell 1.

SELECTION / DIRECTION	OUTPUT REGISTER		OUTPUT CONTROL	INPUT STROBE
INPUT	N/A		N/A	INTERNAL / EXTERNAL
OUTPUT	YES	NO	INTERNAL	N/A
			EXTERNAL	
EXTERNAL	YES / NO		INTERNAL / EXTERNAL	INTERNAL / EXTERNAL

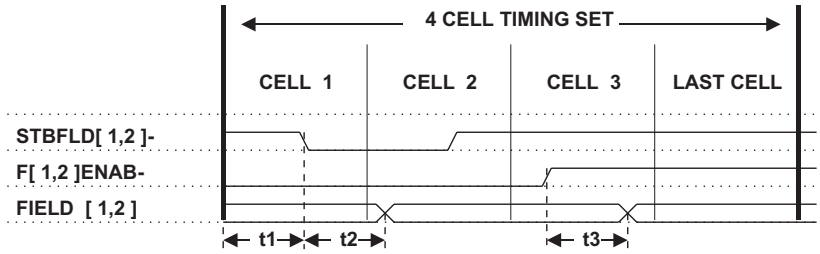


t1: 8 ns typ, 10 ns max. (not applicable for cell 1)

FIGURE 1-9 DATA VALID, INTERNALLY STROBED

Figure 1-8 illustrates the time that data becomes valid when the strobe is controlled externally and the external enable has already occurred. Due to signal delays, the timing specification for when data becomes valid (t2) only applies if the restriction on the strobe signal with respect to the beginning of cell 1 (t1) is met. The external strobe must not occur prior to t1. Figure 1-8 also illustrates the time that data becomes invalid when the field is externally disabled.

SELECTION / DIRECTION	OUTPUT REGISTER		OUTPUT CONTROL	INPUT STROBE
INPUT	N/A		N/A	INTERNAL / EXTERNAL
OUTPUT	YES	NO	INTERNAL	N/A
			EXTERNAL	
EXTERNAL	YES / NO		INTERNAL / EXTERNAL	INTERNAL / EXTERNAL



t1: 11 ns min.
t2: 18 ns typ, 20 ns max.
t3: 20 ns typ, 19 ns min.

FIGURE 1-8 DATA VALID, EXTERNALLY STROBED

1.4.2.3 Input Timing

In this section the direction of the field is set to input. When set to input, the field is always registered and enabled.

Figure 1-11 illustrates the data set-up and hold times for the internally controlled strobe.

SELECTION / DIRECTION	OUTPUT REGISTER	OUTPUT CONTROL	INPUT STROBE
INPUT	N/A	N/A	INTERNAL
			EXTERNAL
OUTPUT	YES / NO	INTERNAL / EXTERNAL	N/A
EXTERNAL	YES / NO	INTERNAL / EXTERNAL	INTERNAL / EXTERNAL

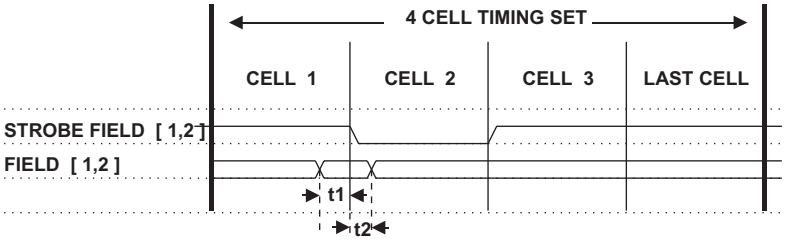


FIGURE 1-11 DATA SET-UP AND HOLD, INTERNALLY STROBED

Figure 1-10 illustrates the data set-up and hold times for the externally controlled strobe. There is also a restriction as to how close the external strobe can occur prior to the end of the last cell.

SELECTION / DIRECTION	OUTPUT REGISTER	OUTPUT CONTROL	INPUT STROBE
INPUT	N/A	N/A	INTERNAL / EXTERNAL
OUTPUT	YES / NO	INTERNAL / EXTERNAL	N/A
EXTERNAL	YES / NO	INTERNAL / EXTERNAL	INTERNAL / EXTERNAL

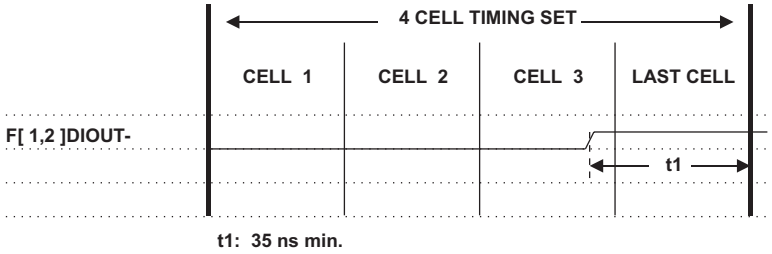


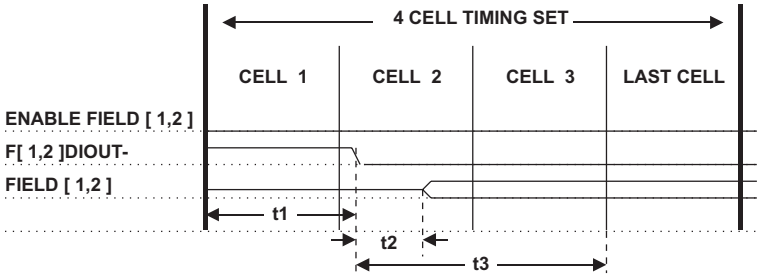
FIGURE 1-10 EXTERNAL DIRECTION TO INPUT

1.4.2.4 Bidirectional Timing

In this section the direction of the field is externally controlled. To ensure that the field functions correctly as an input or output, it is recommended that the external direction signal remain true during the entire timing set, especially in the last cell.

Should the external direction signal change state during the timing set, then figure 1-13 illustrates the time that data becomes valid when the external direction is set to output and the field is continuously enabled.

SELECTION / DIRECTION	OUTPUT REGISTER	OUTPUT CONTROL	INPUT STROBE
INPUT	N/A	N/A	INTERNAL / EXTERNAL
OUTPUT	YES / NO	INTERNAL / EXTERNAL	N/A
EXTERNAL	YES / NO	INTERNAL / EXTERNAL	INTERNAL / EXTERNAL



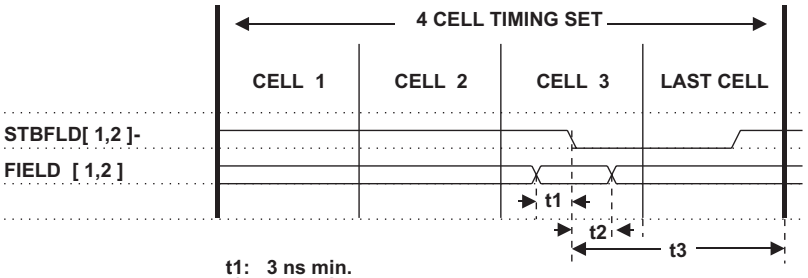
t1: 0 ns min.
t2: 24 ns typ., 26 ns max.
t3: 18 ns min.

FIGURE 1-13 EXTERNAL DIRECTION SET TO OUTPUT

To ensure that the correct word of data becomes available at the output, there is a restriction as to how close the external direction signal can change prior to the beginning of the last cell.

Figure 1-12 illustrates the restriction on the external direction signal in order to set the field to input

SELECTION / DIRECTION	OUTPUT REGISTER	OUTPUT CONTROL	INPUT STROBE
INPUT	N/A	N/A	INTERNAL EXTERNAL
OUTPUT	YES / NO	INTERNAL / EXTERNAL	N/A
EXTERNAL	YES / NO	INTERNAL / EXTERNAL	INTERNAL / EXTERNAL



t1: 3 ns min.
t2: 7 ns min.
t3: 30 ns min.

FIGURE 1-12 DATA SET-UP AND HOLD, EXTERNALLY STROBED

To ensure that the data is stored in the correct memory location, the external direction signal must be set to input at least t1 prior to the end of the last cell. See section 1.4.2.3 for relevant data set-up and hold times.

1.5 BE-64 Drive Specifications

Appendix G includes the type of device used as a driver as well as any termination for all BE-64 signals.

2 PREPARATION

2.1 RECEIVING INSPECTION

Check the shipment at the time of delivery and inspect each box for damage. Describe any box damage and list any shortages on the delivery invoice.

2.1.1 Unpacking Instructions

1. **Unpack the boxes.** Unpack the boxes in a clean and dry environment. Save all the packing material in case the instrument must be returned for repair.
2. **Inspect the shipment for damage.** Inspect the equipment carefully for any signs of mechanical damage regardless of the condition of the shipping boxes.
3. **If necessary, file a claim.** In the case of mechanical damage, call the shipper immediately and start the claim process.
4. **Call Talon.** Call Talon's Customer Service representative (909-599-0690) to inform them that the shipment arrived damaged. Please be prepared to provide a detailed damage report.

2.1.2 Returning Equipment

Follow these steps when you return equipment to Talon:

1. **Save the packing material.** Always return equipment in its original packing material and boxes. If you use inadequate material, you'll be responsible for any shipping damage repair as carriers won't accept responsibility on incorrectly packed equipment.
2. **Call Talon's Customer Service and ask for a return authorization.** The Talon Customer Service representative (909-599-0690) will ask for your name, telephone number, company name, equipment type, model number, serial number, and a description of your problem.
3. **Pack and ship the equipment to:**

   Talon Instruments
   150 E. Arrow Highway
   San Dimas, CA 91773

2.2 PREPARATION FOR STORAGE OR SHIPMENT

2.2.1 Packaging

If at all possible, always use the original shipping container. Use a double-walled cardboard shipping container. Protect all sides, including the top and bottom, with shock absorbing material (minimum of 1 inch thick material) to prevent movement of the Model BE-64 within the container. Seal the shipping container with approved sealing tape. Mark "FRAGILE" on all sides, top, and bottom of the shipping container.

2.2.2 Storage

The Model BE-64 should be stored in a clean, dry environment. In high humidity environments, protect the Model BE-64 from temperature variations that could cause internal condensation. The following environmental conditions apply to both shipping and storage:

Temperature	-40^0C to $+71^0$C
Relative Humidity	not controlled, non-condensing
Altitude	<40000 ft. (12192 m)
Vibration	<2g
Shock	<40g

2.3 PREPARATION FOR USE

Paragraph 2.3 covers the following topics:

- Logical Address Selection
- Rear Panel Connectors
- Jumper Configuration
- Cables
- Installation

2.3.1 Logical Address Selection

The VXI chassis Resource Manager identifies units in the system by the unit's logical address. The VXI logical address can range from 0 to 255. The exceptions are addresses 0 and 255. Address 0 is reserved for the Resource Manager. Address 255 is an invalid address for the Model BE-64.

The logical address of the Model BE-64 can be statically or dynamically configured. A jumper strap, as seen from the rear panel (FIGURE 2-1), is installed across jumper points E42 and E43 (E50 and E51 rev NC PCB) for static configuration. The jumper strap is removed for dynamic configuration. Talon ships the Model BE-64 in the dynamic configuration. The jumper strap, however, is installed on only one jumper point, as seen in FIGURE 2-1 (not connecting E42 and E43).

With the jumper strap installed for static configuration, the Model BE-64's logical address can be changed by setting the eight position DIP switch (FIGURE 2-1). The Model BE-64 uses binary values (2^0 to 2^7) to set the address. The up position (as viewed in FIGURE 2-1) represents a logical 1. The down position represents a logical 0. Switch position number one is the least significant bit of the address.

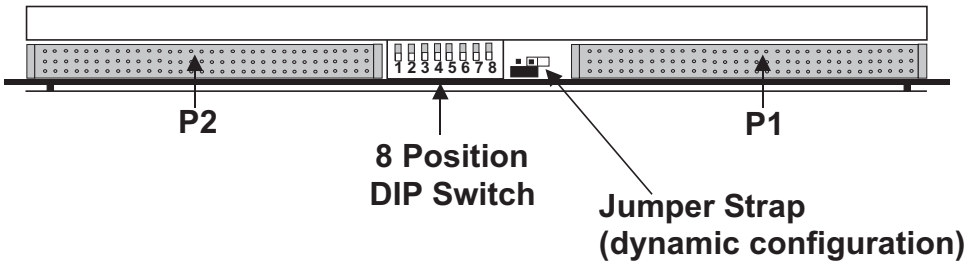


FIGURE 2-1 REAR PANEL

2.3.2 Rear Panel Connectors

Two multi-pin rear panel connectors, P1 and P2, connect the Model BE-64 to the VXI chassis. These connectors provide power supplies, digital interfacing, and the backplane signal lines used by the Model BE-64.

2.3.3 Jumper Configurations

See TABLE 2-1 for all jumper settings. See FIGURE 2-2 for the location of all the jumper points. All jumper points are located on the CPU card (bottom board). The Bus Emulator card (top board) must be carefully removed if the jumper configuration, other than the interrupt settings, is to be checked or changed. The interrupt settings are right angle jumpers accessible from the side.

SIGNAL NAME	"E" LOCATION	IF SHIPPED INSTALLED	FUNCTION
5V+	E50 E51 (E53 E54)	NO	Connects +5V from VXI backplane to front connectors J3 and J4.
5V+	E26 E27 (E28 E29)	NO	Connects +5V from VXI backplane to front connector J1.
SELV+ (+24V)	E61 E62 (E66 E67)	NO	Connects +24V from VXI backplane to front connector J3.
SELV+ (+12V)	E60 E61 (E65 E66)	NO	Connects +12V from VXI backplane to front connector J3.
SELV- (-24v))	E64 E65 (E69 E70)	NO	Connects -24V from VXI backplane to front connector J3.
SELV- (-12V)	E63 E64 (E68 E69)	NO	Connects -12V from VXI backplane to front connector J3.

SIGNAL NAME	"E" LOCATION	IF SHIPPED INSTALLED	FUNCTION
SC-	E43 E42 (E50 E51)	NO OR OFFSET	Static configuration if jumper strapped, then logical DIP switch is used. Dynamic configuration if not strapped.
LEV1	E57 E54 (E59 E62)	NO	VXI interrupt request response level. Binary encoded.
LEV2	E58 E55 (E60 E63)	YES	
LEV3	E59 E56 (E61 E64)	YES	
VIRQ1-	E12 E19 (E14 E21)	YES	VXI interrupt request to VXI interrupt request level output selection. One selection made by the user. Must also match binary encoded level (LEV_) jumper selections.
VIRQ2-	E13 E20 (E15 E22)	NO	
VIRQ3-	E14 E21 (E16 E23)	NO	
VIRQ4-	E15 E22 (E17 E24)	NO	
VIRQ5-	E16 E23 (E18 E25)	NO	
VIRQ6-	E17 E24 (E19 E26)	NO	
VIRQ7-	E18 E25 (E20 E27)	NO	

2.3.4 Cables

Since the user will require customized cables for the front panel connectors (J1-J4), Appendix E (J1-J4

TABLE 2-1 BOTTOM BOARD REV A AND (REV NC) JUMPER SETTINGS

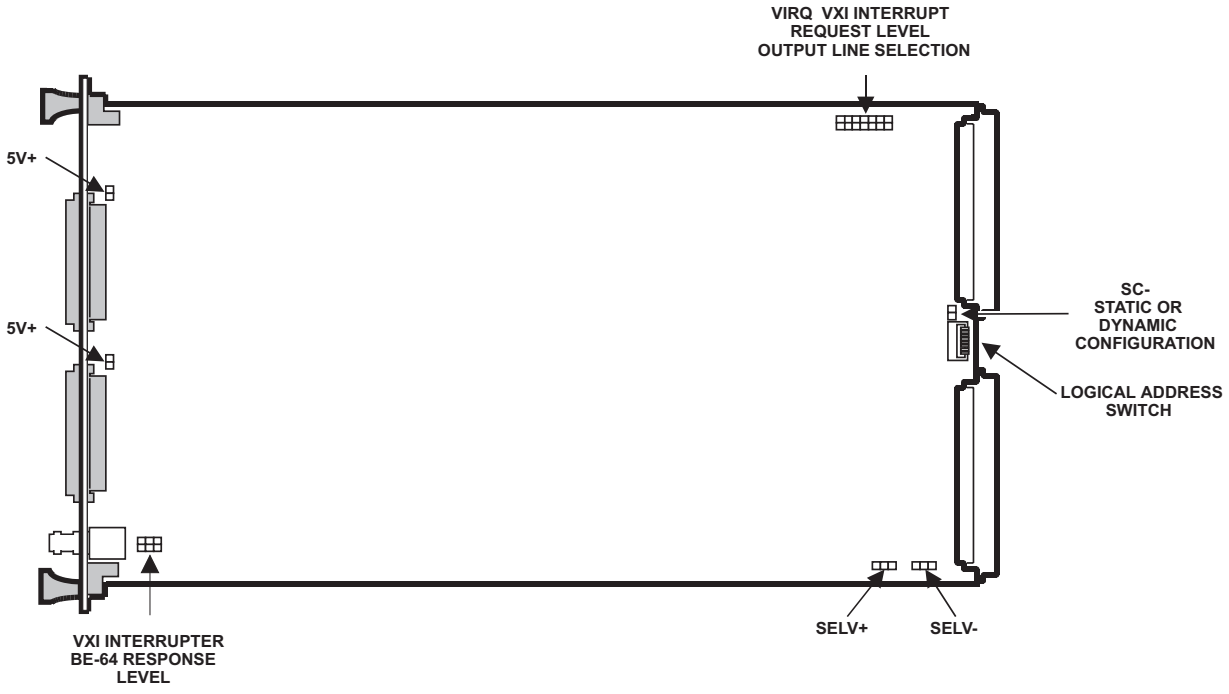


FIGURE 2-2 LOCATION OF JUMPER POINTS (BOTTOM BOARD)

Pinouts) has been provided. Appendix F includes a worksheet for each connector to be copied and filled in with the pinout of the user's custom built cables. Mating connectors for J1-J4 can be obtained from Robinson Nugent, Inc. (part number: P50E-080S-TG).

Appendix E contains examples and pinouts of typical cables using standard 40 pin connectors to mate with the UUT. These cables can be ordered from Talon (part number: BE64/300).

The cable assembly includes the 80 pin mating connector, three feet of 28 AWG ribbon cable and four forty pin headers (two spare and two crimped).

2.4 INSTALLATION

The Model BE-64 must be installed in a VXI mainframe in any slot except slot 0 (zero), which is reserved for the Resource Manager. Always check P1 and P2 for bent pins prior to installation. When inserting the BE-64 into the mainframe, it should be gently rocked back and forth to seat the connectors into the backplane receptacles.

2.4.1 Initial Power-On

With the Model BE-64 properly installed in a VXI chassis, turn the chassis power on. The Model BE-64 will immediately conduct a self-test which tests memory, registers, communication with the host, etc. (Note: a self-test adapter must be connected to the front connectors to test the full integrity of the drivers and receivers). The red FAIL indicator will come on and the SYSFAIL- line on the VXI backplane will be driven true. The other indicators may or may not flash during the self-test. After about three seconds, if the Model BE-64 passes the self-test, then the FAIL indicator will go off, all the other indicators will be off, and the SYSFAIL- line will no longer be driven by the Model BE-64. The Model BE-64 is now ready for use.

If the Model BE-64 fails the self test, then the FAIL indicator will stay on or flash and the SYSFAIL- line will continue to be driven. Should this happen, turn the chassis power off, make certain the Model BE-64 is properly installed, and turn the chassis power back on. Should the Model BE-64 continue to fail, then call Talon's Customer Service representative (909-599-0690) for assistance.

An optional self test fixture (talon part number BE64/201) provides an extended test of the BE-64's front panel I/O signals.

3 OVERVIEW

3.1 INTRODUCTION

The BE-64 can be thought of as a word generator, a timing generator, and a logic recorder all combined into a single “C”+ size VXI bus emulator card. Figure 3-1 is the block diagram of the BE-64. It must be understood that, in the BE-64, bus emulation is controlled by the timing sets. Each timing set is an executable cycle which simulates a bus cycle. The timing sets, in fact, define the characteristics of the bus to the BE-64. Appendix C will help in understanding bus emulation.

3.2 SYSTEM RESOURCES

The BE-64 is a 68010 microprocessor controlled VXI message based servant. The 68010 microprocessor system includes 128K of ROM and 64K of RAM. An additional 128K of ROM and 64K of RAM are available for MACRO programming of the BE-64. The operating system is multi-tasking which, in addition to performing system I/O and resource management, performs the following functions:

- Performs a system power-up self-test.
- Receives, parses and executes the SCPI commands.
- Processes the command data and converts them into an appropriate digital format compatible with the bus emulator logic.
- Executes built in functions such as RAM and ROM tests.
- Calculates CRC’S on field memories such that a “Learn UUT” function can be executed.
- Performs a table compare function on the field memories.

The MACRO programming allows the user to customize the BE-64 with unique timing sets and commands that can be executed by standard SCPI commands.

In order to speed up the access of the BE-64’s field and timing data, these memories have been assigned VXI A24 addresses and can be accessed directly via the VXI controller.

The BE-64 is also capable of generating VXI bus interrupts for any of the events in the status register. The BE-64 is shipped from the factory with VXI IRQ1 enabled. Any of the seven VXI IRQ lines can be selected via hardware jumper.

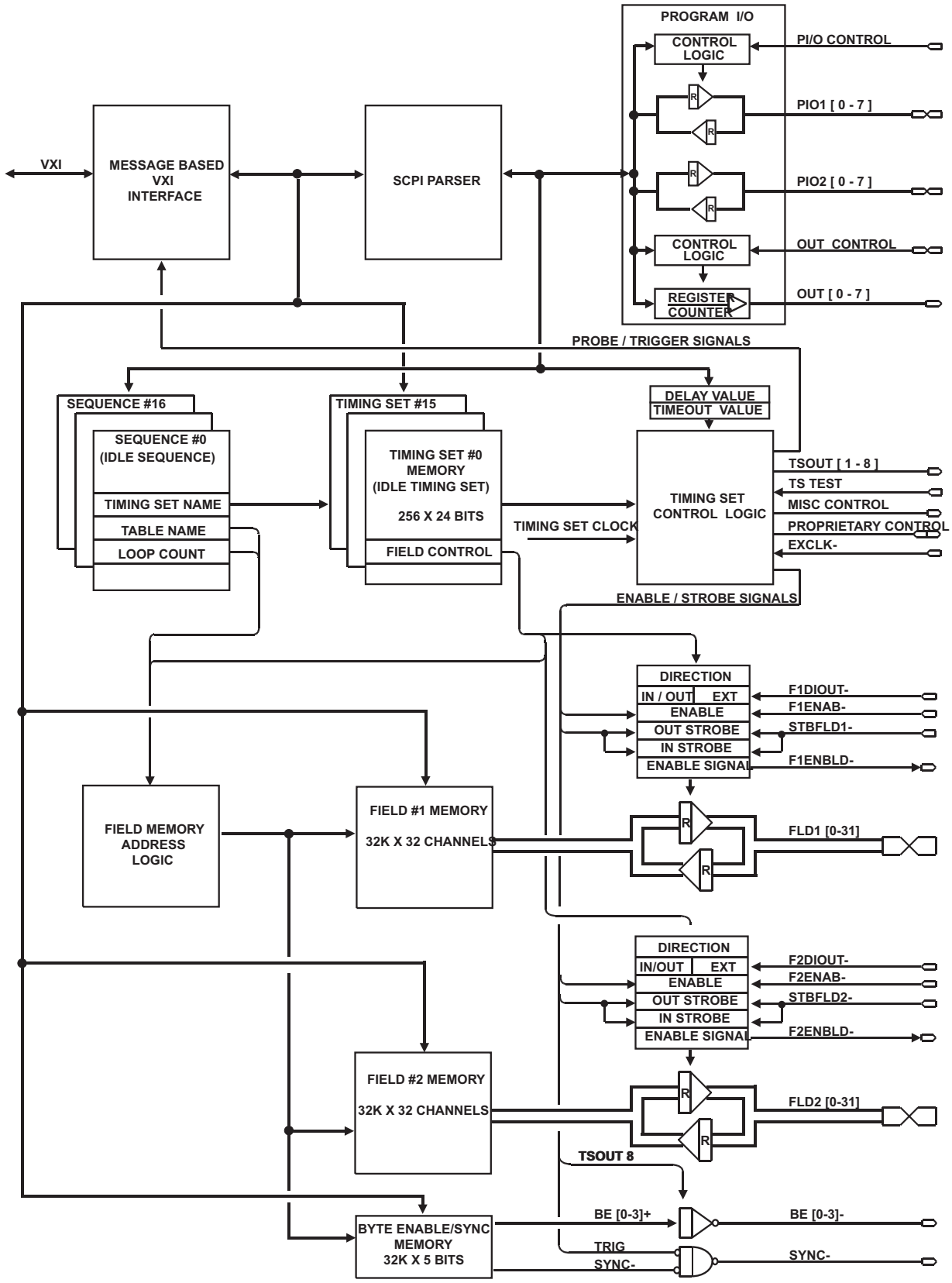


FIGURE 3-1 BE-64 COMPLETE BLOCK DIAGRAM

3.3 CONNECTORS AND INDICATORS

The Model BE-64 has four LED indicators, four 80 pin connectors, and one BNC connector. The front panel connectors and indicators are shown in figure 3-2 with the following descriptions:

1. FAIL/ID. The FAIL/ID indicator is a dual red/green LED. The red LED illuminates during the power-up self-test and stays on or flashes if a system failure is detected. Otherwise, it is turned off. The green LED illuminates whenever the MODULE ID line for this board, on the backplane, is active. An amber color is displayed if both LEDs are illuminated at the same time.
2. BUSY. The BUSY indicator is a red LED which illuminates when the Model BE-64 is executing a non-idle timing sequence. Otherwise, it is off.
3. WAIT. The WAIT indicator is a red LED which illuminates when the Model BE-64 is in a “WAIT” state. The Model BE-64 enters a “WAIT” state during a “test cell” (TSINPUT1, TSINPUT2, TSSTROBE), a “delay cell”, or when the bus acknowledge line (BUSACK-) is asserted by the Model BE-64 in response to a bus request (BUSREQ-) from the UUT. Otherwise, it is off.
4. TIMEOUT. The TIMEOUT indicator is a red LED which illuminates if the programmable timeout function is enabled and a timeout condition has occurred. Otherwise, it is off.
5. J1. The J1 connector provides the signals necessary to interface with the UUT. See Appendix G for a complete description of these signals. See Appendix E for a complete pinout of the J1 connector.
6. J2. The J2 connector provides the signals necessary to interface with the UUT. See Appendix G for a complete description of these signals. See Appendix E for a complete pinout of the J2 connector.
7. J3. The J3 connector provides the signals necessary to interface with the UUT. See Appendix G for a complete description of these signals. See Appendix E for a complete pinout of the J3 connector.
8. J4. The J4 connector provides the signals necessary to interface with the UUT. See Appendix G for a complete description of these signals. See Appendix E for a complete pinout of the J4 connector.
9. SYNC. The SYNC BNC provides an output signal which is the user programmed TRIGGER signal, of the Timing Set, gated with the internal memory table sync signal. SYNC is commonly used to trigger a scope.

FIGURE 3-2 FRONT PANEL CONNECTORS AND INDICATORS

3.4 BE-64 FUNCTIONAL OVERVIEW

The BE-64, as illustrated in figure 3-3, can be separated into seven functional blocks listed below. Each block will be described in the following sections.

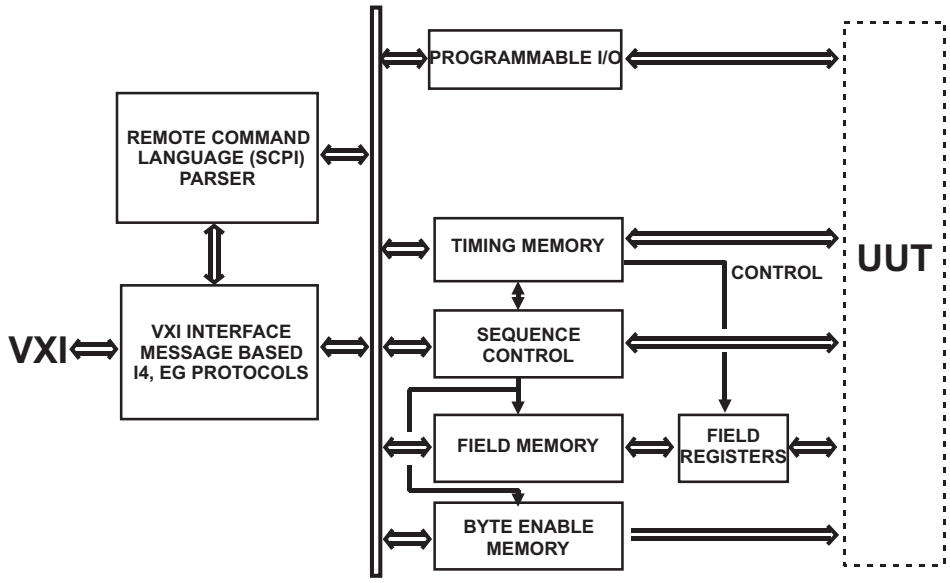


FIGURE 3-3 SIMPLIFIED BLOCK DIAGRAM

- VXI Interface.
- Remote Command Language Parser.
- Field Memory and Registers.
- Timing Memory.
- Sequence Control.
- Programmable I/O.
- Byte Enable Memory.

3.4.1 VXI Interface

The BE-64 is a message based servant that supports the IEEE 488.2 Instrument (I4) and Event Generator (EG) protocol.

Messages and commands are sent to the BE-64 via word serial protocol. In addition, the timing and field memories are assigned A24 VXI addresses so that the VXI controller can access the memory faster using direct memory access.

3.4.2 Remote Command Language Parser

The BE-64 conforms to the SCPI (Standard Commands for Programmable Instruments) version 1991.0. See Appendix A for an introduction to SCPI. See Section 5 for a description of the SCPI commands used by the BE-64.

3.4.3 Field Memory and Registers

The 64 X 32K field memory of the BE-64 is connected to the UUT via field registers as depicted in figure 3-4.

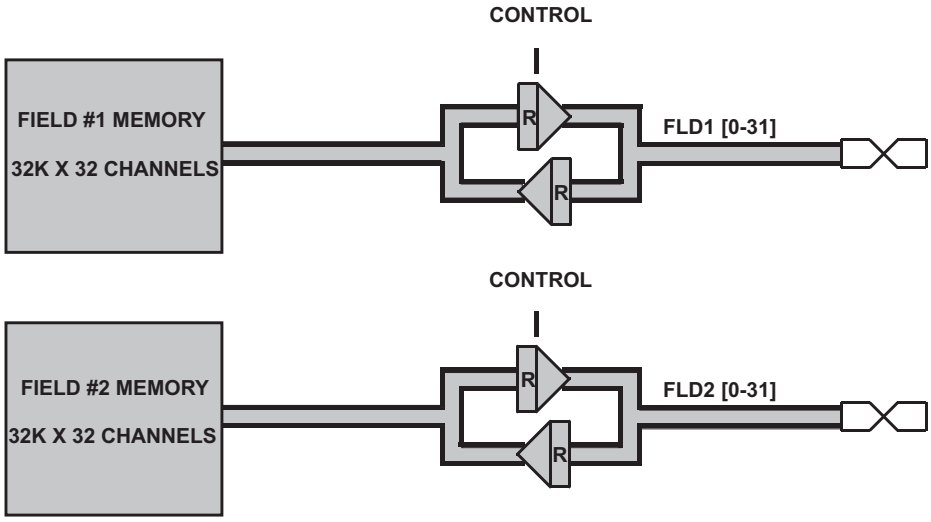


FIGURE 3-4 FIELD MEMORY/REGISTER BLOCK DIAGRAM

The field memory is allocated in groupings called tables. Tables are used in the BE-64 to store stimulus or response data from the UUT. Each table is dedicated to a specific UUT operation. Up to 256 tables may be defined. Additional tables are allocated depending on the availability of memory. The length of each table may be from 1 word to the full 32K of memory. The maximum data rate of the field memory is 25 MHz.

The field registers are separated into two groups called FLD1 and FLD2. The two fields are identical, yet independently controlled. For example, one field could be in the output mode while the other is in the input mode. The two fields are used to emulate synchronous interfaces such as the address and data bus. The field registers are programmed by the timing memory. The timing memory specifies the following settings of the field registers:

- Field Direction: The field direction can be set to input, output or external control. External field control allows the field to be bi-directional.
- Output Register: For fields programmed as output or external, the output register can be turned on or off. Turning the output register off puts the field register in a transparent mode, i.e., field memory is passed immediately through the register.
- Output Control: For fields programmed as output or external, the strobe and enable signal source can be internal or external. Internal signals are programmed in the timing memory.
- Input Control: For fields programmed as input or external the strobe signal source can be internal or external. The internal strobe is programmed in the timing memory.

Refer to section 3.4.4 for a complete description of the timing memory.

3.4.4 Timing Memory

The Timing Memory of the BE-64 is split into 16 separate timing memories called timing sets as depicted in figure 3-5.

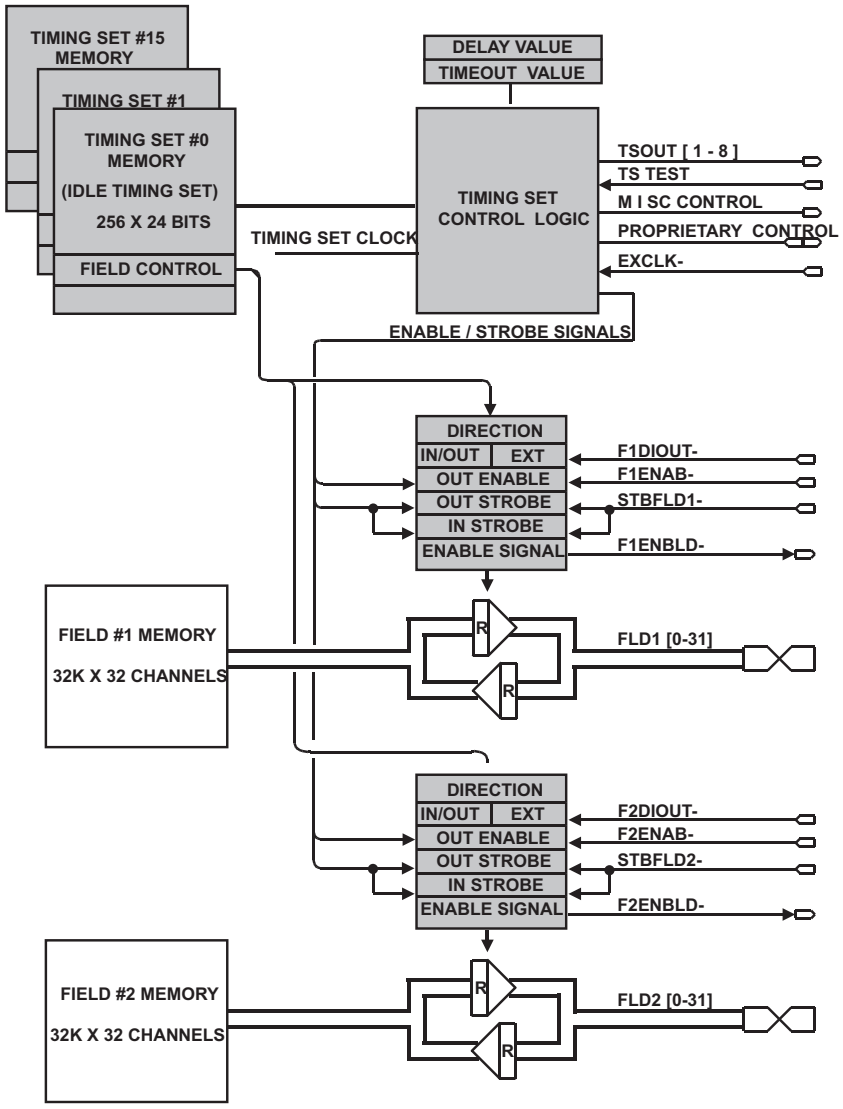


FIGURE 3-5 TIMING MEMORY BLOCK DIAGRAM

The first seven timing sets have pre-defined names and functions as described below:

1. IDLE: Used in the idle sequence, refer to section 3.4.5.1.
2. WRITE_MEM: Used for RAM test functions, Drives front panel signals DATR/W-T low and MI/O-T high.
3. WRITE_IO: Drives front panel signals DATR/W-T and MI/O-T low.
4. READ_MEM: Used for RAM and ROM test functions, Drives front panel signals DATR/W-T and MI/O-T high.
5. READ_IO: Drives front panel signals DATR/W-T high and MI/O-T low.
6. INT_ACK: Drives front panel signal INTCYC-T low.
7. BUS_TEST: Used for BUS test function, Drives front panel signal BUSTST-T low.

The timing set memory is connected to the UUT via the timing set control logic. The control logic contains the settings for the timing set clock as well as the delay and timeout values. The timing set control logic also contains Miscellaneous Control Signals, Proprietary Control Signals, as well as the external clock input signal. Descriptions of these signals can be found in Appendix G.

Each timing set can be viewed as a structure which defines the following data:

- Field Register Settings.
- 256 timing states called ‘cells’.

3.4.4.1 Field Register Settings

The Field Register settings define how the 64 I/O field registers are configured. The field registers are configured in two separate groups of 32 channels each called fields. The following selections must be programmed for each field:

- Field Direction.
- Output Mode.
- Input Mode.

3.4.4.1.1 Field Direction

Each timing set (1 of 16) specifies the field direction to be either input, output, or controlled externally (F1DIOUT-, F2DIOUT-), see figure 3-5. When configured to the output mode, the present word from the 32 bit x 32K memory is available for output to the 32 bit I/O field bus. When configured to the input mode, the present data on the 32 bit I/O field bus may be strobed into the input register and subsequently transferred to the field memory at the end of each timing set cycle.

3.4.4.1.2 Output Mode

When the field is configured to the output mode, the timing set specifies whether or not it is registered. If it is registered, then a strobe signal is required to transfer the data from the field memory into the output register, see figure 3-5. The timing set specifies the strobe signal to be from either the timing set (STROBE FIELD1, STROBE FIELD2) or the external strobe line (STBFLD1-, STBFLD2-). If it is not registered, then the data is buffered directly from the field memory to the I/O field bus.

Both registered and non-registered configurations (output mode) require an enable signal, see figure 3-5. The timing set specifies the enable signal to be from either the timing set (ENABLE FIELD1, ENABLE FIELD2) or the external enable line (F1ENAB-, F2ENAB-). When disabled, the I/O field bus is in the high impedance state.

3.4.4.1.3 Input mode

If the field is configured to the input mode, then a strobe signal is required to transfer the data from the I/O field bus into the input register, see figure 3-5. The timing set specifies the strobe signal to be from either the timing set (STROBE FIELD1, STROBE FIELD2) or the external strobe line (STBFLD1-, STBFLD2-).

3.4.4.2 Timing States/Cells

The 256 timing cells in each timing set represents a unique state in a timing cycle. The transition from cell to cell is made on the falling edge of the timing set clock, i.e., every cell is one period of the clock.

The maximum signal rate of the timing set clock is 50 MHz (20 nsec minimum cell width). Figure 3-6 is an example of a 12 cell timing set.

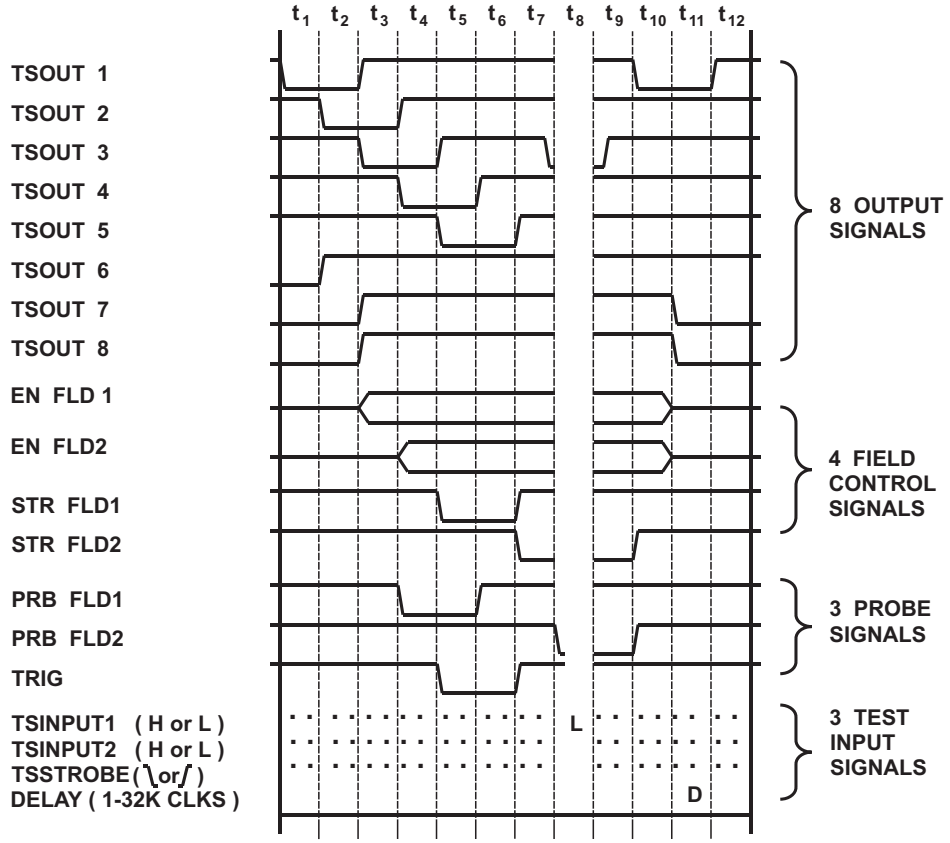


FIGURE 3-6 TIMING SET SIGNALS

Each timing set can be programmed from 2 to 256 cells. The total number of cells must be an even number. Two of the timing cells (first cell and last cell) perform a special function. During the first cell, data from the field memory is enabled to the output field registers. During the last cell data from the input field registers is written to the field memory only if the field is set to input.

For each cell the following data must be programmed:

1. TTL level for the eight timing set outputs (TSOUT1 - TSOUT8).
2. Internal Field Control settings for both fields.
3. Probe and trigger levels.
4. Test Input/Delay code.

3.4.4.2.1 Timing Set TSOUT Signals.

The eight TSOUT output signals are general purpose programmable signals available for the UUT or test fixture interface. They can be programmed to a high or low level with 20 nsec resolution. The TSOUT signals are typically used to simulate signals such as address strobes, data strobes, read/write control, etc.

Additionally TSOUT8 is used as the internal enable for the byte enable memory outputs.

3.4.4.2.2 Timing Set Field Control Signals

The two EN FLD signals are controls to the respective I/O field. When selected, and when the respective I/O field is in the output mode, the EN FLD signal sets the I/O field from the high impedance state to an active state.

The two STR FLD signals are controls to the respective I/O field. When the I/O field is in the output registered mode, the STR FLD signal strobes the data from the field memory into the output register. The data transfer occurs on the positive to negative transition of the strobe signal. Since data from the field memory is not available until after the beginning of the timing set, the strobe signal must occur after the first cell (t1).

When the I/O field is in the input mode, the STR FLD signal strobes the data from the I/O field bus into the input register. The data transfer occurs on the positive to negative transition of the strobe signal. Since the data in the input register is transferred into the field memory during the last cell of the timing set the strobe signal must occur 20ns prior to the end of the last cell.

3.4.4.2.3 Timing Set Probe Signals

The two PRB FLD signals can be routed to one of the VXI TTLTRG lines, TTLTRG0 - TTLTRG3. The PRB FLD signals are typically used to trigger the UUT or other VXI instruments. The user should program these signals within the enable time of the respective EN FLD signal.

The TRIG signal is gated with the memory table sync signal. This combinatorial signal is available on the front panel through either the BNC (SYNC) or J3 pin 15B (Rev. A PCB only)and is typically used to trigger an oscilloscope. This signal can also be routed to one of the VXI TTLTRG lines, TTLTRG4 - TTLTRG7.

3.4.4.2.4 Timing Set Test/Delay Options

The two TSINPUT input signals can be sampled by the timing set. These signals allow the timing set to “handshake” with the UUT. They can be tested for either a logic “0" or a logic ”1" state. Figure 3-6 shows TSINPUT1 defined to test for a logic “0" in timing cell eight (t8).

If the tested condition is true, then the timing set advances to the next state on the next falling edge of the timing set clock. If the tested condition is false, then the timing set enters a “WAIT” state. It remains in the “WAIT” state until one of the following conditions is met:

- The tested condition becomes true.
- The programmable timeout function is detected.
- The VXI controller sets execute mode to STOP or RESet.

The TSSTROBE input signal is similar in function to the TSINPUT signals with the exception that the timing set tests for the occurrence of either a positive or negative edge.

The edge detection logic will reset from any of the following conditions:

- The initiation of an idle cycle.
- Immediately after a tested true condition is detected.
- During the last cell of the timing set.

Therefore, an edge can be detected after the start of the timing set and multiple edges can be detected during the timing set.

The detection of a positive edge does not affect the negative edge logic, and vice versa.

The edge detection logic is reset during the last cell of the timing set, therefore, TSSTROBE input signal cannot be tested during this time.

The DELAY option allows the user to program one or more cells with a delay from 1 to 32K clocks. Since there is only one delay counter, all of the delays in the timing set will be the same user defined length. Figure 3-6 shows cell t11 defined with a delay.

Note: Only one test or delay code is allowed per cell, i.e., you can’t test TSINPUT1 low and TSINPUT2 high in the same cell.

During a test or delay cell, a timeout counter (programmable from 1 to 32K clocks) will start counting. When enabled and timeout count is reached, the timeout logic will force the tested condition true, set the appropriate bit in the VXI status register, and illuminate the TIMEOUT LED on the front panel.

3.4.5 Sequence Control

Once the field memories and the timing set memories are loaded, the user can define a sequence or a series of sequences to be executed, see figure 3-7.

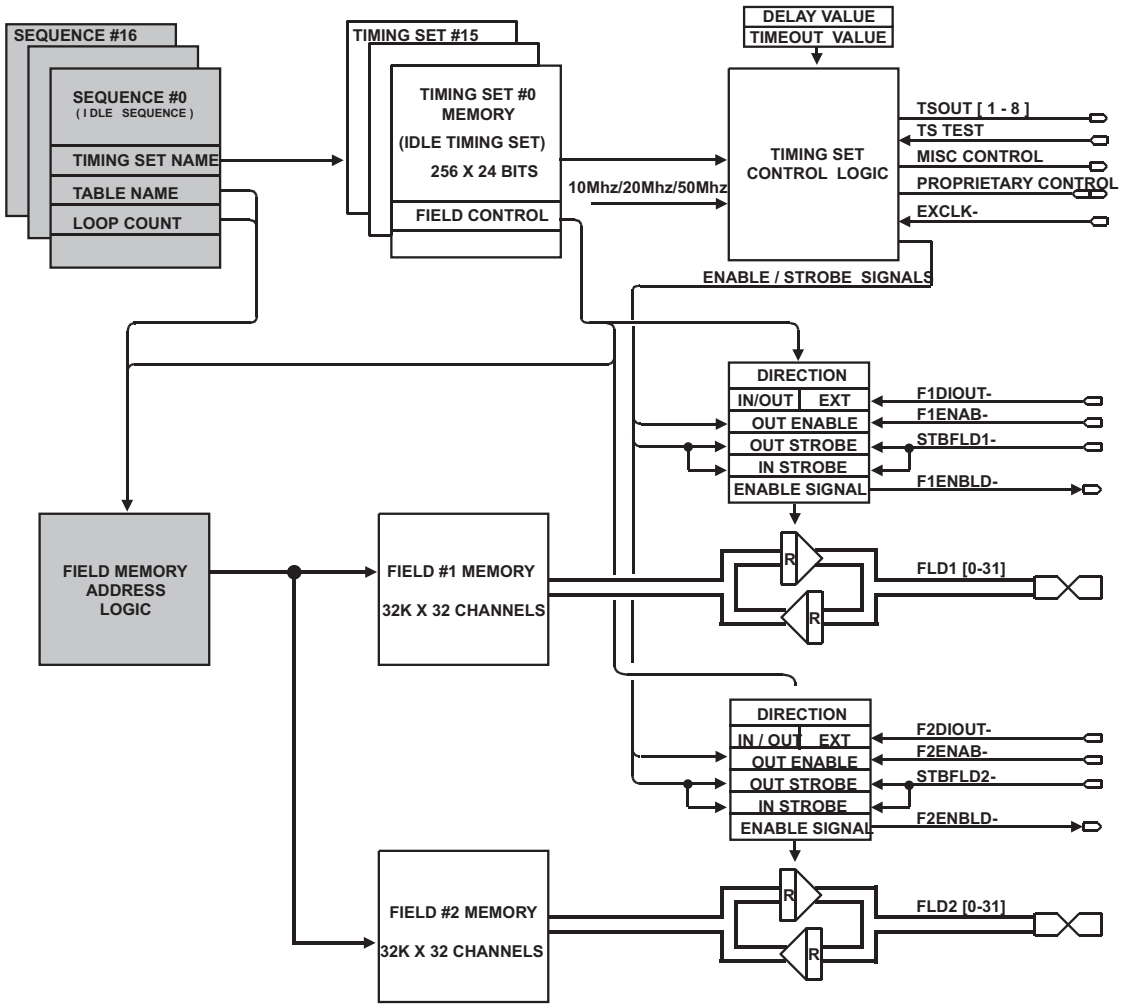


FIGURE 3-7 SEQUENCE CONTROL BLOCK DIAGRAM

A total of 16 sequences can be defined with the following parameters:

- A timing set name.
- A field table name.
- A loop count.

The idle sequence (sequence #0) is limited to just the idle timing set.

3.4.5.1 Idle Sequence

The idle sequence is intended to be active prior to and after the sequence “run time”, see figure 3-8. One of the key functions of the idle sequence is to allow the VXI controller to download new data tables, loop count values, and sequence of operation while the idle sequence is running. This feature enables the

UUT to continue to receive required timing signals while new data is defined for the next operation. The idle sequence, however, must be off while timing sets are being loaded. Set execute mode to RESet from the VXI controller to turn the idle sequence off.

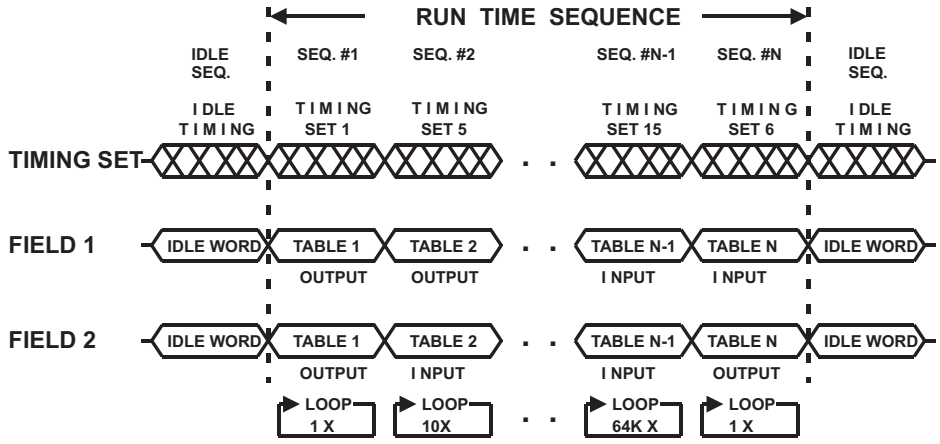


FIGURE 3-8 SEQUENCE OVERVIEW

The idle sequence table is programmed in the last word of the field memories, word 32768. This is the data which is used when the idle sequence is executed prior to or after the sequence “run time”. The idle timing set can also be used for a sequence during the sequence “run time”, in which case any table in the field memories can be used.

3.4.5.2 Sequence Initiation

Once the overall sequence is defined, the VXI controller can start the sequence. If the idle sequence is not running, then the BE-64 initiates it. Upon detection of the last timing set cell and the last word of the idle sequence, the BE-64 steps to Sequence #1, selecting the respective timing set, table name, and loop count. There is zero delay while changing timing sets and data tables or while looping any of the data tables.

The BE-64 remains in Sequence #1 until the last loop, the last word of the table, and the last cell of the timing set is detected. At this time the next sequence is accessed or, if the series of sequences is complete, the BE-64 returns to the idle sequence.

3.4.5.3 Sequence Looping

Each sequence can be looped from 1 to 64K times, or continuously. When looping, there is zero delay time between the end of one sequence and the beginning of the next sequence. When all the loops of one sequence are complete, then the next sequence is executed. Stopping a continuous loop sequence can be accomplished in one of three ways:

- Set execute mode to STOP from the VXI controller. The timing set completes through to the last cell, the data table does not continue with the next word, and the idle sequence is initiated.
- Set execute mode to RESet from the VXI controller. The timing set does not complete through to the last cell, the data table does not continue with the next word, the idle sequence is not initiated. The I/O channels are in the state defined by cell 1 of the idle cycle.

- Set execute mode to SINGle from the VXI controller. The BE-64 completes through to the last timing set cell of the last word of the last table, of the continuous looping sequence, and then starts the table of the next sequence. If there isn't another sequence to run, then the idle sequence is initiated.

Figure 3-9 below represents the operational states of the BE-64.

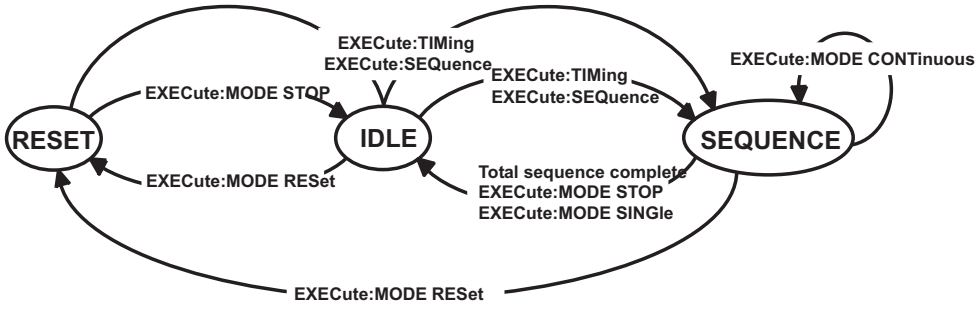


FIGURE 3-9 SEQUENCE STATE DIAGRAM

Additionally, the list of sequences can be commanded to run continuously. In this mode, the last defined sequence will be followed by the first defined sequence without returning to the idle sequence. Exiting this continuous loop is the same as before.

When the "run time" is complete, the VXI controller can interrogate the input field memories, request CRC's to be generated on the input field memories, and execute other commands, including the execution of another sequence of operations.

3.4.5.4 UUT Bus Acquisition Operation

The BE-64 allows synchronous UUT bus control through the use of the BUSREQ-, and BUSACK- sequencer control lines. These signals are active when the BE-64 is in the IDLE mode or while executing other timing sets or sequences as illustrated in figure 3-10. This hardware function can be enabled or disabled by the user via software.

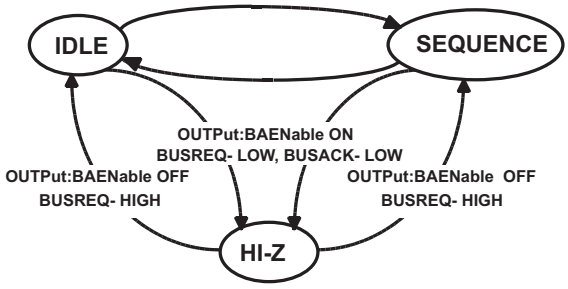


FIGURE 3-10 DMA STATE DIAGRAM

The operation of these two signal lines is enabled or disabled via the following SCPI commands:

```
OUTPut:BAENable ON        Enables UUT bus access
OUTPut:BAENable 1
OUTPut:BAENable OFF              Disables UUT bus access
OUTPut:BAENable 0
```

BUSREQ- is an active low signal that is pulled high internally on the BE-64 through a 3.3K resistor. With bus access enabled, assertion of this signal (logic 0) will cause the BE-64 to synchronously stop operation in the following manner;

1. BUSREQ- is internally synchronized by the TIMING SET CLOCK, generating an internal BUSREQuest pending signal.
2. When the currently running TIMING SET reaches it's last cell LASTBIT- is generated. On the next clock edge the following events occur:
   a. A WAIT- condition is generated for the TIMING SET sequencer. The TIMING SET is actually sitting at the first cell of the next timing set to be executed.
   b. The pending BUSREQuest forces FLD1, FLD2, and TSOUT lines TRISTATE.
   c. BUSACK- is asserted (logic 0) to the UUT indicating that DMA has been granted. The UUT now has access to the bus and can perform DMA operations.

3 BUSREQ- is de-asserted by the UUT. This is synchronized to the TIMING SET clock on the BE-64 On the same clock edge the following events occur:
   a. The WAIT- condition is removed and the TIMING SET sequencer resumes executing the current timing set from the first cell.
   b. FLD1, FLD2, and TSOUT lines are enabled to operate normally.
   c. BUSACK- is de-asserted by the BE-64

3.4.6 Programmable I/O

The BE-64 has 24 programmable I/O. This logic consists of three 8 bit registers, figure 3-11. Two of the registers (PIO1, PIO2) can be configured as either input or output while the third register (OUT) can be configured as either an output register or a counter.

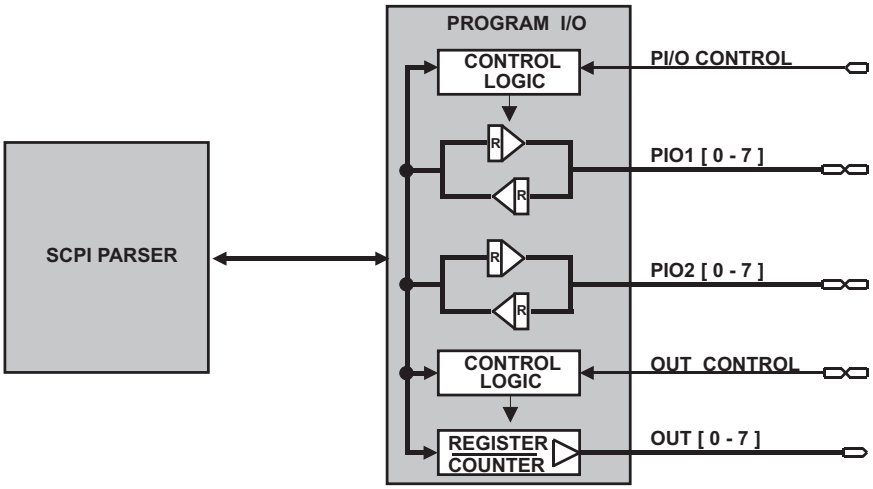


FIGURE 3-11 PROGRAMMED I/O BLOCK DIAGRAM

3.4.6.1 PIO1/PIO2

PIO1 and PIO2 are two independent 8 bit I/O registers, see figure 3-11. Each register can be configured as either input or output. When in the input mode, the VXI controller can sense any bit of the registers for either a high or low condition. When in the output mode, the VXI controller can either set or reset any bit of the registers. Upon power-up or a system reset, both registers are set to the input mode.

These two programmable I/O registers are accompanied by two PIO CONTROL lines, PIOEN- and PIOSTB-. Descriptions of these signals can be found in Appendix G.

3.4.6.2 OUT0-7

The third 8 bit register, OUT0 through OUT7, can be configured as either an output register or a counter. When in the output register mode, the VXI controller can either set or reset any bit of the register. When in the counter mode, the VXI controller can define a first count address. The first count address is saved and may be used to load the counter later when the terminal count is detected.

The counter is accompanied by six OUT CONTROL lines, OUTEN-, CNTCK-, CNTEN-, CNTLD-, UP/DN- and OUTCAR-. Descriptions of these signals can be found in Appendix G.

The counter logic has been designed such that the control signals are compatible with the Proprietary Control Signals shown on the timing set. With the proper connection, the counter can be used to generate address bits A16-A23 of a UUT memory address bus (with A0-A15 generated by Field 1). This enables memory card testing where 16 megabytes of RAM can be continuously accessed with zero delay between memory address locations.

3.4.7 Byte Enable Memory

Most bus and microprocessor structured interfaces have separate control lines to indicate the present size of the data bus (byte, word, or longword). Moreover, byte and word transfers may be defined to be in the LSB or MSB positions. The byte enable output signals are defined by respective VXI compatible commands. The active condition of the BE0- through BE3- is defined by the present transfer type (byte, word, or longword), as well as Bit 0 and Bit 1 of Field 1 (which represents A0 and A1 of the address bus). These signals are further qualified by timing set signal #8 (TSOUT8).

Also see section 4.4.6.

3.4.8 SYNC

SYNC-T, as well as SYNC (front panel BNC), provide a programmable sync pulse. SYNC is available through a BNC in order to trigger a scope. The operator must define a sync position within the field memory. This would be the particular word in a table. The sync pulse logic must be enabled. Also, TRIG in the timing set must be programmed low when the sync pulse is to occur during the timing set cycle.

To insure a single pulse, the TRIG signal should never be programmed with a low in the first cell and a high in the last cell. If a sync pulse at the beginning of cell 1 is desired, then it is recommended to program TRIG low for the entire timing set.

The sync pulse cannot be programmed while the sequence is running.

4 OPERATION

4.1 INTRODUCTION

This section describes the operation of the BE-64. It begins with a overview of the SCPI (Standard Commands for Programmable Instruments) remote command language, followed by programming examples and tutorials.

SCPI commands are sent to the BE-64 via the VXI word serial protocol. The following sections describes how to program the BE-64 using the SCPI language. Refer to section 5 and appendix A and B for more information on SCPI and the command language.

4.2 SCPI CONVENTIONS

SCPI formats (version 1991.0) are used in this instrument to provide a general purpose programming structure.

4.2.1 SCPI Command Structure

The basic structure of a SCPI command is:

PROGRAM_HEADER PARAMETER_LIST

4.2.1.1 Program Header Definition

The PROGRAM_HEADER consists of two distinct types, common command headers and instrument control headers.

4.2.1.1.1 Common Command Header

A common command header is a command keyword preceded by an asterisk (*). The command keyword is a string of ASCII characters that represent a specific command.

Examples:

```
*RST
*IDN?
```

4.2.1.1.2 Instrument Control Header

An instrument control header is a hierarchical command keyword list separated by colons (:). The following diagram illustrates the tree-like structure of the instrument-control headers:

Each instrument control command keyword is a string of ASCII characters that represent a specific command or subsystem.

The upper case letters of each keyword are the short form of the keyword, the upper and lower case letters are the long form of the keyword. Only the entire short or long form of each keyword is valid. Command keywords enclosed within square brackets [ ] are default and may be excluded from the header.

Examples:

```
OUTPut:TTLTrg1:SOURce
OUTP:TTLT1:SOUR
```

```
TABLe:DEL:ALL
```

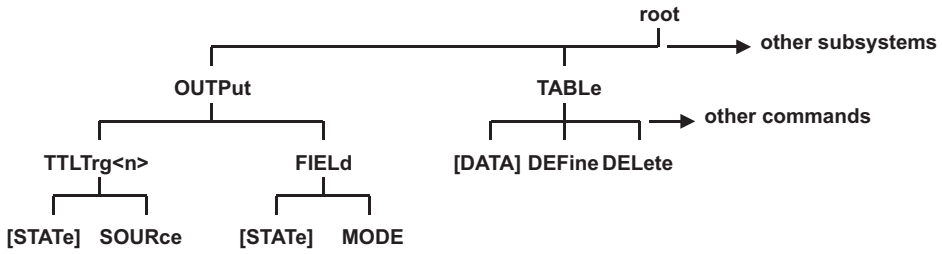


FIGURE 4-1 SCPI TREE-LIKE STRUCTURE

4.2.1.2 Parameter List Definition

The parameter list defines the settings for the specified command. Some commands may not require any parameters (i.e. TABLe:DELete:ALL) whereas other commands may have several parameters (TABLe:DEFine ADDRESS,16).

4.3 BUS EMULATION TUTORIAL

This section is an abbreviated version of Appendix C, GUIDE TO BUS EMULATION. Before any discussion on programming the user should be familiar with Talon's approach to bus emulation testing.

Bus Emulation is a technology which enables an engineer to simulate all the signal characteristics of a given interface. Programmable bus emulation allows the user to software configure the programmable bus emulator to simulate literally an infinite number of digital interfaces.

Bus Emulation Testing is a test method in which the unit under test (UUT) is exercised, in real time, through all of its functions. This is accomplished by precisely simulating all the interfaces into and out of the UUT as well as transferring functional data to/from the UUT.

4.3.1 Word Generator View

Since "bus emulator" is a relatively new technology, we will first describe the basic concept of a word generator. This will establish a baseline for the bus emulation description. It should be noted that the BE-64 module is capable of executing all the concepts which are described in this section.

A basic digital word or pattern generator is composed of a number of I/O channels where the group of channels define a single "word". Each channel is further described by a serial data stream. When the

Talon BE-64 is looked at from this viewpoint, its word generator capacity is 64 I/O channels with a 32K word depth as illustrated in figure 4-2.

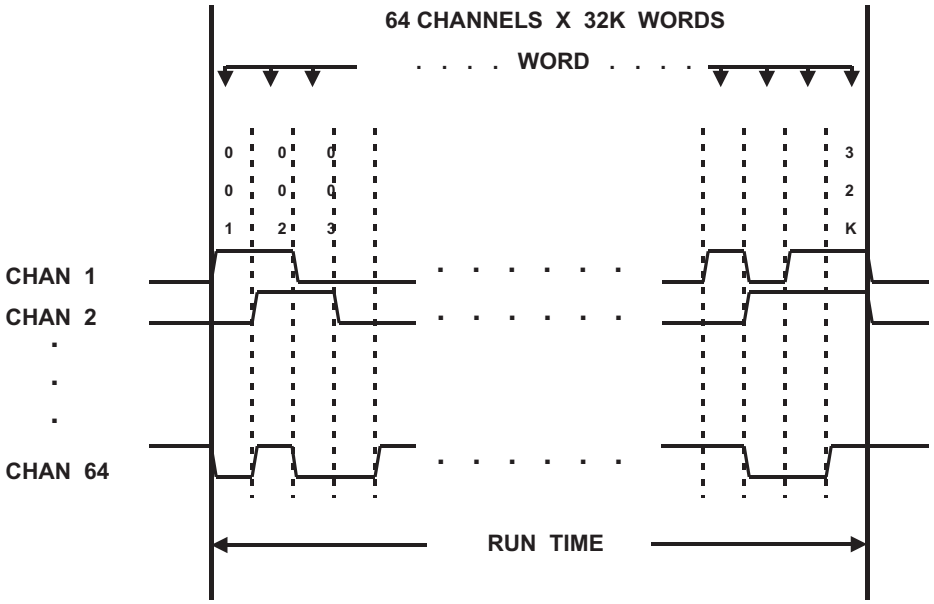


FIGURE 4-4 WORD GENERATOR VIEW

In a run mode, the data would be output across the channels starting with word 1 and continuing through word 32K, generating a continuous stream of data. The time required to execute the entire table is referred to as RUN TIME.

For illustrations' sake we will consolidate figure 4-2 into the abbreviated figure 4-3 below.

FIGURE 4-2 SIMPLIFIED WORD GENERATOR VIEW

When using a word generator, it is desirable to subdivide the entire word generator memory (field memory) into smaller sections, each section dedicated to a specific UUT operation. The BE-64 refers to the divided memory as tables, figure 4-4. The BE-64 can be divided into hundreds of tables. The table length can be individually configured from 1 word to the full 32K memory depth. The RUN TIME is the time required to execute all the tables in the sequence. The execution of the tables is continuous with zero dead time between tables.

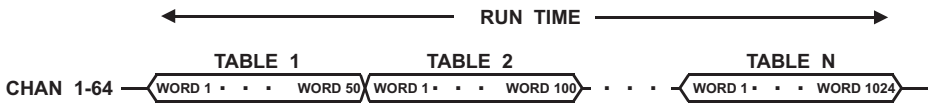


FIGURE 4-3 FIELD MEMORY TABLES

The ability to individually loop the tables is often desirable in order to create a repeating stimulus to the UUT. The BE-64 allows individual tables to be looped from 1 to 64K times, figure 4-5. In addition, tables can be looped indefinitely with the stop or exit from a loop directed by the VXI Slot 0 Controller.

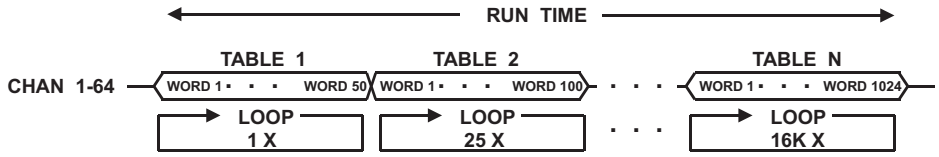


FIGURE 4-5 TABLE LOOPING

The BE-64 incorporates a unique function which is not found on present data generators, this feature being the idle cycle. The idle cycle allows the user to define a timing and data sequence prior to and after the “run time”, figure 4-6. This idle cycle can be programmed such that no signal activity is seen by the UUT, or a complete repetitive timing cycle with associated data patterns can be defined.

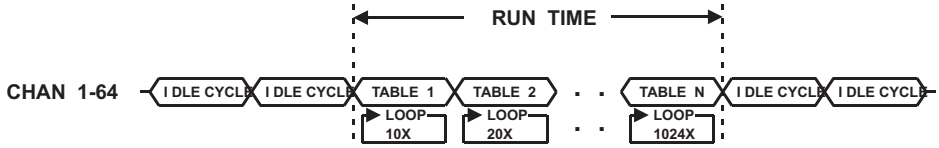


FIGURE 4-6 IDLE CYCLE

One of the key functions of the idle cycle is the ability of the VXI controller to download new data tables, loop count values, and sequence of operation while the idle sequence is running. This feature enables a UUT to continue to receive required timing signals while new data is defined for the next operation.

4.3.2 Bus Emulation Overview

In general, word generator parameters describe the sequencing of data from one word to the next, and from one table of data to the next table. Bus emulation describes what happens “inside” each word. For example, let’s take a look at what’s happening inside word 2 of table 1, figure 4-7.

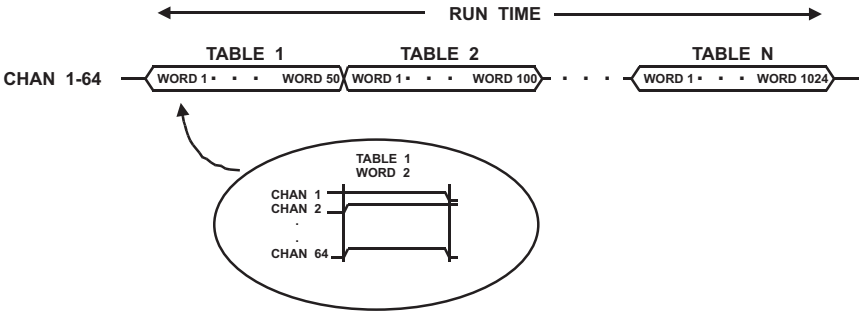


FIGURE 4-7 TABLE WORD EXAMPLE CLOSE-UP

The BE-64 allows an individual word to be divided into timing increments, cells, with resolution up to 20 ns

per cell. Figure 4-8 depicts word 2 of table 1, where word 2 has a total period of 200ns and is referred to as a timing bus cycle.

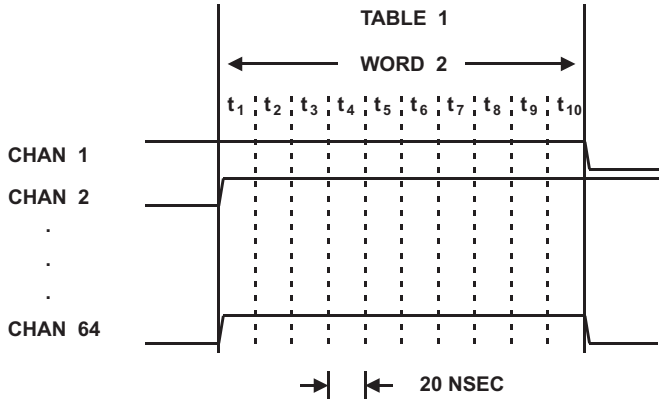


FIGURE 4-10 TABLE WORD TIMING BUS CYCLE

Figure 4-9 again depicts word 2, table 1, however in this figure we will break up the 64 channels into two fields, an address field and a data field. This configuration will allow our example to follow a more traditional bus structure interface.

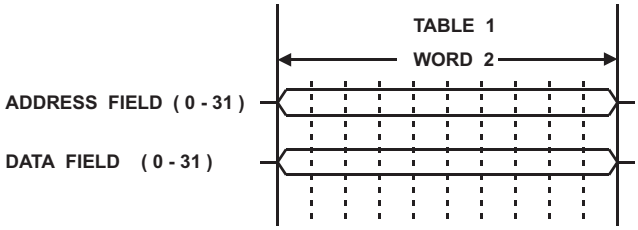


FIGURE 4-8 BUS CYCLE FIELDS

In most applications, the signals comprising a word in a field are not active during the entire word period. As shown in figure 4-10, the address field period may be active from 20ns (t1) to 180ns (t9), while the data field is active from 60ns (t3) to 160ns (t8).

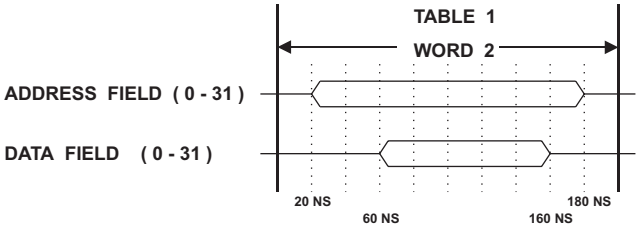


FIGURE 4-9 BUS CYCLE FIELD TIMING

In order to achieve the “bus signals” depicted in figure 4-10, the buses require control signals. In

particular, they require an ADDRESS ENABLE and DATA ENABLE signal, figure 4-11. These signals are generated from a separate timing generator, nomenclated the “timing set”.

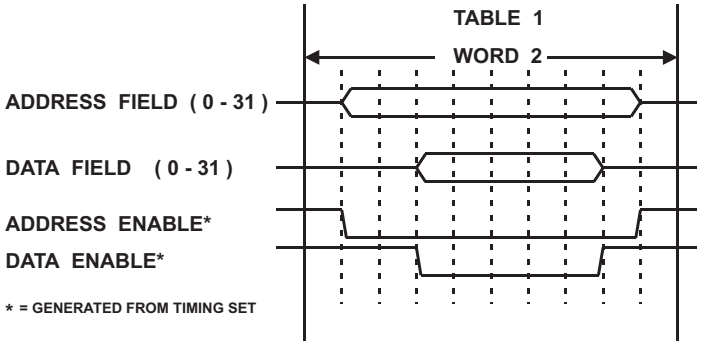


FIGURE 4-12 TIMING SET

Typically, the UUT requires additional control signals which further define the “bus cycle”. The BE-64 timing sets include eight output lines (TSOUT) and four enable/strobe lines, two per field.

Once active data has been placed on the data bus and the data enable signal has been asserted, there must be a means to determine that the UUT is ready for data. The timing set has the ability to test a UUT signal and enter a “Wait” state until the test condition is true. This sequence is often referred to as a “handshake” sequence.

Figure 4-12 indicates that the timing set will remain in cell t5 until the UUT READY- signal goes low. Once the UUT READY- = Low condition is met, the timing set will continue. If the test condition does not occur in a specified time, a timeout condition will force the continuation of the timing set and inform the VXI controller that a timeout occurred. The timeout logic can be disabled, which in turn will cause the BE-64 to remain in state t5 until commanded to complete via stop or reset commands by the VXI controller.

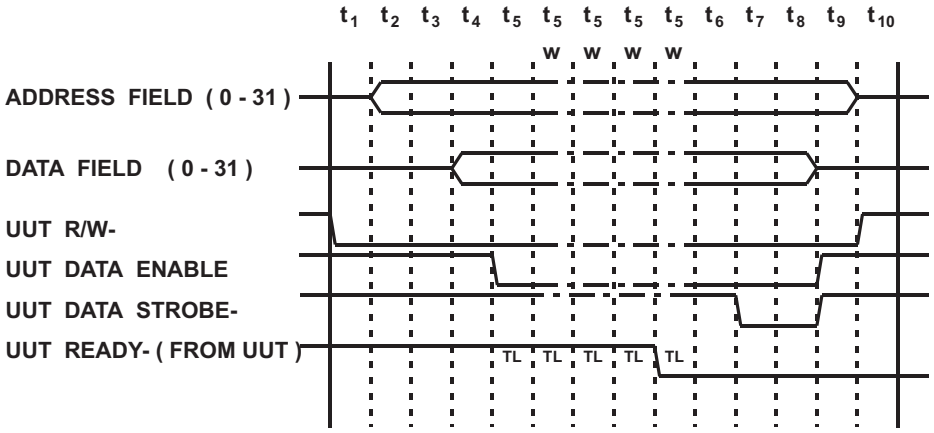


FIGURE 4-11 HANDSHAKE EXAMPLE

The BE-64 incorporates 16 timing sets, each timing set programmed to simulate a particular bus cycle type or UUT timing cycle.

Bus emulation is programmed in the BE-64 by linking the Field Memory with the Timing Set memory.

4.4 PROGRAMMING EXAMPLES

The following sections will provide examples of programming the BE-64 using SCPI commands.

The operation and programming of the BE-64 is closely linked with the implementation of the interconnect between the “tester” of which the BE-64 is a part, and the UUT (Unit Under Test).

Typically resources of the BE-64 will be split between the UUT and the test fixture.

All of the following examples will use a hypothetical UUT which is described in the following block diagram, figure 4-13.

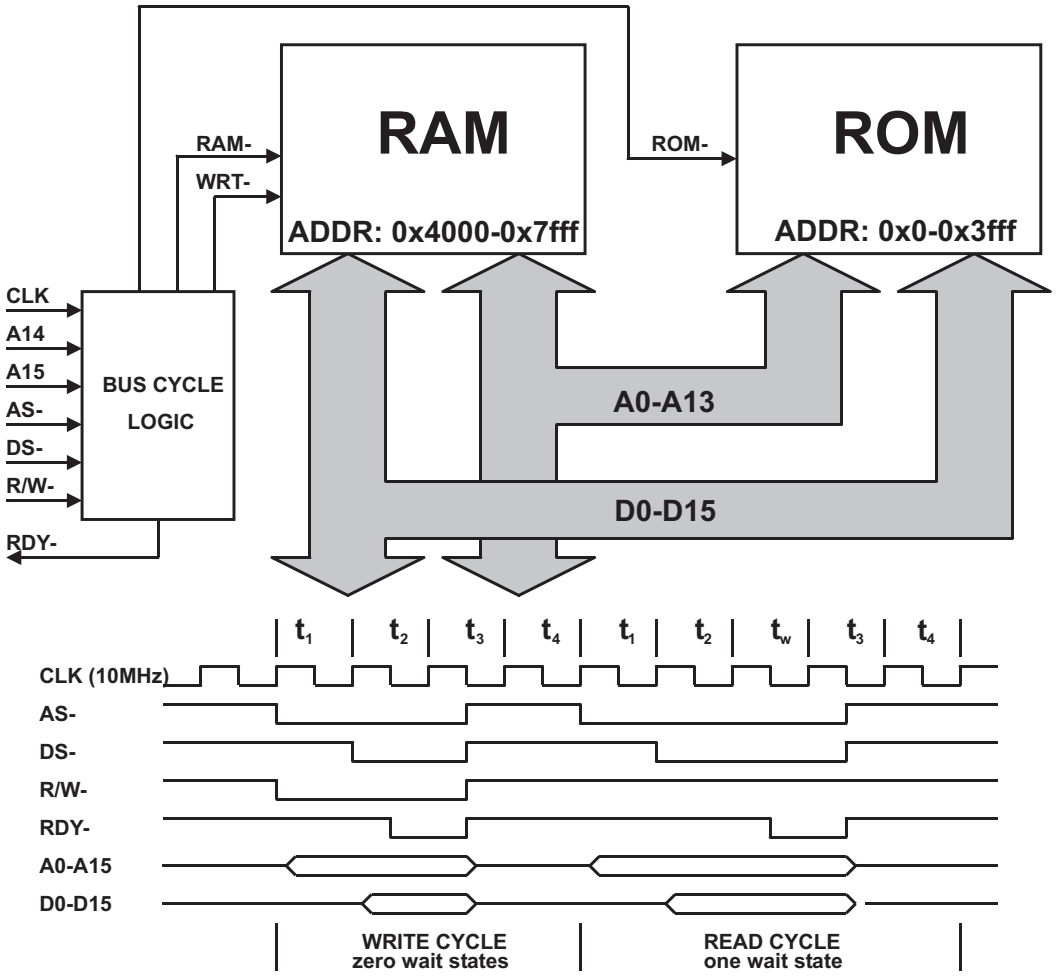


FIGURE 4-13 SAMPLE UUT BLOCK DIAGRAM

4.4.1 Programming the Timing Memory

As previously discussed the timing memory is segmented into timing control registers and sixteen timing

sets as illustrated in figure 4-14. Each timing set contains a separate field register setting as well as 256 timing cell states.

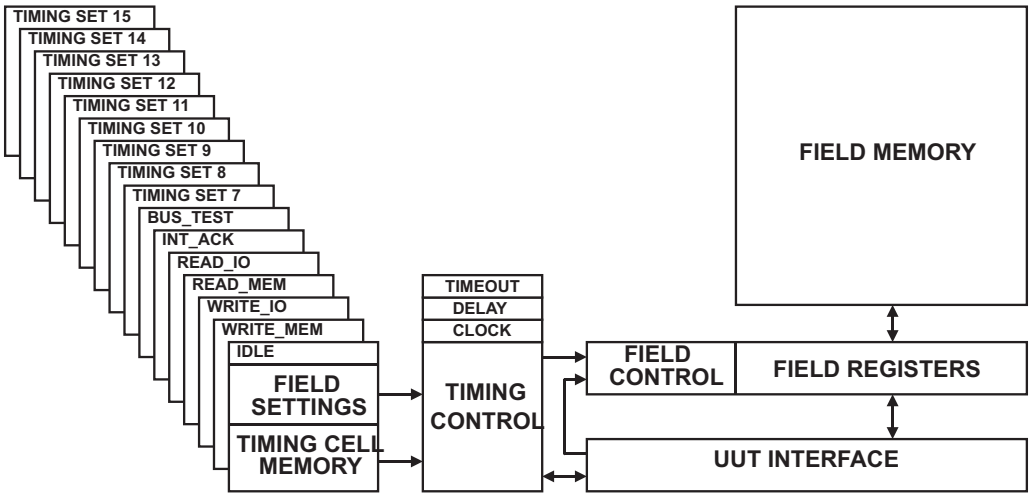


FIGURE 4-14 TIMING MEMORY

The timing memory is generally programmed in the following order:

1. Program the timing setup data (clock, delay and timeout).

For each timing set:

2. Define the timing set.
3. Program the field control settings for both fields.
4. Program the timing cell data.
5. Program the Handshake cells.

The remainder of this section will discuss all the steps outlined above and the commands used to perform them.

4.4.1.1 Programming The Timing Setup Data

The Timing Setup data is global to all the timing sets and is comprised of the following selections:

- Clock Source.
- Delay Count.
- Timeout Count.

4.4.1.1.1 Programming the Clock Source

In our hypothetical UUT, shown in figure 4-13, a clock is shown in the bus cycle description. In order to simulate the bus cycles we must first select a clock that will be synchronous with the UUT and provide our timing outputs enough resolution to simulate the control signals (AS-, DS-, R/W-). The pulse width of the timing outputs will be the inverse of the clock frequency, i.e., a 20MHz clock selection will produce timing outputs with a 50ns pulse width.

The BE-64 has four clock selections available, three internal and one external. The format of the command is:

```
TIMing:SETup:CLOCk (EXTernal | 50 | 20 | 10)
```

In our hypothetical UUT the clock source is generated from the card edge, therefore, the internal clock selection will be made as follows:

```
TIMing:SETup:CLOCk 20
```

20MHz was selected to allow a finer resolution in the timing cells, 50ns/cell.

An external clock could be used if the clock source was generated from the UUT.

4.4.1.1.2 Programming the Delay Count

The Delay count specifies the number of clocks a delay cell will wait before continuing to the next cell. This feature is used when a timing specification requires numerous identical states in a timing cycle.

The format of the command to set the timeout delay count is:

```
TIMing:SETup:DELay <delay_count>
```

The <delay_count> range is from 0 - 32768. A <delay_count> of 0 disables the delay.

For example, if our sample UUT cycle definition in figure 4-13 was modified such that the AS- signal is required to be low for 1ms before DS- goes low we could program a delay in cell t1 and set the delay count to 19. Since we selected a 20MHz clock, each timing cell is 50ns so a delay count of 19 will extend the cell by 950ns.

4.4.1.1.3 Programming the Timeout Count

The Timeout count is used to keep a timing set test instruction from locking up the BE-64. When a test instruction is programmed into a timing set cell the timing cycle will not sequence to the next cell until the test condition becomes true. If the test condition never occurs due to a failure of the UUT, the timeout option will terminate the test condition and continue to the next cell.

The format of the command is:

```
TIMing:SETup:CTIMout <timeout_value>
```

The <timeout_value> range is from 0 - 32768. The value programmed specifies the number of clocks to wait before terminating a test condition. A value of zero disables the timeout logic.

For our hypothetical UUT the timing sets will be testing the RDY- signal to determine when the bus cycle is complete, no more than ten wait states are allowed. A timeout of 10 clocks will be programmed by the following command:

```
TIMing:SETup:CTIMout 10
```

The operator can test whether a timeout has occurred by reading the status register

4.4.1.2 Defining A Timing Set

The timing memory of the BE-64 is segmented into sixteen groups called timing sets. Each timing set is identified by a unique ten character name. Seven of the sixteen timing sets have pre-defined names and locations within the timing memory of the BE-64 as listed in table 4-1. The remaining nine timing sets can have user-defined names. Defining a timing set associates a name and size with a physical timing memory location.

The format of the command is:

```
TIMing:DEFine <timing_set_name>,(<number_of_cells> | <timing_set_name>)
```

TIMING SET NAME	A24 Offset (HEX)	FUNCTION
IDLE	0	Executed automatically during dead time.
WRITE_MEM	400	Drives front connector signals DATR/W-T low and MI/O-T high. Used in ram test functions.
WRITE_IO	800	Drives front connector signals DATR/W-T and MI/O-T low.
READ_MEM	C00	Drives front connector signals DATR/W-T and MI/O-T high. Used in ram and rom functions.
READ_IO	1000	Drives front connector signals DATR/W-T high and MI/O-T low.
INT_ACK	1400	Drives front connector signal INTCYC-T low.
BUS_TEST	1800	Drives front connector signal BUSTST-T low. Used in bus test function.

TABLE 4-1 PRE-DEFINED TIMING SET NAMES

The <timing_set_name> can be from one to ten alphanumeric characters, the first must be an alpha character. The <number_of_cells> range is from 2 to 256 in increments of 2.

It should be noted that the BE-64 must be in the reset mode before any of the timing sets can be defined or edited.

In our sample UUT we will be using three of the pre-defined timing sets;

```
EXECute:MODE RESet                Places BE-64 in reset state
TIMing:DEFine IDLE,2
TIMing:DEFine WRITE_MEM,8
TIMing:DEFine READ_MEM,WRITE_MEM  Copies contents of WRITE_MEM to READ_MEM
```

From the sample UUT bus cycle description the minimum bus cycle length is 400ns. With a clock source of 20MHz our number of cells must be less than or equal to 8 for both READ_MEM and WRITE_MEM.

The IDLE timing is a special timing set in that it will execute automatically between execution of runtime sequences. This feature allows critical signals to be generated without interruption while new field memory data can be downloaded/analyzed.

A listing of all the defined timing sets, sizes and A24 memory offsets can be obtained with the following timing directory command:

```
TIMing:DIRectory?
```

4.4.1.3 Programming the Timing Memory

The timing memory is comprised of three functional groups, field control settings, timing cell levels and handshake tests.

There are two methods to program the timing memory. The first method uses SCPI commands to program the fields and cells. The second method uses the A24 access to directly program the timing memory. Programming the timing memory using the A24 access is described in appendix D. The following sections describes programming the timing memory using SCPI commands.

4.4.1.3.1 Programming the Field Control Registers

The BE-64's 64 I/O channels are split into two groups called fields. These fields, referenced as FLD1 and FLD2, are connected to the UUT via the field registers as illustrated in figure 4-14.

The following selections must be programmed for each field register and are illustrated in figure 4-15:

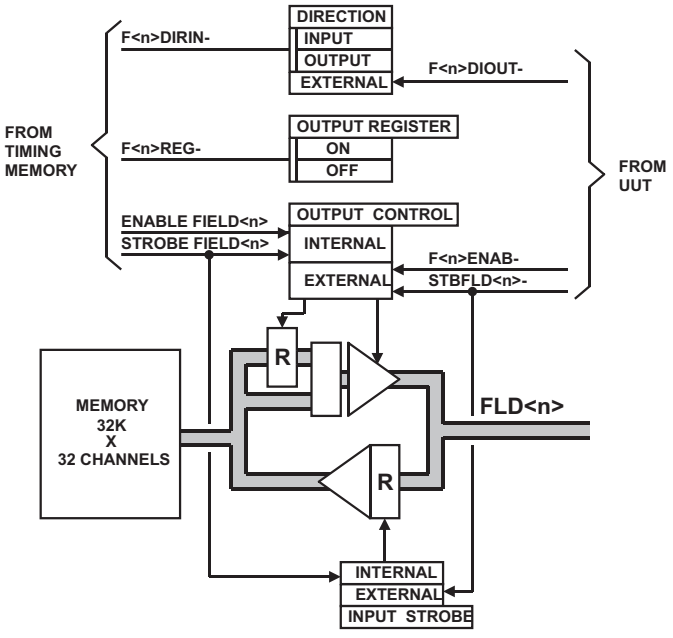


FIGURE 4-15 FIELD CONTROL PROGRAMMING

1. Field Direction
2. Output Register

3. Output Control
4. Input Strobe

The program selections vary according to the direction setting, i.e., the output selections do not need to be programmed if the direction is set to input. Table 4-2 lists the programming selections required for each of the three direction choices (INPUT, OUTPUT and EXTERNAL).

SELECTION/ DIRECTION	OUTPUT REGISTER	OUTPUT CONTROL	INPUT STROBE
INPUT	N/A	N/A	INTERNAL/EXTERNAL
OUTPUT	YES/NO	INTERNAL/EXTERNAL	N/A
EXTERNAL	YES/NO	INTERNAL/EXTERNAL	INTERNAL/EXTERNAL

TABLE 4-2 FIELD DIRECTION/SELECTION CHOICES

The format of the command to set the field register direction is:

```
TIMing:FCONtrol:DIRection <timing_set_name>,<field>,(INPut | OUTPut | EXTernal)
```

The format of the command to set the output register selection is:

```
TIMing:FCONtrol:OREGister <timing_set_name>,<field>,(ON | OFF)
```

The format of the command to set the output control is:

```
TIMing:FCONtrol:OCONtrol <timing_set_name>,<field>,(INTernal | EXTernal)
```

The format of the command to set the input strobe is:

```
TIMing:FCONtrol:ISTRobe <timing_set_name>,<field>,(INTernal | EXTernal)
```

Where <timing_set_name> is one of the defined timing sets, <field> is either FLD1 or FLD2.

In our example UUT setup we are using three timing sets, IDLE, WRITE_MEM and READ_MEM.

The following commands program the field registers for the IDLE timing set. The figure graphically represents the control selections for each field.

```
TIMing:FCONtrol:DIRection IDLE,FLD1,OUTPut
TIMing:FCONtrol:OREGister IDLE,FLD1,ON
TIMING:FCONtrol:OCONtrol IDLE,FLD1,INTernal
TIMing:FCONtrol:DIRection IDLE,FLD2,OUTPut
TIMing:FCONtrol:OREGister IDLE,FLD2,ON
TIMING:FCONtrol:OCONtrol IDLE,FLD2,INTernal
```

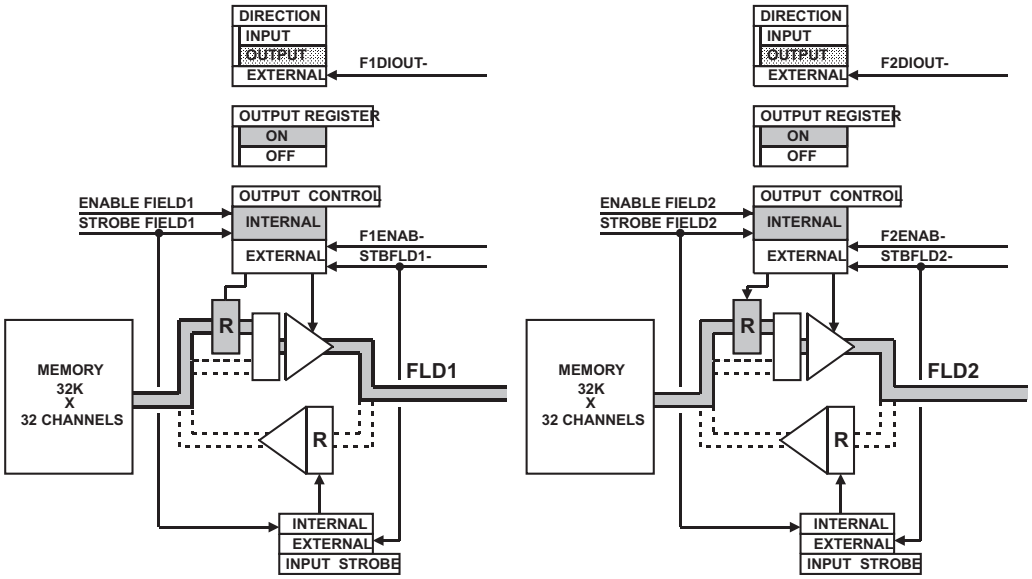


FIGURE 4-16 IDLE TIMING SET FIELD SETTINGS

The following commands program the field registers for the WRITE_MEM timing set. The figure graphically represents the control selections for each field.

```
TIMing:FCONtrol:DIRection WRITE_MEM,FLD1,OUTPut
TIMing:FCONtrol:OREGister WRITE_MEM,FLD1,OFF
TIMING:FCONtrol:OCONtrol WRITE_MEM,FLD1,INTernal
TIMing:FCONtrol:DIRection WRITE_MEM,FLD2,OUTPut
TIMing:FCONtrol:OREGister WRITE_MEM,FLD2,OFF
TIMING:FCONtrol:OCONtrol WRITE_MEM,FLD2,INTernal
```

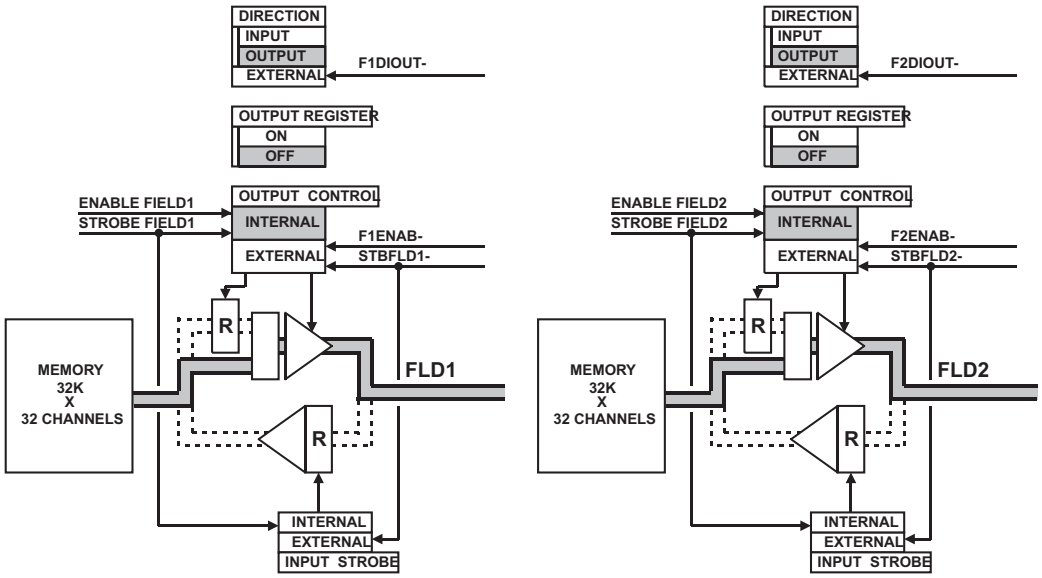


FIGURE 4-17 WRITE_MEM TIMING SET FIELD SETTINGS

The following commands program the field registers for the READ_MEM timing set. The figure graphically represents the control selections for each field.

```
TIMing:FCONtrol:DIRection READ_MEM,FLD1,OUTPut
TIMing:FCONtrol:OREGister READ_MEM,FLD1,OFF
TIMING:FCONtrol:OCONtrol READ_MEM,FLD1,INTernal
TIMing:FCONtrol:DIRection READ_MEM,FLD2,INPut
TIMing:FCONtrol:ISTRobe READ_MEM,FLD2,EXTernal
```

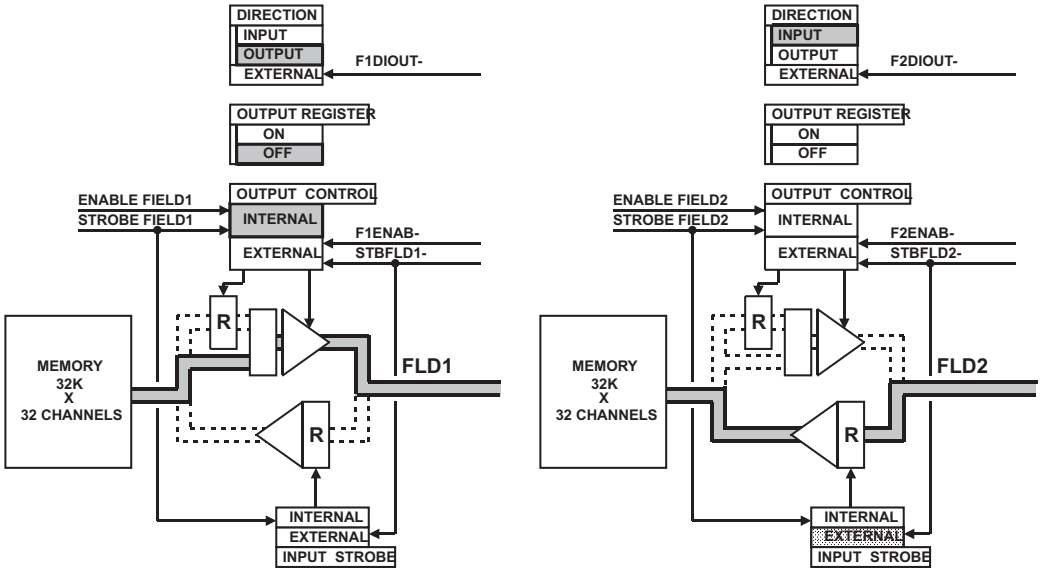


FIGURE 4-18 READ_MEM TIMING SET FIELD SETTINGS

4.4.1.3.2 Programming the Timing Cells

The timing cells are best described by thinking of each cell as a timing state. The length of each state is one period of the selected clock. Each state can be programmed with different TSOUT output levels, handshake test conditions and field control levels. The number of states/cells is set by the TIMing:DEFine command.

There are two SCPI commands to program the timing cells:

```
TIMing:DATA <timing_set_name>,<binary block>
```

and

```
TIMing:CELL <timing_set_name>,<cell_number>,<cell_contents>
```

The <binary_block> represents the settings (TSOUT levels, Internal Field control, Handshake tests and the field control settings described above) for each cell in the named timing set. The format of the binary block data is specified in the TIMing:DATA command description.

The <cell_contents> only represent the TSOUT and Internal Field control levels settings for the individual <cell_number>. The cell contents format is specified in the TIMing:CELL command description, see Appendix F for the Timing Set programming worksheet. From our sample UUT Block Diagram and the sample worksheet listed in figure 4-19, the following commands program the WRITE_MEM timing cells:

```
TIMing:CELL WRITE_MEM,1,#h7EF5
TIMing:CELL WRITE_MEM,2,#h7EF4
TIMing:CELL WRITE_MEM,3,#h7CF1
TIMing:CELL WRITE_MEM,4,#h7CF0
TIMing:CELL WRITE_MEM,5,#h7EF1
TIMing:CELL WRITE_MEM,6,#h7FFE
TIMing:CELL WRITE_MEM,7,#h7FFF
TIMing:CELL WRITE_MEM,8,#h7FFE
```

4.4.1.3.3 Programming the Handshake Tests

Handshake test conditions must be programmed separately when the TIMing CELL command is used to program the cell data. Handshake test codes are embedded within the binary block and therefore are not required for the TIMing:DATA command.

A Handshake test will cause the timing cycle to enter a wait condition, i.e., the next timing cell will not be entered until either the test condition becomes true or a timeout occurs. In our example timing set worksheet TSOUT1 (CLK) is used only as a reference. TSOUT1 cannot be used as the UUT CLK because if a test condition is not true the timing set will loop on the current cell.

There are three events that can be tested for:

1. Input Level of TSINput1 or TSINput2.
2. Rising or Falling edge of TSSTrobe.
3. Specified Delay count is complete.

The format of the commands used to specify a handshake test is:

```
TIMing:TEST:LEVel <timing_set_name>,(TSINput1 | TSINput2),(HIGH | LOW),<cell_number>
TIMing:TEST:STRobe <timing_set_name>,(HIGH | LOW),<cell_number>
TIMing:TEST:DELay <timing_set_name>,<cell_number>
```

The test in cell four is programmed with the following command:

```
TIMing TEST:STRobe WRITE_MEM,LOW,4
```

Timing Set Name WRITE_MEM Sheet 1 of 1

TIMING SET WORKSHEET

CELL	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
TSOUT 1 CLK																				
BIT 0	1	0	1	0	1	0	1	0												
TSOUT 2 AS-																				
BIT 1	0	0	0	0	0	1	1	1												
TSOUT 3 DS-																				
BIT 2	1	1	0	0	0	1	1	1												
TSOUT 4 R/W-																				
BIT 3	0	0	0	0	0	1	1	1												
TSOUT 5																				
BIT 4																				
TSOUT 6																				
BIT 5																				
TSOUT 7																				
BIT 6																				
TSOUT 8																				
BIT 7																				
BYTE LOW	F5	F4	F1	F0	F1	FE	FF	FE												
EN FLD 1 AEN																				
BIT 8	0	0	0	0	0	1	1	1												
EN FLD 2 DEN																				
BIT 9	1	1	0	0	0	1	1	1												
STR FLD 1																				
BIT 10																				
STR FLD 2																				
BIT 11																				
PRB FLD 1																				
BIT 12																				
PRB FLD 2																				
BIT 13																				
TRIG																				
BIT 14																				
BIT 15 (unused)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BYTE HIGH	7E	7E	7C	7C	7E	7F	7F	7F												
TS INPUT 1																				
TS INPUT 2																				
TS STROBE RDY				L																
DELAY _____ CLOCKS																				

BIN---HEX

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

1/92

4.4.2 Programming The Table Field Memory

Similar to the timing memory, the field memory can be programmed using either A24 access or SCPI commands. Appendix D details the A24 programming. The following section describes programming the table memory using SCPI commands.

As discussed in chapter 3 the BE-64's I/O memory is 64 channels wide by 32K deep. The 64 channels are divided into to groups called fields as shown in figure 4-20.

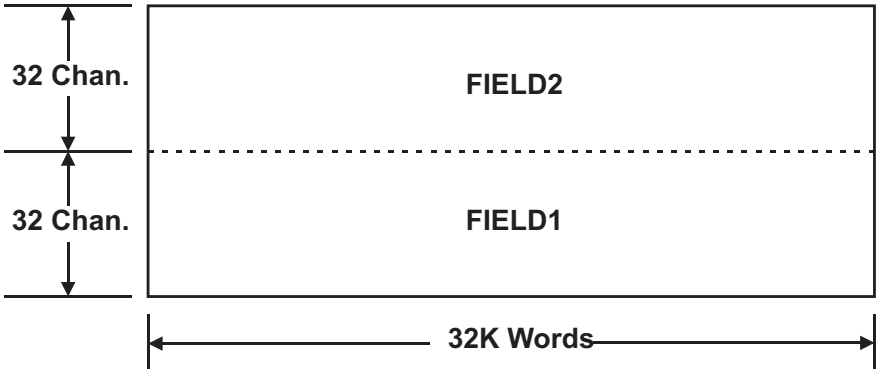


FIGURE 4-20 FIELD MEMORY ALLOCATION

Each 64 bit element is referred to as a word. A sequential group of words is referred to as a table. The BE-64's 32K words can be organized into tables as illustrated in figure 4-21. Tables in the BE-64 are used to store stimulus and response data from the UUT.

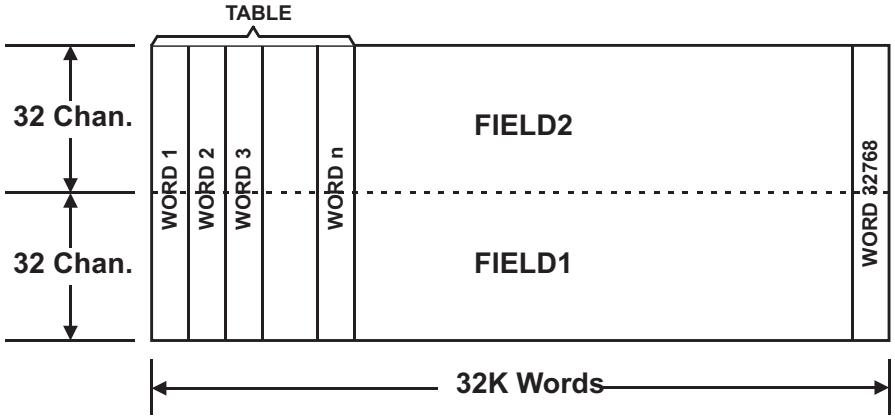


FIGURE 4-19 FIELD MEMORY TABLES

The following command is used to create a new table:

```
TABLe:DEFine <table_name>,(<number_of_words> | <table_name>)
```

The <table_name> can be from one to ten alphanumeric characters, the first must be an alpha character. The <number_of_words> range is from 1 to 32768.

It should be noted that the BE-64 must be in the idle or reset state before any of the tables can be defined or edited.

With our sample UUT we will define two tables each of 16K so that we can load or fill the entire ROM or RAM array.

```
EXECute:MODE STOP          Places BE-64 into idle state
TABLe:DEFine EXPECT,#h3fff
TABLe:DEFine RECORD,EXPECT Copies contents of EXPECT to RECORD
```

A listing of all the defined tables is generated using the following command:

```
TABLe:DIRectory?
```

There are two SCPI commands to program stimulus data into the table memory:

```
TABLe:DATA <table_name>,<block>
```

or

```
TABLe:WORD <table_name>,<word_number>,<FLD1 data>,<FLD2 data>
```

A specific field can be programmed using the following command variation:

```
TABLe:FIELd:DATA <table_name>,(FLD1 | FLD2),<block>
```

or

```
TABLe:FIELd:WORD <table_name>,(FLD1 | FLD2),<word_number>,<field_data>
```

The DATA command programs the field memory using the binary representation of the memory.

The WORD command programs a single word using the ASCII representation of the memory. The field_data value is converted to binary before being stored into the field memory.

Stimulus data can also be generated using the built-in fill functions in the BE-64.

Response data can be read back from the BE-64 by using the query form of the commands listed above.

For our sample UUT we will need to generate the addresses for the RAM in order to write or read RAM data into a table.

```
EXECute:MODE STOP          Places BE-64 in idle state.
TABLe:FIELd:WORD EXPECT,FLD1,1,#h4000
TABLe:FIELd:FILL EXPECT,FLD1,INCRement,1,2
```

The following commands perform the same function for the RESPONSE table.

```
EXECute:MODE STOP          Places BE-64 in idle state.
TABLe:FIELd:WIDTh RESPONSE,FLD1,WORD     Sets the word size to 2 bytes (16 bits).
TABLe:FIELd:DATA RESPONSE,FLD1,#12byte1byte2     byte1 = hex 40, byte2 = hex 0.
TABLe:FIELd:FILL RESPONSE,FLD1,INCRement,1,2
```

4.4.3 Executing a Timing Cycle

Timing Cycles are executed on the BE-64 using the following command,

```
EXECute:TIMing <timing_set_name>,<table_name>
EXECute:TIMing <timing_set_name>,<field1_data>,<field2_data>[,<byte_enable>])
```

The <timing_set_name> selects one of the sixteen timing set memories that will control the current execution.

A table or literal field data can be specified. When a table is used then every word in the table will be executed with the selected timing set. Literal data will be stored into the last field memory address and executed with the selected timing set.

The optional <byte_enable> selects one of the three byte enable codes for the literal data.

In our sample UUT we can write and read one word of data with the following commands,

```
EXECute:TIMing WRITE_MEM,#h4000,#h1234   Write hex 1234 to hex 4000.
EXECute:TIMing? READ_MEM,#h4000,#h0      Read the data at hex 4000
```

BE-64 will return decimal 16384,4660 (4000,1234) hex.

We can read the entire contents of ROM with the following commands,

```
TABLe:FIELd:WORD RESPONSE,FLD1,1,0       Store a zero in the first word.
TABLe:FIELd:FILL RESPONSE,FLD1,INCR,1,2  Fill table with ROM addresses.
EXECute:TIMing READ_MEM,RESPONSE  Reads the entire ROM contents.
```

The results will be stored in FLD2 of table RESPONSE.

4.4.4 Executing a Sequence

As shown above one word or one table can be executed with a single timing set. The BE-64 contains sixteen sequence registers that allow the user to program multiple timing set/table cycles to be executed sequentially.

The following command programs and executes a sequence of timing sets,

```
EXECute:SEQuence <timing_set_name>,<table_name>,<loop_count>{,<timing_set_name>...}
```

The <loop_count> can be used to loop a sequence from 1 to 64K times or continuously.

For example if we wanted to write a pattern of data to RAM and then read it back to verify it on our sample UUT we would use the following commands,

```
TABLe:FIELd:WORD EXPECT,FLD1,1,#h4000    Store hex 4000 to FLD1 of EXPECT
TABLe:FIELd:WORD RESPONSE,FLD1,1,16384   Store hex 4000 to FLD1 of RESPONSE
TABLe:FIELd:FILL EXPECT,FLD1,INCR,1,2    Generate addresses of RAM in EXPECT.
TABLe:FIELd:FILL RESPONSE,FLD1,INCR,1,2  Generate addresses of RAM in RESPONSE.
TABLe:FIELd:FILL EXPECT,FLD2,RAMP,1      Generate ramp pattern for data.
EXECute:SEQuence WRITE_MEM,EXPECT,1,READ_MEM,RESPONSE,1
```

The results will be stored in FLD2 of RESPONSE and should match the contents in FLD2 of EXPECT.

4.4.5 Programming the PIO/OUT Channels

The BE-64 contains 24 channels of static I/O. The 24 channels are segmented into three eight bit fields called PIO1, PIO2 and OUT. PIO1 and PIO2 can be configured to be either input or output. OUT can be configured to be a counter or output. These 24 channels are used for setting/testing static UUT signal levels such as reset or power good.

The format of the command to configure the desired field is,

```
OUTPut:FIELd:MODE <field_name>,<mode>
```

The <mode> for PIO1 and PIO2 can be output (OUTPut), output enabled (OENabled), input (INPut) or input strobed (ISTRobed). The enable and strobe signals must be externally supplied by the UUT.

The <mode> settings for the OUT field are output (OUTPut), output enabled (OENabled) or counter (COUNt). The enable and count controls must be externally supplied.

For example if our sample UUT was modified such that a UUT enable and power good signal is required as shown in figure 4-22, we could connect PGOOD and UUTEN to PIO1. The following commands program PIO1 to simulate PGOOD and UUTEN,

```
OUTPut:FIELd:MODE PIO1,OUTPut
EXECute:FIELd PIO1,#h0          Sets PGOOD and UUTEN low
EXECute:FIELd PIO1,#h3          Sets PGOOD and UUTEN high.
```

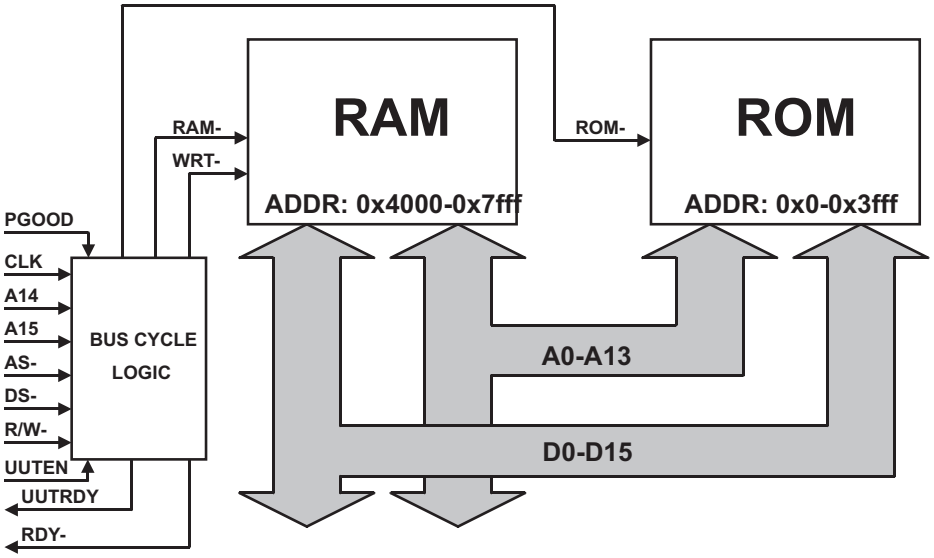


FIGURE 4-21 UUT PIO/BEN EXAMPLE

Additionally if we wanted to monitor the UUTRDY signal we could use PIO2 as input and strobe it with the CLK signal. The following commands program PIO2 as input strobed.

```
OUTPut:FIELd:MODE PIO2,ISTRobed
EXECute:FIELd? PIO2             Returns the last strobed data.
```

4.4.6 Programming the Byte Enable (BEN) Memory

The Byte Enable Memory on the BE-64 is four output channels (BE0 - BE3) that are algorithmically programmed based on the current byte enable code setting and the logic value of Field One channel zero and one (FLD1-0, FLD1-1). The output channels (BE0 - BE3) are internally enabled using TSOUT8, low - enabled. The BE-64 has three byte enable codes, referenced as BYTE, WORD and LONGword. The Byte Enable Outputs are used to simulate bus structured interfaces that require Data Bus Enable signals such as the VME bus (LONGWORD, DS0, DS1) and the 80486 (BE0-BE3).

The format of the command to program the BEN memory is,

```
OUTPut:BENable:DATA <ben_code>,<data0>,<data1>,<data2>,<data3>
```

The <ben_code> selects one of the three settings, BYTE, WORD or LONGword. The four data values represent the output levels and field 2 data mask for each logic state of FLD1-0 and FLD1-1 as described below.

Data format:

Bit 0	BE0 level, 1 = low.
Bit 1	BE1 level, 1 = low
Bit 2	BE2 level, 1 = low.
Bit 3	BE3 level, 1 = low
Bit 4	FLD2 byte1 mask, 1 = enabled.
Bit 5	FLD2 byte2 mask, 1 = enabled.
Bit 6	FLD2 byte3 mask, 1 = enabled.
Bit 7	FLD2 byte4 mask, 1 = enabled.

For example if our sample UUT's RAM array were connected to the bus via eight bit registers controlled by two separate data enable signals, as illustrated in figure 4-23, the byte enable memory could be used to simulate DEN0 and DEN1.

The first step in simulating the data enable signals is to program the Byte Enable Codes. For our example above we will only need to program the BYTE and WORD codes. The following commands program the BYTE and WORD codes to simulate the DEN0 and DEN1 UUT signals,

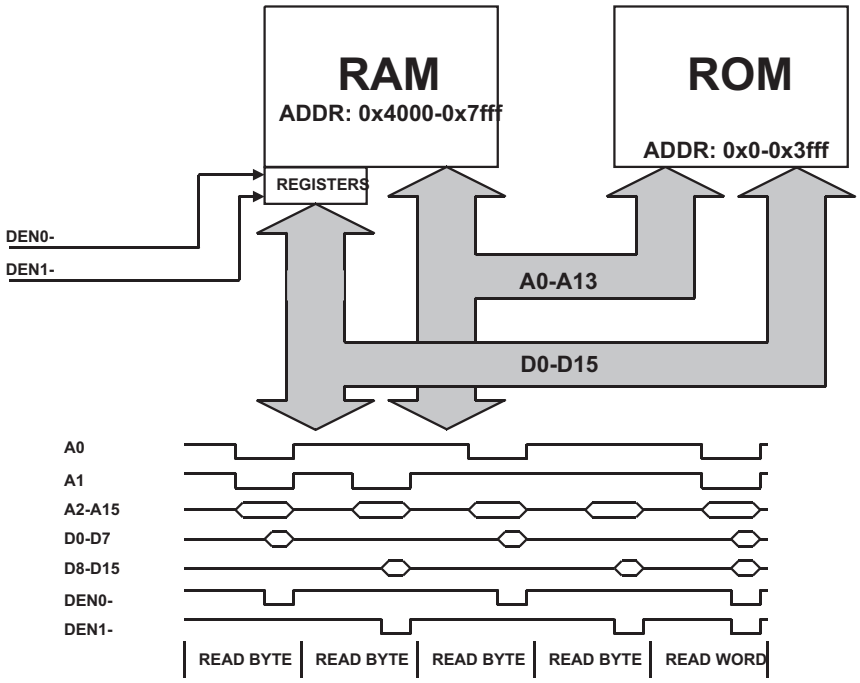


FIGURE 4-22 BEN EXAMPLE

```
OUTPut:BENable:DATA BYTE,#h11,#h22,#h11,#h22
OUTPut:BENable:DATA WORD,#h33,#h33,#h33,#h33
```

The preceding commands assume the DEN0- is connected to BE0 and DEN1- is connected to BE1.

The timing sets must be edited to set TSOUT8 low to enable BE0 and BE1

To link and enable a byte enable code to a table the following commands would be used,

```
OUTput:BENable:STATe ON
TABLe:BENable <table_name>,<ben_code>
```

4.4.7 Programming SYNC

SYNC-T, as well as SYNC (front panel BNC), provide a programmable active low sync pulse. The sync position (the particular word in memory) is defined by writing to the sync pulse memory. The following command defines word 12 of PATT1 as the sync position:

```
OUTPut:SYNC:POSition PATT1,12
```

The sync pulse must be enabled with the following command:

```
OUTPut:SYNC ON
```

See section 4.4.1.3.2 for programming the timing cells. TRIG must be programmed low in all the cells where the sync pulse is to occur. To ensure a single pulse, TRIG should never be low in the first cell and high in the last cell.

If the sequence is programmed for multiple table loops, then the sync pulse could be programmed to occur during a particular loop. To define a sync pulse to occur during the third loop, the following command is used:

```
OUTPut:SYNC:LOOP 3
```

4.5 BE-64 Status/Event Reporting Structures

SCPI requires the status mechanism described in the IEEE 488.2 and is described in the following sections. Figure 4-24 illustrates the status reporting registers used by the BE-64.

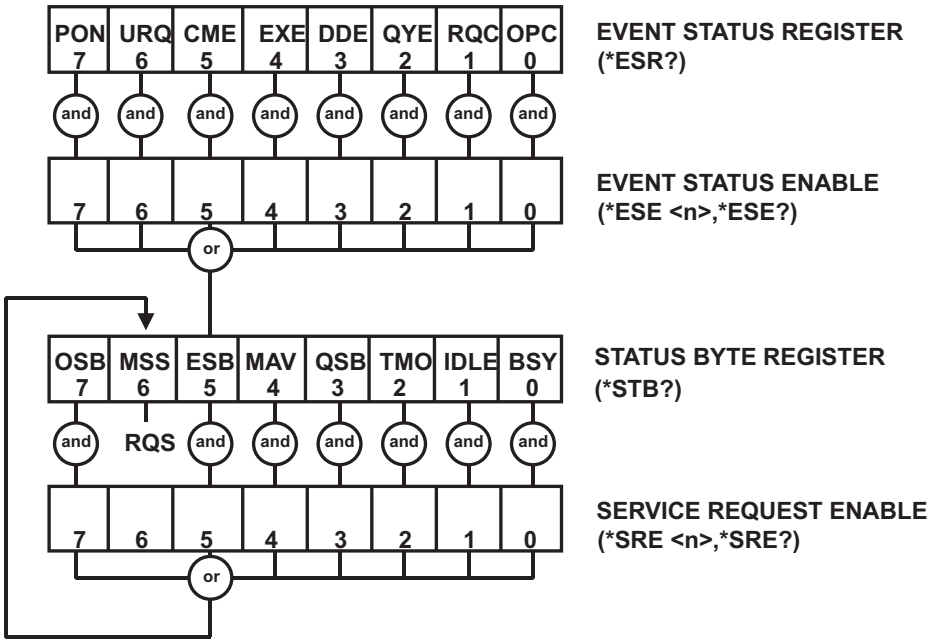


FIGURE 4-23 STATUS REPORTING REGISTERS

Event Status Register bit definitions:

PON	Power on, always zero.
URQ	User request, always zero.
CME	Command Error, SCPI syntax or parameter error.
EXE	Execution Error, Valid SCPI command could not execute.
DDE	Device Dependant Error,
QYE	Query Error, BE-64 query received before previous query was read.
RQC	Request Control, always zero.
OPC	Operation Complete, Set to one when either *OPC or *OPC? executed.

Status Byte Register bit definitions:

OSB	Operation Summary Bit, always zero.
MSS	Master Status Summary, One or more status register bits set.
ESB	Event Summary Bit, One or more event register bits set.
MAV	Message Available, Query response in the output buffer.
QSB	Questionable Summary Bit, Always zero.
IDLE	BE-64 idle state running.
TMO	BE-64 Cycle Timeout occurred.
BSY	BE-64 sequence state running.

All the bits in the Event Status Register are cleared when read via the *ESR? command query.

The MSS bit can be used to signal an event via a VXI interrupt. Selected bits can be enabled or disabled from generating an event by programming the enable registers as shown in figure 4-24. For example the following commands would program the BE-64 to generate an event/interrupt when either an execution or command error occurs.

```
*ESE #H60          Enable EXE and CME bits.
*SRE #H20          Enable the ESB bit.
```

There are two events that can generate an interrupt from the BE-64, request true and request false. Request true occurs when the MSS bit goes from low to high. Request false occurs when the MSS bit goes from high to low. The power on state of the BE-64 has all the events enabled. Selective event generation can be disabled using the word serial Control Event command. All events can be disabled using the word serial Asynchronous Mode Control command

The OSB and QSB bits in the status register are mandated by SCPI but not used by the BE-64. These bits are always zero.

5 COMMAND DESCRIPTION

5.1 INTRODUCTION

This section describes the remote command language of the BE-64. It begins with a overview of the SCPI (Standard Commands for Programmable Instruments) remote command language conventions, followed by the command descriptions.

5.2 PARAMETER LIST CONVENTIONS

SCPI formats (version 1991.0) are used in this instrument to provide a general purpose programming structure.

5.2.1 Parameter List Definition

The parameter list defines the settings for the specified command. Some commands may not require any parameters (i.e. TABLe:DELete:ALL) whereas other commands may have several parameters.

5.2.2 Parameter List Symbols

The following symbols are used in the parameter list definitions:

Comma ,	Parameter separator: If a command has more than one parameter then each parameter is separated by a comma.
Square Brackets [ ]	Optional parameter: The square brackets indicate an optional parameter entry.
Curly Brackets { }	Repeating parameter: The curly brackets represent possible repeating parameter(s).
Parentheses ( )	Parameter option: The parentheses will enclose a list of parameter choices separated by vertical lines.
Vertical Line |	See Parentheses above.

5.2.3 Parameter List Keywords

The following list describes the parameter list keywords and their restrictions where applicable:

<table>	User defined alphanumeric character name for a table. First character must be an alpha character. Maximum length is ten characters. Underscore character is allowed.
<timing_set>	User defined alphanumeric character name for a timing set memory. First character must be an alpha character. Maximum length is ten characters. Underscore character is allowed.
<field>	Pre-defined BE-64 field names; (FLD<n> | PIO<n> | OUT). Not all pre-defined field names are allowable for every command.
<byte_enab>	Pre-defined byte enable mode names; (BYTE | WORD | LONGword).
<pattern>	Pre-defined BE-64 table fill patterns; (COMPlement | INCRement | RAMP | RANDom | REPeat | ROTate | TOGGle).
<block>	Represents Definite Length Arbitrary Block Data as described by the IEEE 488.2 standard. The <block> parameter is used to transfer 8-bit binary data and has the following format: #<n><size><byte data> Where <n> is a single number from 1 to 9 and defines how many digits are in <size>. The <size> is the number of bytes in <byte data> . Each <byte data> element is formatted as eight bit binary bytes. EXAMPLE #210byte1byte2byte3 . . .byte9byte10
<boolean>	The <boolean> parameter is used as a shorthand for ON | OFF | 1 | 0. ON corresponds to 1 and OFF corresponds to 0. Any non zero numeric value will be interpreted as 1.

<others>	The <others> parameter represent any other name that may be given to a parameter to enhance its understanding.

5.2.3.1 Parameter Types Keywords

<ASCII>	Represents entries composed of Talon's pre-defined commands.
<numeric>	Represents a numeric entry. Numeric values can be entered as decimal, hexadecimal (#H), octal (#Q) or binary (#B). Examples: Decimal 256, 69, 42 Hexadecimal #H100, #H45, #H2A Octal #Q400, #Q105, #Q52
<name>	User defined alphanumeric character name.

5.2.4 Events and Queries

All commands, unless otherwise noted, have an event form and a query form.

The event form of a command programs a setting or sets a mode, i.e., performs an event. For example the command "TABLE:WORD DATA,1,0,0" programs the first word of the table named DATA to zero.

The query form returns the associated events current setting or value. A query is a command header with a question mark appended. For example the query "TABLE:WORD? DATA,1" returns the contents of word 1 from the table named DATA.

All ASCII numeric results are returned in decimal.

5.3 CALCulate SUBSYSTEM

The CALCulate subsystem is used to perform post acquisition table analysis.

KEYWORD	PARAMETER FORM
CALCulate	
:CRC?	<table>,<field>,<seed>[,<mask>]
:CRC32?	<table>,<field>,<seed>[,<mask>]
:TCOMpare?	<record_table>,<expect_table>,<field>[,<mask>]

5.3.1 :CRC?

Perform a CRC polynomial on the specified memory.

SYNTAX

CALCulate:CRC? <table>,<field>,<seed>[,<mask>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See section 5.2	The table name identifies the memory locations to perform the CRC on.	NA
<field>	Ascii	FLD1	Perform the CRC on FLD1 memory.	
		FLD2	Perform the CRC on FLD2 memory.	
<seed>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	The seed is used to accumulate multiple CRC results into a single CRC.	
[<Mask>]	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	The mask is used to disable specific channels in the field group from the CRC calculation.	$FFFFFFFF_h$

QUERY RESPONSE

<crc>

Query Data	Type	Value	Description
<crc>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Calculated CRC for the response data specified by <table> and <field>.

COMMENTS

1. A zero in each bit position of the <mask> disables the corresponding bit of the field group.
   For example
   <mask> = #h000000FF.
   Causes a CRC to be generated on channels 0 through 7, channels 8 through 31 are masked off.
2. A “Parameter error” will be generated if <table> is not defined.
3. Table data is only available when the BE64 is idle or reset.

5.3.2 :CRC32?

Perform a CRC32 polynomial on the specified memory.

SYNTAX

CALCulate:CRC32? <table>,<field>,<seed>[,<mask>]

PARAMETER LIST

Parameter	Parameter Type	Values	Description	Default
<table>	Name	User-defined. See section 5.2	The table name identifies the memory locations to perform the CRC on.	NA
<field>	ASCII	FLD1	Perform the CRC on FLD1 memory.	
		FLD2	Perform the CRC on FLD2 memory.	
<seed>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	The seed is used to accumulate multiple CRC results into a single CRC.	
[<Mask>]	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	The mask is used to disable specific channels in the field group from the CRC calculation.	$FFFFFFFF_h$

QUERY RESPONSE

<crc>

Query Data	Type	Value	Description
<crc>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Calculated CRC for the response data specified by <table> and <field>.

COMMENTS

1. A zero in each bit position of the <mask> disables the corresponding bit of the field group.
   For example:
   <mask> = #h0000FF0F.
   Causes a CRC to be generated on channels 0 through 3 and 8 through 15, channels 4 through 7 and 16 through 31 are masked off.
2. A “Parameter error” will be generated if <table> is not defined.
3. Table data is only available when the BE64 is idle or reset.

5.3.3 :TCOMpare?

Execute a table compare function on the tables specified.

SYNTAX

CALCulate:TCOMpare <record_table>,<expect_table>,<field>[,<numeric>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<record_table>	Name	User-defined. See section 5.2	This table contains the data to be compared.	NA
<expect_table>	Name	User-defined. See section 5.2	This table contains the expected data that will be compared to.	
<field>	ASCII	FLD1	Perform the CRC on FLD1 memory.	
		FLD2	Perform the CRC on FLD2 memory.	
[<Mask>]	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	The mask is used to disable specific channels in the field group from the CRC calculation.	$FFFFFFFF_h$

QUERY RESPONSE

<failures>,<channels>,<first_fail>,<stuck>

Query Data	Type	Value	Description
<failures>	Numeric	0-32,768 (0_h-$7FFF_h$)	Number of table locations that did not match.
<channels>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Bit representation of the failing channels where bit 0 high indicates that channel 0 failed and bit 31 high indictes channel 31 failed.
<first_fail>	Numeric	0-32,768 (0_h-$7FFF_h$)	Table index of the first failure.
<stuck>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Bit representation of the channels that are either always high or always low. Bit 0 high indicates that channel 0 is stuck high/low and bit 31 high indicates channel 31 is stuck high/low.

COMMENTS

1. A zero in each bit position of the <mask> disables the corresponding bit of the field group.
   For example:
   <mask> = #hFFFF.
   Causes channels 16 through 31 to be excluded from the table comparison.
2. A “Parameter error” will be generated if either table is not defined.
3. Both tables must be the same size.
4. Table data is only available when the BE64 is idle or reset.

EXAMPLE

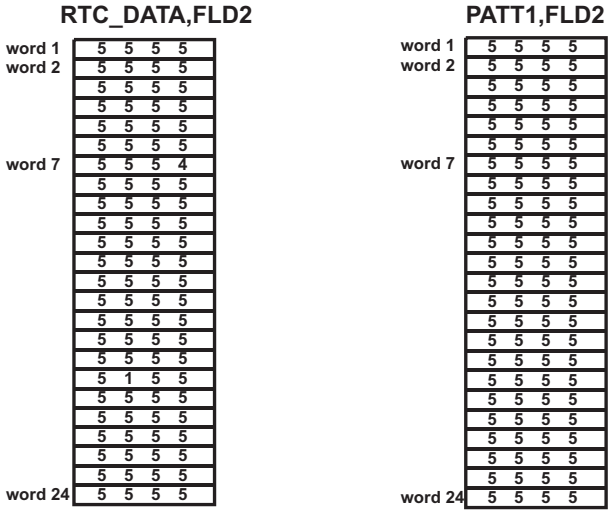


FIGURE 5-1 CALCulate:TCOMpare? Example

```
CALCULATE:TCOM? RTC_DATA,PATT1,FLD2,#HFFFF
```

{Compares FLD2 of table RTC_DATA to FLD2 of table PATT1.}
{returns “2,260,7,65275" (2,104,7, FEFB) hex.}
Indicating there has been 2 failures at Channels 2 and 8 (”0x104”), starting at word 7, and verifying stuck Channels (high/low) are 2 and 8.

5.4 EXECute SUBSYSTEM

The EXECute subsystem allows the operator to control and enable the BE-64 cycle, sequence, PIO, OUT and function execution.

KEYWORD	PARAMETER FORM	NOTES
EXECute		
[:TIMing]	<timing_set>,(<table> | <fld1_data>,<fld2_data>,<byte_enab>)	
:SEQuence	[<timing_set>,<table>,<loop> {,<timing_set>,<table>,<loop>}]	event only
:FUNCtion		
:BUS	[<fld1_mask>,<fld2_mask>]	
:BRAM	<start>,<end>,<pattern>,<page_size>,<delay>,<byte_enab>	
:EEPRom	<start>,<end>,<pattern>,<page_size>,<delay>,<byte_enab>	
:FLASh	<start>,<end>,<pattern>,<write_delay>,<erase_delay>,<byte_enab>	
:ROM	<start>,<end>[,<byte_enab>]	
:RAM	<start>,<end>,<pattern>[,<byte_enab>]	
:WIDTh	<byte_enab>	
:FIELd	<field>[,<data>]	
:MODE	<mode>[,<loop>]	event only

5.4.1 [:TIMing]

The TIMing command will execute the named timing cycle for each item in the data list.

SYNTAX

EXECute[:TIMing] <timing_set>,(<table>|<fld1_data>,<fld2_data>,<byte_enab>)

EXECute:TIMing? <timing_set>,(<table>|<fld1_data>,<fld2_data>,<byte_enab>)

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. See section 5.2	Identifies the timing definition to be used for this execution.	NA
<table>	Name	User-defined. See section 5.2	Identifies the memory location to be used for this execution.	
<fld1_data>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Specifies the FLD1 data for single word execution.	
<fld2_data>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Specifies the FLD2 data for single word execution.	
<byte_enab>	ASCII	BYTE	Specifies the byte enable setting for single word execution.	
		WORD		
		LONG		

QUERY RESPONSE

<fld1_data>,<fld2_data>

Query Data	Type	Value	Description
<fld1_data>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Single word execution returns the contents of the FLD1 channels. Table execution returns the FLD1 CRC of the specified table.
<fld2_data>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Single word execution returns the contents of the FLD2 channels. Table execution returns the FLD2 CRC of the specified table.

COMMENTS

1. If the idle sequence is not running then the BE-64 initiates it. Upon detection of the last cell of the idle timing set, the BE-64 steps to the selected timing set without additional timing set clocks (zero delay).
2. The single word execution FLD1 and FLD2 values are programmed into word 32767 of the table memory.
3. If the parameters are omitted, then the last “EXECute:TIMing” command will be repeated.

EXAMPLES

```
EXECute:TIM READ_MEM,PATT1
```

{Execute the READ_MEM timing cycle for every word in table PATT1.}

```
EXEC WRITE_MEM,#H1000,#H55,BYTE
```

{Execute a WRITE_MEM cycle using hex 1000 as FLD1 and hex 55 as FLD2, byte aligned.}

```
EXEC:TIM? READ_MEM,#H1000,0,BYTE
```

{Execute a READ_MEM cycle using hex 1000 as FLD1, Returns 4096,85.}

5.4.2 :SEQuence

The SEQuence command allows the operator to specify a list of timing sets and tables to execute in sequential order.

SYNTAX

EXECute:SEQuence <timing_set>,<table>,<loop>{,<timing_set>,...}

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. See section 5.2	Identifies the timing definition to be used for this execution.	NA
<table>	Name	User-defined. See section 5.2	Identifies the memory location to be used for this execution.	
<loop>	Numeric	0-65535 (0_h-FFFF$_h$)	Represents a loop count for each sequence entry. Each sequence entry can be programmed to execute a specified number of loops or execute continuously (0).	

COMMENTS

1. If the idle sequence is not running then the BE-64 initiates it. Upon detection of the last cell of the idle timing set, the BE-64 steps to the selected timing set without additional timing set clocks (zero delay).
2. Up to sixteen “<timing_set>,<table>,<loop>” can be specified.
3. If the parameters are omitted, then the last “EXECute:SEQuence” command will be repeated.

EXAMPLE

```
EXEC:SEQ WRITE_MEM,PATT1,1,READ_MEM,PATT2,10
```

{Execute WRITE_MEM with each word of PATT1, then READ_MEM with PATT2 ten times.}

5.4.3 :FUNCtion

The FUNCtion subtree allows the operator to select from one of three built-in tests.

5.4.3.1 :BUS

The TIMing command will execute the named timing cycle for each item in the data list.

SYNTAX

EXECute:FUNCtion:BUS? [<fld1_mask>,<fld2_mask>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<fld1_mask>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Specifies the FLD1 mask.	0
<fld2_mask>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Specifies the FLD2 mask.	

QUERY RESPONSE

<fld1_result>,<fld2_result>

Query Data	Type	Value	Description
<fld1_result>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	FLD1 test result, Bit 0 high indicates channel 0 fail, Bit 31 high indites channel 31 fail.
<fld2_result>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	FLD2 test result, Bit 0 high indicates channel 0 fail, Bit 31 high indites channel 31 fail.

COMMENTS

1. If the idle sequence is not running then the BE-64 initiates it. Upon detection of the last cell of the idle timing set, the BE-64 steps to the selected timing set without additional timing set clocks (zero delay).
2. The FLD1 and FLD2 mask allows error bits to be ignored or to target testing specific bits.

3. The bus test will test both the FLD1 and FLD2 channel signals to make sure they are not shorted or tied in common.
4. The bus test algorithm executes the BUS_TEST timing cycle.
5. The BUS_TEST timing cycle must be greater than two cells and should be setup so that both fields are set to output. During the time when the fields are enabled and prior to the last two cells, strobe the input buffers. The input buffer will be copied to the field memory and compared to expected data.
6. One table word location at the top of table memory (word 32767) will be used for the BUS_TEST sequence.

EXAMPLE

```
EXEC:FUNCtion:BUS?
```

{Execute a bus test function and place the results in the output buffer.}

5.4.3.2 EXECute:FUNCtion:BRAM?

This command performs a Block RAM test pattern using a block mode with a programmable delay between blocks.

SYNTAX

EXECute:FUNCtion:BRAM? <start>,<end>,<pattern>,<page_size>,<delay>,<byte_enab>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<start>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Starting address of the block ram to be tested.	NA
<end>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	End address of the block ram to be tested.	
<pattern>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Data that will be written to the block ram.	
<page_size>	Numeric	1-16384	Page size of the block ram.	
<delay>	Numeric	1-65535	Delay after write sequence.	
<byte_enab>	ASCII	BYTE	Byte enable set to “BYTE”, increment = 1.	
		WORD	Byte enable set to “WORD”, increment = 2.	
		LONG	Byte enable set to “LONG”, increment = 4.	

QUERY RESPONSE

<fail_addr>,<expected>,<actual>

Query Data	Type	Value	Description
<fail_addr>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Address of failed transfer.
<expected>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Expected data.
<actual>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Actual data.

COMMENTS

1. 0,0,0 will be returned if function passes.
2. Required timing sets:
   “WRITE_MEM” Used to write data.
   “READ_MEM” Used to read data.
   “DELAY1US” Used to delay after write block.
3. Generated tables, deleted after function executes:
   “TABLE IN” Created to record read results, size = <page_size>
   “TABLE OUT” Created to write page data, size = <page_size>
   “DUMMY”
   Dummy table for delay, size = 1.
4. The following describes the BRAM algorithm:
   A: Write background data

```
For ADDR = <start> to <end> step <page>
    Program “TABLE OUT” with ADDR(increment) and <pattern>
    Execute sequence
        WRITE_MEM,TABLE OUT,1,DELAY1US,DUMMY,<delay>
Endfor
```

B: Read verify/write compliment.

```
For ADDR = <start> to <end> step <page>
    Format “TABLE IN” with ADDR(increment)
    Format “TABLE OUT” with ADDR(increment) and compliment <pattern>
    Execute sequence
        READ_MEM,TABLE IN,1,WRITE_MEM,TABLE OUT,1,DELAY1US,DUMMY,<delay>
    Verify “TABLE IN”
    If error stop
Endfor
```

C: Read verify

```
For ADDR = <start> to <end> step <page>
    Format “TABLE IN” with ADDR(increment)
    Execute sequence
        READ_MEM,TABLE IN,1
    Verify “TABLE IN”
    If error stop
Endfor
```

5.4.3.3 EXECute:FUNCtion:EEPRom?

This command is an alias of the “EXECute:FUNCtion:BRAM?” command.

5.4.3.4 EXECute:FUNCtion:FLASh?

This command performs a RAM test pattern using a flash mode.

SYNTAX

EXECute:FUNCtion:FLASh? <start>,<end>,<data>,<write_delay>,<erase_delay>,<byte_enab>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<start>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Starting address of the flash ram to be tested.	NA
<end>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	End address of the flash ram to be tested.	
<data>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Data that will be written to the flash ram.	
<write_delay>	Numeric	1-65535	Delay after write sequence.	
<erase_delay>	Numeric	1-65535	Delay after erase sequence.	
<byte_enab>	ASCII	BYTE	Byte enable set to “BYTE”, increment = 1.	
		WORD	Byte enable set to “WORD”, increment = 2.	
		LONG	Byte enable set to “LONG”, increment = 4.	

QUERY RESPONSE

<fail_addr>,<expected>,<actual>

Query Data	Type	Value	Description
<fail_addr>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Address of failed transfer.
<expected>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Expected data.
<actual>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Actual data.

COMMENTS

1. 0,0,0 will be returned if function passes.
2. Required timing sets:
   “WRITE_MEM” Used to write data.
   “READ_MEM” Used to read data.
   “DELAY1US” Used to delay after write sequence.
   “DELAY”
   Used to delay after erase sequence.
3. Required tables:
   “WRITE_SEQ” Holds the address and data to enable a flash write command.
   “ERASE_SEQ” Holds the address and data to enable a bulk erase command.
4. Generated tables, deleted after function executes:
   “DUMMY”
   Dummy table for delay, size = 1.

"DATA"
Read/write data, size = 1.

5. The following describes the BRAM algorithm:

A: Erase flash

```
Execute sequence
    WRITE_MEM,ERASE_SEQ,1,DELAY,DUMMY,<erase_delay>
```

B: Write background data

```
For ADDR = <start> to <end> step <byte_enab>
    Program "DATA" FLD1 = ADDR, FLD2 = <pattern>
    Execute sequence
        WRITE_MEM,WRITE_SEQ,1,WRITE_MEM,DATA,1,DELAY1US,DUMMY,<write_delay>
Endfor
```

C: Read verify

```
For ADDR = <start> to <end> step <byte_enab>
    Execute sequence
        READ_MEM,DATA,1
    Verify DATA
    If error stop
Endfor
```

D: Erase flash

```
Execute sequence
    WRITE_MEM,ERASE_SEQ,1,DELAY,DUMMY,<erase_delay>
```

E: Write compliment data

```
For ADDR = <start> to <end> step <byte_enab>
    Program "DATA" FLD1 = ADDR, FLD2 = complimented <pattern>
    Execute sequence
        WRITE_MEM,WRITE_SEQ,1,WRITE_MEM,DATA,1,DELAY1US,DUMMY,<write_delay>
```

F: Read verify

```
For ADDR = <start> to <end> step <byte_enab>
    Program "DATA" FLD1 = ADDR.
    Execute sequence
        READ_MEM,DATA,1
    Verify DATA
    If error stop
Endfor
```

5.4.3.5 :ROM

The ROM command will execute the built-in rom test using FLD1 as address and FLD2 as data.

SYNTAX

EXECute:FUNCtion:ROM <start>,<end>[,<byte_enab>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<start>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Starting address of the ROM to be tested.	NA
<end>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Ending address of the ROM to be tested.	
<byte_enab>	ASCII	BYTE	Byte enable set to "BYTE", increment = 1.	
		WORD	Byte enable set to "WORD", increment = 2.	
		LONG	Byte enable set to "LONG", increment = 4.	

QUERY RESPONSE

<checksum>

Query Data	Type	Value	Description
<checksum>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	The 32 bit checksum of the FLD2 channels.

COMMENTS

1. The ROM test executes the READ_MEM timing set sequence at each FLD1 address in the range specified by the first two parameters and forms a 32 bit checksum of the FLD2 data.
2. One table word location at the top of table memory (word 32767) will be used for the READ_MEM sequence.

3. The <byte_enab> parameter defines a mask and an increment for the FLD1 addresses, i.e., BYTE(increment=1), WORD(increment=2), LONGword(increment=4). The mask can be specified differently from the increment by using the “EXECute:FUNCtion:WIDTh” command.
4. The following describes the rom test algorithm:

   A: Initialze variables

```
address = <start>
if (<byte_enab> = BYTE)
    increment = 1
else if <byte_enab> = WORD
    increment = 2
else if <byte_enab> = LONG
    increment = 4
mask = (byte(0xFF), word(0xFFFF), long(0xFFFFFFFF)
```

   B: Read data

```
while address less or equal to <end>
    FLD1 = address
    Execute timing
        READ_MEM,FLD1,FLD2
    checksum = checksum + (FLD2 and mask)
    address = address + increment
Endwhile
```

EXAMPLE

```
EXEC:FUNC:ROM #HF8000,#HFFFFF,BYTE
```

{Execute a ROM test from address F8000 to FFFFF hex.}

5.4.3.6 :RAM

The RAM command will execute the built-in ram test using FLD1 as address and FLD2 as data

SYNTAX

EXECute:FUNCtion:RAM <start>,<end>,<data>[,<byte_enab>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<start>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Selects FLD1 starting address.	NA
<end>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Selects FLD1 ending address.	
<data>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	FLD2 data.	
<byte_enab>	Numeric	BYTE	Byte enable set to “BYTE”, increment = 1.	
		WORD	Byte enable set to “WORD”, increment = 2.	
		LONG	Byte enable set to “LONG”, increment = 4.	

QUERY RESPONSE

<test-results>

Query Data	Type	Value	Description
<test-results>	Numeric	NA	Returns a 0 if test passes otherwise returns a non-zero or error.

COMMENTS

1. The RAM test executes the WRITE_MEM/READ_MEM timing set sequence at each FLD1 address in the range specified by <start> and <end> parameters using the <data> parameter as FLD2 data.
2. Two table word locations at the top of table memory (word 32766 and 32767) will be used for the WRITE_MEM/READ_MEM sequence.
3. The <byte_enab> parameter defines a mask and an increment for the FLD1 addresses, i.e., BYTE(increment=1), WORD(increment=2), LONGword(increment=4).
4. The following describes the ram test algorithm:

   A. Initialize Variables

```
start_addr = 1st <numeric> parameter.
end_addr = 2nd <numeric> parameter.
data = 3rd <numeric> parameter
if (<byte_enab> = BYTE)
```

```
        increment = 1
else if <byte_enab> = WORD
        increment = 2
else if <byte_enab> = LONG
        increment = 4
```

B. “PASS 1 write background data”

```
while start_addr is less or equal to end_addr
        FLD1 = start_addr
        FLD2 = data
        exec WRITE_MEM sequence
        start_addr = start_addr + increment
Endwhile
reset start_addr = 1st <numeric> parameter.
```

C. “PASS 2 verify background, write compliment data”

```
while start_addr is less or equal to end_addr
        FLD1 = start_addr
        FLD2 = complemented data
        execute READ_MEM, WRITE_MEM sequence
        verify FLD2 = data
        start_addr = start_addr + increment
Endwhile
reset start_addr = 1st <numeric> parameter
```

D. “PASS 3 verify compliment data”

```
while start_addr is less or equal to end_addr
        FLD1 = start_addr
        exec READ_MEM sequence
        verify FLD2 = complemented data
        start_addr = start_addr + increment
Endwhile
```

The query form returns the results as two numeric values. The first numeric value represents the failing address and the second is the logical exclusive OR of the expected data and the actual data.

EXAMPLE

```
EXEC:FUNC:RAM? #H0,#H9FFFF,#H5555,WORD
```

{Execute a RAM test with hex 5555 data from 0 to 9FFFF hex, returns 0,0 if test passes}

5.4.4 :WIDTh

This command allows the user to select a different byte enable size from the size specified in the function subsystem commands (ROM, RAM, EEPROM, FLASH and BRAM).

SYNTAX

EXECute:FUNCtion:WIDTh <byte_enab>

EXECute:FUNCtion:WIDTh?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<byte_enab>	ASCII	BYTE	Data bus set to BYTE mode.	NA
		WORD	Data bus set to WORD mode.	
		LONG	Data bus set to LONG mode.	

QUERY RESPONSE

<byte_enab>

Query Data	Type	Value	Description
<byte_enab>	ASCII	DEFAULT	The byte enable size is specified in the specific command.
		BYTE	Byte enable for the function subsystem set to “BYTE”
		WORD	Byte enable for the function subsystem set to “WORD”
		LONG	Byte enable for the function subsystem set to “LONG”

COMMENTS

1. The <byte_enab> specified in the function commands specify the data bus size as well as the address increment. This command allows mixed addressing, i.e., byte addressing with word data.

5.4.5 :FIELd

The FIELd command allows the operator to set or read the PIO and OUT fields.

SYNTAX

EXECute:FIELd <field>,<data>

EXECute:FIELd? <field>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<field>	ASCII	PIO1	Program/query PIO1	NA
		PIO2	Program/query PIO2	
		OUT	Program/query OUT	
<data>	Numeric	0-255 (0_h-FF_h)	Data that will be programmed in the selected filed	

QUERY RESPONSE

<data>

Query Data	Type	Value	Description
<data>	Numeric	0-255 (0_h-FF_h)	Contents off the selected field.

COMMENTS

None

EXAMPLE

```
EXEC:FIELd PIO1,#H1F
```

{Set the PIO1 field to hex 1F.}

```
EXECUTE:FIEL? PIO2
```

{Returns the contents of PIO2}

5.4.6 :MODE

The FIELd command allows the operator to set or read the PIO and OUT fields.

SYNTAX

EXECute:MODE <mode>[,<loop>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mode>	ASCII	SINGle	The BE-64 will execute subsequent timing or sequence executions once. IDLE state is active.	NA
		CONTinuous	The BE-64 will execute subsequent timing or sequence executions continuously until a SINGle, RESet or STOP mode command is executed. IDLE state is active.	
		RESet	The RESet mode will stop any current timing or sequence execution. IDLE state is not active.	
		STOP	The STOP mode will stop any current timing or sequence execution. IDLE state is active.	
		LOOP	The LOOP mode will execute subsequent timing or sequence executions the specified number of times. IDLE state is active.	
<loop>	Numeric	0-65535 (0_h-$FFFF_h$)	Loop mode count, 0 = continuous	

COMMENTS

None

EXAMPLE

```
EXEC:MODE STOP
```

{Stops all timing cycle activity and enters the idle state.}

```
EXEC:MODE SINGle
```

{Sets the mode to single execution.}

5.5 OUTPut SUBSYSTEM

The OUTPut subsystem controls the output ports of the BE-64.

KEYWORD	PARAMETER FORM	NOTES
OUTPut		
:RESet	<level>, [<pulse_width>]	
:BAENable	<state>	
:TTLTrg<trigger>		
[:STATe]	<state>	
:SOURce	<trigger_signal>	
:FIELd		
:MODE	<field>, <field_mode>	
:BENable		
[:STATe]	<state>	
:DATA	<byte_enab>,<numeric>,<numeric>,<numeric>,<numeric>	
:SYNC		
[:STATe]	<state>	
:POSition	<table>,<sync_position>	
:LOOP	<count>	

5.5.1 :RESet

The RESet command allows the operator to define the level of the UUTRST output signal as well as generate a programmable pulse.

SYNTAX

OUTPut:RESet <level>, [<pulse_width>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<level>	ASCII	HIGH	Generates a high level on the UUTRSToutput signal.	NA
		LOW	Generates a low level on the UUTRST output signal.	
		PULSe	Generates a high or low pulse depending on prior HIGH or LOW setting.	
<pulse_width>	Numeric	0-65535 (0_h-$FFFF_h$)	Sets the pulse width of the UUTRST output signal.	1mS or 122 (0_h-$7A_h$)

COMMENTS

1. The pulse width can be programmed from ~20us to 0.5s in increments of ~8us using the following equation:

   Pulse width= 20uS + <pulse_width>*8uS

EXAMPLE

```
OUTPut:RESet LOW
```

{Sets UUTRES low.}

```
OUTPut:RESet PULSe,6250
```

{Generate a ~50ms high pulse on UUTRST, 20us + 6250 * 8us}

5.5.2 :BAENable

The BAENable (Bus Access Enable) command enables/disables the BUSACK- response generation.

SYNTAX

OUTPut:BAENable <state>

OUTPut:BAENable?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<state>	ASCII/ Numeric	ON/1	Enables UUT bus access	0
		OFF/0	Disables UUT bus access	

QUERY RESPONSE

<state>

Query Data	Type	Value	Description
<boolean>	Numeric	1/0	Returns status (enabled/disabled) of the BUSACK- response generation

COMMENTS

1. If the BAENable is set ON/1 the the BE-64 will generate a BUSACK- as described in section 3.4.5.4 of this manual. Setting the BAENable off disables BUSACK- response.

EXAMPLE

```
OUTPut:BAENable ON
```

{Enables BUSACK- response.}

```
OUTPut:BAEN OFF
```

{Disables BUSACK- generation.}

5.5.3 :TTLTrg

The TTLTrg subtree controls the driving of the eight VXI backplane trigger lines and the two front panel trigger lines. A numeric suffix is required to identify the particular VXI trigger line, TTLTRG0 - TTLTRG7. A alpha suffix is required to identify the particular front panel trigger line, TTLTRGA - TTLTRGB.

5.5.3.1 [:STATe]

The STATe command selects whether or not the BE-64 drives the trigger line.

SYNTAX

OUTPut:TTLTrg<trigger>[:STATe] <state>

OUTPut:TTLTrg<trigger>[:STATe] ?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<trigger>	Numeric	0-7	Selects the Trigger Line, TTLTRG0-TTLTRG7	NA
	ASCII	A or B	Selects the Front Panel Trigger, TTLTRGA or TTLTRGA	
<state>	Numeric / ASCII	ON/1	Enables BE-64 to drive the selected trigger.	
		OFF/0	Disables BE-64 to drive the selected trigger.	

QUERY RESPONSE

<state>

Query Data	Type	Value	Description
<state>	Numeric	1/0	Returns the status of selected trigger (enabled/disabled)

COMMENTS

None

EXAMPLE

```
OUTP:TTLT3 ON
```

{Enables BE-64 to drive the VXI TTLTRG3 trigger.}

```
OUTPUT:TTLTRGA:STATE 1
```

{Enables the BE-64 to drive front panel signal.}

```
OUTPUT:TTLTRGA? Or OUTPUT:TTLTRGA:STATE?
```

{Returns the status of TTLTrigger A.}

5.5.3.2 :SOURce

The SOURce command controls which signal will drive the selected trigger..

SYNTAX

OUTPut:TTLTrg<trigger>:SOURce <trigger_signal>

OUTPut:TTLTrg<trigger>:SOURce?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<trigger>	Number	0-7	Selects the Trigger Line, TTLTRG0-TTLTRG7	NA
	ASCII	A or B	Selects the Front Panel Trigger, TTLTRGA or TTLTRGA	
<trigger_signal>	Name	PFLD1	Selects the PROBE FLD1 timing set channel and is programmed from the TIMing subsystem. Valid trigger lines are TTLTRG4 - TTLTRG7 only.	TTLT4
		PFLD2	Selects the PROBE FLD2 timing set channel and is programmed from the TIMing subsystem. Valid trigger lines are TTLTRG4 - TTLTRG7 only.	
		EXT	Selects the EXTernal strobe signal. Valid trigger lines are TTLTRG4 - TTLTRG7 only.	
		TRIG	Selects the TRIGger timing set channel and is programmed from the TIMing subsystem. This signal is internally gated with the sync signal to generate a signal compatible with Talons logic analyzer/probe card. Valid trigger lines are TTLTRG0 - TTLTRG3.	TTLT0
		TTLT0-7	RoutsTTLT0-7 to Front Panel Trigger A or B.	NA

QUERY RESPONSE

<state>

Query Data	Type	Value	Description
<state>	Numeric	1/0	Returns the source of the selected trigger.

COMMENTS

1. The <trigger_signal> must agree with the respective valid trigger lines.

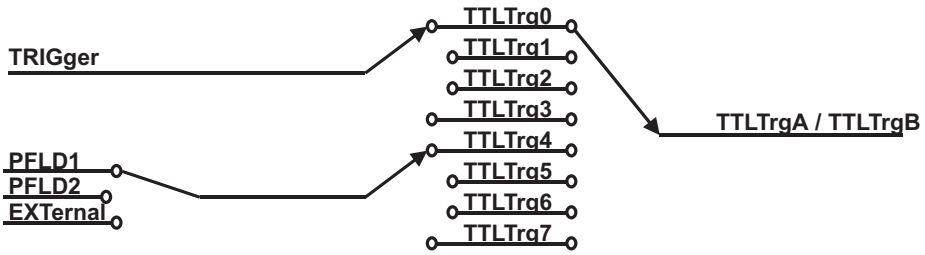


FIGURE 5-2 TTLTRG:SOURce CHOICES

EXAMPLE

```
OUTPUT:TTLTRG4:SOURCE PFLD2
```

{Routes the PFLD2 signal to the VXI TTLTRG4.}

```
OUTP:TTLTB:SOUR  TTLT4
```

{Routes the TTLTRG4 signal to the front panel TTLTRGB

```
OUTP:TTLTB:SOUR?
```

{Returns the trigger source of TTLTB.

5.5.4 :FIELd

The FIELd subtree controls the BE-64 programmed I/O fields.

5.5.4.1 :MODE

The MODE command controls the configuration of the PIO and OUT fields.

SYNTAX

OUTPut:FIELd:MODE <field>,<field_mode>

OUTPut:FIELd:MODE? <field>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<field>	ASCII	PIO1	Program PIO1	INPut
		PIO2	Program PIO2	
		OUT	Program OUT	OUTPut

Parameter	Type	Values	Description	Default
<field_mode>	ASCII	OUTPut	Sets the PIO1, PIO2, and OUT fields to OUTput or OENable.	NA
		OENabled		
		INPut	Sets the PIO1 and PIO2 fields to INPut or ISTRobe(Input Strobe).	
		ISTRobed		
		COUNtet	Sets the OUTput field to COUNter.	

QUERY RESPONSE

<field>

Query Data	Type	Value	Description
<field_mode>	ASCII	OUTPut	Modes pertaining to PIO1, PIO2, and OUT fields.
		OENabled	
		INPut	
		ISTRobed	
		COUNtet	

COMMENTS

1. Power on default for PIO1 and PIO2 is INPut. OUT default is OUTPut with zero loaded.

EXAMPLE

```
OUTP:FIEL:MODE PIO1,ISTROBED
```
{Set PIO1 to input strobed.}
```
OUTPUT:FIELD:MODE OUT,COUNT
```
{Set OUT to counter operation.}
```
OUTPUT:FIELD:MODE?PIO1
```
{Queries the setting for the PIO1 field}

5.5.5 :BENable

The BENable subtree allows the operator to define and enable the byte enable process for the BE0-BE3 output field. The BE0-BE3 levels are programmed according to the enable mode (BYTE, WORD or LONGword) and the two least significant bits of FLD1 for each word.

The byte enable process performs the following actions after an EXECute:TIMing or EXECute:SEQuence command but prior to actual execution:

- For each table word, the first two channels of FLD1 along with the BEN code are used as an index into an array of data values programmed by the OUTput:BENable:DATA command. The byte enable memory is programmed to the least significant nibble of the data value.
- For each table word, the byte enable code BYTE will cause the least significant byte of FLD2 to be copied into the three remaining bytes of FLD2. WORD byte enable code will cause the least significant word of FLD2 to be copied into the most significant word.

Byte enable post processing is only done when data is requested from the BE-64. Post processing involves masking the data in FLD2 according to the byte enable code and the same index used to program the byte enable memory prior to execution. The mask is programmed in the upper nibble of the data value indexed by the BEN code and the two least significant channels of FLD1.

5.5.5.1 [:STATe]

The STATe command selects whether or not the BE-64 programs the byte enable codes prior to emulator execution.

SYNTAX

OUTPut:BENable[:STATe] <state>

OUTPut:BENable[:STATe] ?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<state>	ASCII/Numeric	ON/1	Enables the BYTE processing algorithm	0
		OFF/0	disables the BYTE processing algorithm	

QUERY RESPONSE

<state>

Query Data	Type	Value	Description
<state>	Numeric	1	BYTE processing algorithm is enable
		0	BYTE processing algorithm is disable

COMMENTS

None

EXAMPLE

```
OUTP:BENABLE:STAT OFF
```

{Turns off the byte enable algorithm processing.}

```
OUTP:BEN? or OUTP:BEN:STAT?
```

{Returns the current state of the BYTE processing algorithm}

5.5.5.2 :DATA

The DATA command builds an array that is used to program the byte enable memory as well as mask specific bytes in FLD2 after execution as shown in TABLE 5-1.

SYNTAX

OUTPut:BENable:DATA <byte_enab>,<numeric>,<numeric>,<numeric>,<numeric>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<byte_enab>	ASCII	BYTE	Defines the type of data.	NA
		WORD		
<numeric>	Numeric	0-255 (0_h-FF_h)	defines data mask and the BE0-BE3 levels	

COMMENTS

1. The four <numeric>'s are bytes which define a data mask and the BE0-BE3 levels for each binary state of bit 0 and bit 1 of FLD1 as shown below.

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
31-24	23-16	15-8	7-0	BE3	BE2	BE1	BE0
FLD2 BYTE MASK (1 = ON)				BYTE ENABLE (1 = LOW)			

FIGURE 5-3 BEN DATA FORMAT

EXAMPLE

```
OUTPut:BEN:DATA BYTE,#H11,#H22,#H44,#H88
```

{Causes the mask to move from to MSB and the BEn signal to follow the FLD1 address.}

5.5.6 :SYNC

The SYNC subtree allows the operator to define and enable a sync pulse during a specific word execution.

5.5.6.1 [:STATe]

The STATe command selects whether or not the BE-64 drives the sync pulse output low.

SYNTAX

OUTPut:SYNC[:STATe] <state>

OUTPut:SYNC[:STATe]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<state>	Numeric	ON/1	Enables Sync Drive	0
		ON/0	Disables Sync Drive	

QUERY RESPONSE

<state>

Query Data	Type	Value	Description
<state>	Numeric	1/0	Status of Sync Drive(Enabled/Disabled)

COMMENTS

None

EXAMPLE

```
OUTPUT:SYNC 0
```

{Turns off sync drive.}

```
OUTPUT:SYNC?
```

{Queries status of sync drive.}

5.5.6.2 :POSition

The POSition command allows the operator to define a sync position within the field memory.

SYNTAX

OUTPut:SYNC:POSition <table>,<sync_position>

OUTPut:SYNC:POSition?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See section 5.2	Describes the Name of Table	NA
<sync_position>	Numeric	1-32768	Defines a sync position	

QUERY RESPONSE

<sync_position>

Query Data	Type	Value	Description
<sync_position>	Numeric	1-32768	Sync Position

COMMENTS

None

EXAMPLE

```
OUTPut:SYNC:POS PATT1,12
```

{Causes a pulse at the SYNC BNC whenever word 12 of PATT1 is active on the address and data fields. This assumes TRIG in the timing set is low for the entire timing set.}

5.5.6.3 :LOOP

The LOOP command allows the operator to define a loop count for the selected sync position.

SYNTAX

OUTPut:SYNC:LOOP <count>

OUTPut:SYNC:LOOP?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<count>	Number	1 to 65535	Defines Loop count	1

QUERY RESPONSE

<count>

Query Data	Type	Value	Description
<count>	Numeric	1-65535	Loop Count Value.

COMMENTS

None

EXAMPLE

```
OUTPUT:SYNC:LOOP 12
```

{Generate a pulse on word 12 of table PATT1 during the twelfth loop.}

5.6 STATus SUBSYSTEM

The STATus subsystem contains the SCPI status reporting registers. The BE-64 does not implement the SCPI status reporting registers. The query commands will always return a zero and the enable and preset commands will perform no function.

KEYWORD	PARAMETER FORM	NOTES
STATus		
:OPERation		
[:EVENt]?		query only
:CONDition?		query only
:ENABle	<numeric>	
:QUEStionable		
[:EVENt]?		query only
:CONDition?		query only
:ENABle	<numeric>	
:PRESet		

5.6.1 :OPERation

STATus:OPERation

5.6.1.1 [:EVENt]

STATus[:EVENt]?

5.6.1.2 :CONDition

STATus:CONDition?

5.6.1.3 :ENABle

STATus:ENABle <numeric>

5.6.2 :QUEStionable

STATus:QUEStionable

5.6.2.1 [:EVENt]

STATus[:EVENt]?

5.6.2.2 :CONDition

STATus:CONDition?

5.6.2.3 :ENABle

STATus:ENABle <numeric>

5.6.3 :PRESet

STATus:PRESet

5.7 SYSTem SUBSYSTEM

KEYWORD	PARAMETER FORM	NOTES
SYSTem		
:ERRor?		query only
:VERSion?		query only

5.7.1 :ERRor

SYSTem:ERRor?

The ERRor command returns an error code/message from the error queue. The queue contains an integer in the range of [-32768...32767]. Negative numbers are SCPI defined and positive numbers are BE-64 dependent errors. An error message enclosed in double quotes follows the error code separated by a comma. The error queue is a first in first out list of error messages. If the error queue overflows the last error in the queue is replaced with the following error:

```
-350,"Queue overflow"
```

When all the errors have been read from the queue, (i.e. error queue is empty), the error query will return the following:

```
0,"No error"
```

The following is a list of all the BE-64's errors.:

```
-100, "Command error"     Error occurred traversing command tree.
-102, "Syntax error"      Misspelled Keyword.
-108, "Parameter not allowed"     Invalid parameter for command format.
-109, "Missing parameter" Expected parameter missing.
-160, "Block data error"  Error during DATA command.
-220, "Parameter error"   Invalid parameter for command.
-221, "Settings conflict" Valid command could not execute due to settings
-311, "Memory error"      Attempt to allocate memory failed.
-350, "Queue overflow"    Too many errors.
-400, "Query error"               Query command received before prior query read.
```

5.7.2 :VERSion

The VERSion query returns the SCPI version that the BE-64 complies with.

SYNTAX

SYSTem:VERSion?

QUERY RESPONSE

<version>

Query Data	Type	Value	Description
<version>	<ASCII>	System Version	Returns the system version.

COMMENTS

None

EXAMPLE

```
:SYST:VERS?
```

{Returns 1991.0.}

5.8 TABLE SUBSYSTEM

The TABLe subsystem is used to define and edit tables.

KEYWORD	PARAMETER FORM	NOTES
TABLe		
:DEFine	<table>[,<size> |<table-source>]	
[:DATA]	<table>[,<block>]	
:WORD	<table>,<numeric>[,<field1>,<field2>]	
:FIELd		
:WIDTh	<table>,<field>, <field-size>	
[:DATA]	<table>,<field>[,<block>]	
:WORD	<table>,<field>,<word>,<data>	
:CHANnel	<table>,<field>,<channel>,<channel-data>	
:CEXPand	<table>,<field>,<channel>,<channel-data	
:FILL	<table>,<field>,<pattern>,<numeric1>,<numeric2.>	
:DELete		
[:NAME]	<table>	
:ALL		
:BENable	<table>,<byte_enab>	
:DIRectory?		
:FREE?		

5.8.1 :DEFine

The DEFine command allocates and initializes a new table.

The first parameter is the name of the new table and can be 10 characters. The first character of the name must be an alpha character. The other characters can be alphanumeric or the underscore character (_).

SYNTAX

TABLe:DEFine <table>,(<size> | <table-source>)

TABLe:DEFine? <table>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Name of Table to be created.	NA
<size>	Number	1 - 32768	Defines size of table	
<table-source>	Name	User-defined. See Section 5.2	Name of Table to be copied from.	

QUERY RESPONSE

<table>, <size>, <first address>

Query Data	Type	Value	Description
<table>	Name	User-defined. See Section 5.2	Table Name
<size>,	Numeric	1 - 32768	Size of table
<first address>	Numeric	OFFSET + (1 - 32768) (0_h-8000_h)	Address of first word

COMMENTS

1. Given the range of <size> is from 1 - 32768, the maximum number is decreased by each previous tables size such that the commutative number of words cannot exceed 32767.
2. If the second parameter is a <size>, the table will be allocated with the specified number of words initialized to zero. If the second parameter is an existing <table> then the new table will be a copy of the existing table.
3. The query form of the command returns the name, size and the offset address of the first word within the BE-64's A24 memory. See appendix D for a description of the A24 memory.

EXAMPLE

```
TABLe:DEF PATT1,24
```

{Defines a new table called PATT1 with 24 words. Word data is initialized to zero.}

```
TABL:DEFine RTC_DATA,PATT1
```

{Defines a new table named RTC_DATA and is a copy of table PATT1.}

```
TABL:DEF? PATT1
```

{Returns “PATT1",24,262144.}

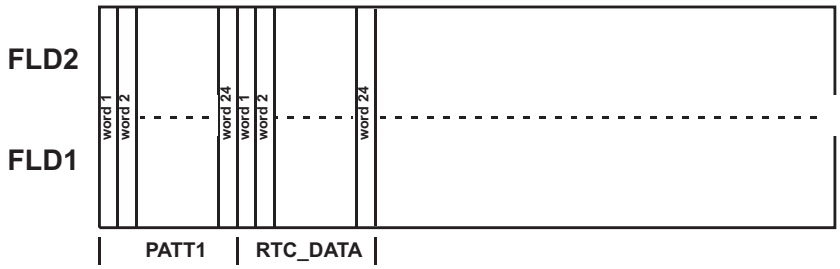


FIGURE 5-4 :DEFine EXAMPLE

5.8.2 [:DATA]

The DATA command sets the contents of both FLD1 and FLD2 field memory to the contents of block as binary, i.e., eight bytes per table word.

SYNTAX

TABLe[:DATA] <table>,<block>

TABLe[:DATA]?<table>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Selects the Table to be Modified	NA
<block>	Numeric	See section 5.2.3.	Values to be written to the selected table in block format.	

QUERY RESPONSE

<block>

Query Data	Type	Value	Description
<block>	Numeric	See section 5.2.3.	Contents of table in block data

COMMENTS

1. For reference, the byte order of the block data is illustrated in the following figure 5-1.

FIGURE 5-5 TABLE DATA BLOCK FORMAT

EXAMPLE

```
TABLe RTC_DATA,#21612345678ABCDEFGH
```

{Sets RTC_DATA word 1 field 1 to hex 31323334 and word 1 field 2 to hex 35363738.}

{Sets RTC_DATA word 2 field 1 to hex 41424344 and word 2 field 2 to hex 45464748.}

```
TABL:DATA? RTC_DATA
```

{Returns: #600001612345678ABCDEFGH}

5.8.3 :WORD

The WORD command allows individual table memory locations to be edited or queried.

SYNTAX

TABLe:WORD <table>,<word>,<field1>,<field2>

TABLe:WORD? <table>,<word>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Selects the table whose word is to be modified	NA
<word>	Number	1 - 32768	Indexes the Word within the table. It is limited by the size of the table	
<field1>	Number	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Value for Field 1	
<field2>	Number	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Value for Field 2	

QUERY RESPONSE

<field1>, <field2>

Query Data	Type	Value	Description
<field1>	Number	0-4,294,967,295	Value of a Field 1 or 2 in decimal
<field2>			

COMMENTS

1. The <word> is the index of the word within the table and the <field1>and <field2> are the FLD1 and FLD2 values respectively.
2. The query form returns the contents of the specified location.

EXAMPLE

```
TABLe:WORD RTC_DATA,12,#H4000,#HFF
```

{Sets word 12 of table RTC_DATA FLD1 to 0x4000 hex and FLD2 to FF hex.}

```
TABLe:WORD? RTC_DATA,12
```

{Returns 16384,255}

5.8.4 :FIELd

The FIELd subtree allows the operator to program the specified table and field.

5.8.4.1 :WIDTh

This command sets the field width for subsequent TABLe:FIELd:FILL and TABLe:FIELd:DATA commands.

The query form returns the width as an ASCII string.

SYNTAX

TABLe:FIELd:WIDTh <table>,<field>, <field-size>

TABLe:FIELd:WIDTh? <table>,<field>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Selects the table whose field size is to be modified	NA
<field>	ASCII	FLD1, FLD2	Selects the Field whose size is to be defined	
<field-size>	ASCII	BYTE, WORD or LONGword	Defines data size to 8, 16 or 32-bit words, respectively.	

QUERY RESPONSE

<field-size>

Query Data	Type	Value	Description
<field-size>	ASCII	BYTE, WORD, LONG	Returns the current size the selected Field.

COMMENTS

None

EXAMPLE

```
TABLe:FIEDL:WIDTh PATT1,FLD1,LONGword
```

{Sets PATT1, FLD1 width to LONGword.}

```
TABLe:FIELD:WIDT PATT1,FLD2,WORD
```

{Sets PATT1, FLD2 width to WORD.}

```
TABL:FIELD:WIDT? PATT1,FLD1
```

{Returns “LONG”.}

5.8.4.2 [:DATA]

The DATA command sets the contents of the destination field memory to <block>.

SYNTAX

TABLe:FIELd[:DATA] <table>,<field>,<block>

TABLe:FIELd[:DATA]? <table>,<field>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2.	Selects the name of the table to write the block data to.	NA
<field>	ASCII	FLD1, FLD2	Selects the field of the selected data.	
<block>	Numeric	See Section 5.2.3.	Data in Block format.	

QUERY RESPONSE

<block>

Query Data	Type	Value	Description
<block>	Numeric	See Section 5.2.3.	Return data in respective field in block format.

COMMENTS

1. The number of bytes per word is specified by the WIDTh command, 4/LONGword, 2/WORD, 1/BYTE. The field memory is aligned to the least significant channel in the case of WORD and BYTE widths.

EXAMPLE

```
TABL:DEF PATT1,2
TABL:FIEL:WIDTH PATT1,FLD2 WORD
TABLe:FIEL PATT1,,FLD2,#1812345678
```

{Sets PATT1 word 1 field 2 to hex 31323334 and word 2 field 2 to hex 35363738.}

```
TABL:FIELD:DATA? PATT1,FLD2
```

{Returns: #600000812345678}

5.8.4.3 :WORD

The WORD command allows individual table field memory values to be edited and queried.

SYNTAX

FIELd:WORD <table>,<field>,<word>,<data>

FIELd:WORD? <table>,<field>,<word>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Selects Table to perform write or read operation.	NA
<field>	ASCII	FLD1, FLD2	Select Table Field.	
<word>	Numeric	1 - 32768	Selects the word number to index to, and it is limited by table size.	
<data>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Data to be written to respective word, it can be in Hex or Decimal format.	

QUERY RESPONSE

<data>

Query Data	Type	Value	Description
<data>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Data stored in respective field word.

COMMENTS

None

EXAMPLE

```
TABL:FIELd:WORD PATT1,FLD1,12,#H80000
```

{Sets FLD1 word 12 in table PATT1 to 0x80000 hex.}

```
TABL:FIEL:WORD? PATT1,FLD1,12
```

{Returns 524288.}

5.8.4.4 :CHANnel

This command programs a single channel of BE-64 field memory without affecting the contents of the other channels.

SYNTAX

TABLe:FIELd:CHANnel <table>,<field>,<channel>,<channel-data>

TABLe:FIELd:CHANnel? <table>,<field>,<channel>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Selects Table to perform Channel Write or Read operation.	NA
<field>	ASCII	FLD1, FLD2	Selects Field.	
<channel>	Numeric	0-31	Selects Channel for Write or Read operation.	
<channel-data>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Data to be written to selected Channel. In hex format, MSB(Bit 31) is written to first word, Bit 30 to second Word, and so on.	

QUERY RESPONSE

<channel-data>

Query Data	Type	Value	Description
<channel-data>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Channel Data read in serial format. MSB (Bit 31) would be first word, Bit 30 would be second word, and so on.

COMMENTS

1. If table is larger than 32 words, multiple words may be sent. Next Data word's MSB will be written to the selected channel of word 33.
2. If multiple words are sent, bit 31 of the next word would be word 33. Up to 512 words can be programmed.
3. The query form returns a comma-separated list of numeric values where each numeric is 32 bits of serial channel data.

EXAMPLE

```
TABL:DEF PATT1,24
```

{Define PATT1 table with 24 words initialized to zero}

```
TABL:FIEL:CHAN PATT1,FLD2,12,#H12345600
```

{Programs channel 12 as shown below.}

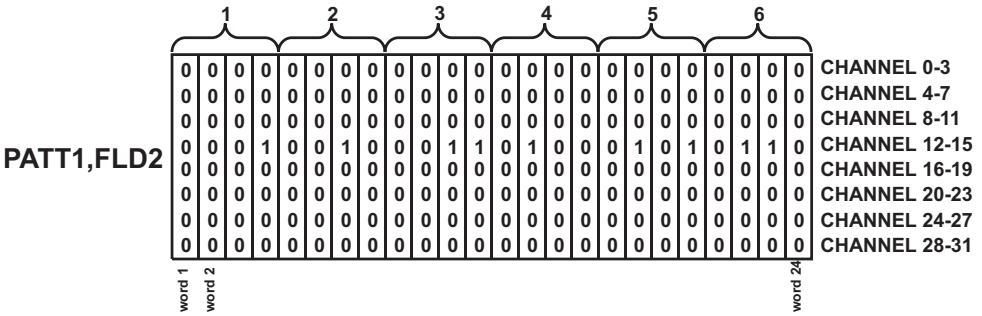


FIGURE 5-6 TABLE AFTER :CHANnel COMMAND

```
TABL:FIEL:CHANNEL? PATT1,FLD2,12
```

{Returns 305419776 (hex 12345600).}

5.8.4.5 :CEXPand

This command programs a single channel of BE-64 field memory without affecting the contents of the other channels. The <channel-data> parameter is 32 bits of serial channel data. Each bit of serial channel data is expanded to three table word bits as follows; logic 1 will be encoded as 110, logic 0 will be encoded as 100.

Additionally each expanded channel will contain a start bit (logic 1) and a stop bit (logic 0). The start bit will occupy the first three words(3 bits) of the table and the stop bit the last three words. Bit 31 of <channel-data> parameter is expanded to words 4, 5 and 6, bit 30 is expanded to words 7, 8, and 9, etc. Up to 32 bits can be programmed. The following function can be used to define the table size required for a channel expand;

(#_of_bits * 3) + 6

SYNTAX

TABLe:FIELd:CEXPand <table>,<field>,<channel><channel-data>

TABLe:FIELd:CEXPand? <table>,<field>,<channel>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Selects Table to perform Channel Write or Read operation.	NA
<field>	ASCII	FLD1, FLD2	Selects Field.	
<channel>	Numeric	0-31	Selects Channel for Write or Read operation.	
<channel-data>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Data to be written to selected Channel. In hex format, Bit 31, 30, and 29 are the start encoded bit (110), Data comes thereafter. The last three word bits are the stop bits (100).	

QUERY RESPONSE

<channel-data>

Query Data	Type	Value	Description
<channel-data>	Numeric	0-4,294,967,295 (0_h-FFFFFFFF$_h$)	Channel Data read in un-expanded serial forma(same format as it was written)t. MSB (Bit 31) would be first word, Bit 30 would be second word, and so on.

COMMENTS

1. The query form returns the unexpanded bits as a 32 bit numeric.

EXAMPLE

For example if we needed six bits expanded we would need to define a table with a size of (6 * 3) + 6 or 24 words.

```
TABL:DEF PATT1,24
```

{Define PATT1 table with 24 words initialized to zero}

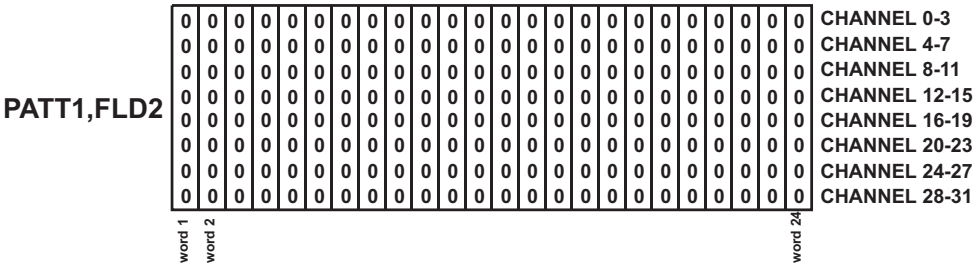


FIGURE 5-7 TABLE BEFORE :CEXPand

```
TABL:FIEL:CEXP PATT1,FLD2,3,#H12345600
```

{Programs channel 3 as shown below.}

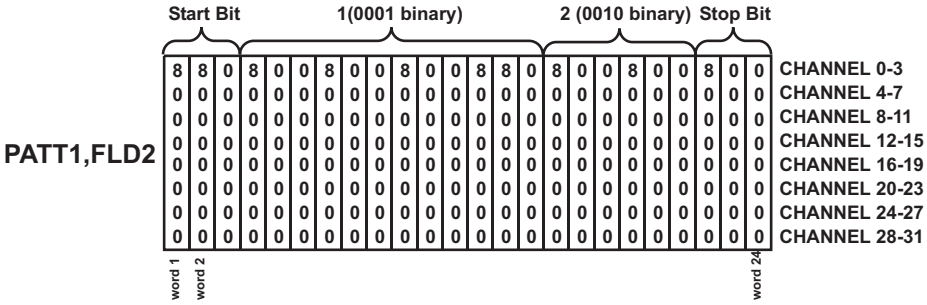


FIGURE 5-8 TABLE AFTER :CEXPand COMMAND

```
TABL:FIEL:CEXPAND? PATT1,FLD2,3
```

{Returns 268435456 (hex 10000000).}

5.8.4.6 :FILL

The FILL command executes the fill pattern on the table and field specified in the command.

SYNTAX

TABLe:FIELd:FILL <table>,<field>,<pattern>,<numeric1>[,<numeric2>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Select Table to perform a Fill operation.	NA
<field>	ASCII	FLD1, FLD2	Selects the Field.	
<pattern	ASCII	COMPlement	Perform a ones complement on the contents of each word.	
		INCRement	Successive words are formed by adding the <numeric2> parameter to the current word and then updating the current word for the next increment.	
		RAMP	The initial word is set to zero. Successive words are formed by setting succeeding bit positions to one until all bit positions are one. Then zeros are written starting with the MSB.	
		RANDom	The <numeric2> parameter is used as the initial word value. A pseudo random number is generated from the initial word and stored in successive words.	
		ROTate	Successive words are generated by left shifting the current word by the <numeric2> value and then updating the current word for the next rotate operation.	
		REPeat	Repeats the first word throughout the tables memory.	
		TOGGle	Successive words are generated by performing a ones complement on the current word and then updating the current word for the next complement operation.	
<numeric1>	Numeric	1 - 32768	Represents the starting transfer word for the beginning of the FILL function.	
<numeric2>	Numeric	NA	The data will be aligned and sized according to the current field width specified in the WIDTh command, (i.e. a BYTE width field will be filled with an eight bit pattern).	

COMMENTS

1. The range of <numeric1> is from 1 to the table size as specified in the TABLe:DEFine.

EXAMPLE

```
TABL:FIELd:FILL PATT1,FLD2,RAMP,1
```

{Fill PATT1 FLD2 with a ramp pattern.}

```
TABL:FIEL:FILL: PATT1,FLD1,ROT,1,1
```

{Fill FLD1 of PATT1 by rotating successive words by one.}

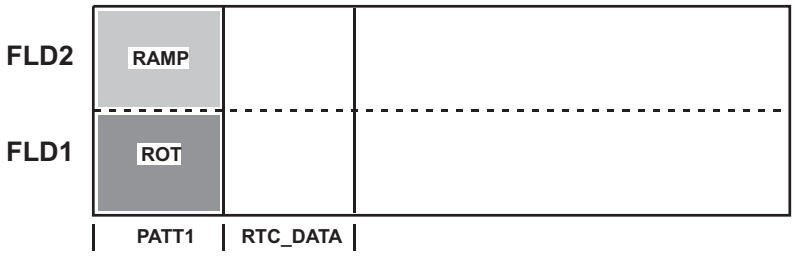


FIGURE 5-9 :FILL EXAMPLE

5.8.5 :DELete

The DELete subtree allows the operator to delete tables and de-allocate the field memory assigned to them.

Table memory allocated above the deleted table is re-aligned to prevent fragmentation.

5.8.5.1 [:NAME]

The NAME command deletes the specified table and de-allocates its field memory.

SYNTAX

TABLe:DELete[:NAME] <table>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Selects the Table to be deleted.	NA

COMMENTS

None

EXAMPLE

```
TABLe:DELete PATT1
```

{Deletes PATT1 from memory and de-allocates PATT1's pattern memory.}

5.8.5.2 :ALL

TABLe:DELete:ALL

The ALL command deletes any defined table and makes available all field memory.

5.8.6 :BENable

This command assigns one of the three byte enable modes to a table.

SYNTAX

TABLe:BENable <table>,<byte_enab>

TABLe:BENable? <table>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table>	Name	User-defined. See Section 5.2	Selects Table to perform a BENable Operation.	NA
<byte_enab>	ASCII	BYTE	Assigns respective Enable Mode to selected Table.	
		WORD		
		LONGword		

QUERY RESPONSE

<byte_enab>

Query Data	Type	Value	Description
<byte_enab>	ASCII	BYTE	Describes the BENable setting for the selected table.
		WORD	
		LONGword	

COMMENTS

1. The byte enable mode along with the least two significant channels of FLD1 defines a unique value for BE0 through BE3. The output signals values are programmed by the OUTP:BENable:DATA command.
2. The query form returns the byte enable code as an ASCII string.

EXAMPLE

```
TABLe:BEN PATT1,WORD
```

{Sets PATT1 enable codes to WORD.}

```
TABL:BEN? PATT1
```

{Returns: "WORD".}

5.8.7 :DIRectory

Returns a directory listing of all the defined table names followed by size and the A24 offset memory location in pattern memory. See appendix D for A24 programming. The Returned value follow the following patern of parameters:

```
<table>,<size>,<offset>;<table>...
```

SYNTAX

TABLe:DIRectory?

QUERY RESPONSE

<Table>, <size>, <offset>

Query Data	Type	Value	Description
<table>	Name	User-defined. See Section 5.2	All tables contained in the BE-64.
<size>	Numeric	1 - 32768	Table size
<offset>	Numeric	NA	A24 offset memory location, see appendix D for A24 programming.

COMMENTS

None.

EXAMPLE

```
TABLe:DIR?
```

{Returns "_DATA",24,262336;"PATT1",24,262144}

5.8.8 :FREE

This command returns the amount of used and free pattern memory in bytes.

SINTAX

TABLe:FREE?

QUERY RESPONSE

<used_memory>, <free_memory>

Query Data	Type	Value	Description
<used_memory>	Numeric	1 - 32768	Memory used by all tables in Memory.
<free_memory>	Numeric	1 - 32768	Total Memory - <used_memory>.

COMMENTS

1. It should be noted that it takes eight bytes to form 1 word (i.e. if there is 8K bytes of pattern memory free then the biggest table that could be defined would be 1K (free_memory / 8).

EXAMPLE

```
TABLe:FREE?
```

{Returns 48, 32720 }

5.9 TIMing SUBSYSTEM

The TIMing subsystem allows the operator to set and edit the timing setup selections and the sixteen timing sets (timing memory, field control settings).

KEYWORD	PARAMETER FORM	NOTES
TIMing		
:DEFine	<timing_set>,(<timing_set-source> | <number-of-cells>)	
:SETup		
:CLOCk	<source>	
:CTIMeout	<number-of-clocks>	
:DELay	<number-of-clocks>	
[:DATA]	<timing_set>,<block>	
:FCONtrol		
:DIRection	<timing_set>,<field>,<field-mode>	event only
:OREGister	<timing_set>,<field>,<field-mode>	event only
:OCONtrol	<timing_set>,<field>,<field_control>	event only
:ISTRobe	<timing_set>,<field>,<field_control>	event only
:CELL	<timing_set>,<timing_set>,<cell-number>,<cell-data>	
:TEST		
:DELay	<timing_set>,<cell-number>	event only
:LEVel	<timing_set>,<test-source>, <level>, <cell-number>	event only
:STRobe	<timing_set>,<level>,<cell-number>	event only
:RESet	<timing_set>,<cell-number>	event only
:DELete		
[:NAME]	<timing_set>	event only
:ALL		event only
:DIRectory?		query only

5.9.1 :DEFine

The DEFine command names and initializes a timing cycle.

SYNTAX

TIMing:DEFine <timing_set>,(<timing_set-source> | <number-of-cells>)

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. See Section 5.2.	Name of the timing set and can be 10 characters. First letter must be an alpha character.	NA
<timing_set-source>	Name	User-defined. See Section 5.2.	Name of existing Timing set to be copied to <timing_set>.	
<number-of-cells>	Numeric	2-256 (Even only)	Number of cells assigned with the specified number of cells set to #HFFFF (see CELL command to modify its values).	

COMMENTS

1. When assigning a <timing_set> parameter, characters can be alphanumeric or the underscore character (_).
2. Timing sets control the behavior of the BE-64 during run time.
3. Copying <timing_set-source> to the new timing implies copying of the existing timing set including the field control settings.
4. Each timing set must have an even number of cells. If an odd number is specified, a parameter error will occur.

EXAMPLE

```
TIMing:DEF READ_MEM,12
```

{Sets the number of cells in READ_MEM cycle to 12.}

```
TIM:DEF READ_IO,READ_MEM
```

{Creates a new timing cycle called READ_IO and copies the data from READ_MEM.}

5.9.2 :SETup

The SETup subtree allows the operator to assign the timing setup parameters, clock source, delay value and timeout.

5.9.2.1 :CLOCk

The CLOCk command sets the clock source for the timing sets.

The valid selections for internal clocks are 10, 20 or 50 MHz.

SYNTAX

TIMing:SETup:CLOCk <source>

TIMing:SETup:CLOCk?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<source>	Numeric|ASCII	10 | 20 | 50 | EXTernal	Sets Clock Frequency to 10, 20, 50 or to External Clock Frequency, respectively.	NA

QUERY RESPONSE

<source>

Query Data	Type	Value	Description
<source>	Numeric|ASCII	10 | 20 | 50 | EXTernal	Returns the Timing Set Clock frequency setting or EXTERNAL if an external CLOCK is used.

COMMENTS

1. External Clock may be set from DC to 50MHz.

EXAMPLE

```
TIM:SETup:CLOCk EXT
```

{Sets the clock source to external.}

```
TIM:SET:CLOC 20
```

{Sets the clock source to 20 Mhz.}

5.9.2.2 CTIMeout

The CTIMeout command allows the operator to specify the number of clocks before a cycle timeout is flagged in the status register for the specified cycle.

SYNTAX

TIMing:SETup:CTIMeout <number-of-clocks>

TIMing:SETup:CTIMeout ?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<number-of-clocks>	Numeric	0 - 32768	Sets the time-out to the given number of clocks. A value of 0 disables the timeout.	0

QUERY RESPONSE

<number-of-clocks>

Query Data	Type	Value	Description
<number-of-clocks>	Numeric	0 - 32768	Describes the number of clocks the time-out is set to.

COMMENTS

1. A cycle timeout indicates that a timing set test condition never occurred within the specified number of clocks. Time-out timing can be calculated as follows:

   Time-out= <number-of-clocks> * (1/ Clock_Frequency)

EXAMPLE

```
TIM:SET:CTIMeout 100
```

{Sets the cycle timeout to 100 Clocks.}

5.9.2.3 :DELay

The DELay command allows the operator to set the delay that each test delay cell will take for the specified cycle.

SYNTAX

TIMing:SETup:DELay <number-of-clocks>

TIMing:SETup:DELay?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<number-of-clocks>	Numeric	0 - 32768	Sets the Delay to the given number of clocks. A value of 0 disables the Delay.	0

QUERY RESPONSE

<number-of-clocks>

Query Data	Type	Value	Description
<number-of-clocks>	Numeric	0 - 32768	Describes the number of clocks the Delay is set to.

COMMENTS

1. A Delay placed on a particular cell will extend its timing to the number of clocks the Delay is set to.
2. Delay timing can be calculated as follows:

   Delay= <number-of-clocks> * (1/ Clock_Frequency)

EXAMPLE

```
TIM:SET:DELay 2
```

{Sets the delay cell value to 2 clocks.}

5.9.3 [:DATA]

The DATA command is used to program the specified timing memory.

SYNTAX

TIMing[:DATA] <timing_set>,<block>

PARAMETER LIST

Each four bytes of data in the block represents a separate cell and its bit representation is defined as follows:

Parameter	Type	Values	Description	Default
<block>	Numeric See Section 5.2.3	Byte 1:	Level of TSOUT1 - TSOUT8 where 0 = low and 1 = high.	#HFFFF
		Byte 2:	Bit 0 Internal EN FLD1 enable state 0 = enabled 1 = disabled Bit 1 Internal EN FLD2 enable state 0 = enabled 1 = disabled Bit 2 Internal STR FLD1 strobe level. Field register will be clocked on a high to low transition. Bit 3 Internal STR FLD2 strobe level. Field register will be clocked on a high to low transition. Bit 4 PRB FLD1 signature clock level 0 = low 1 = high Bit 5 PRB FLD2 signature clock level 0 = low 1 = high Bit 6 TRIG signal leve 0 = low 1 = high Bit 7 Not Used.	
		Byte 3:	Bit 0 Test code 0 (T0) Bit 1 Test code 1 (T1) Bit 2 Test code 2 (T2) Bit 3 - Bit 7 Not used. Bit2|Bit1|Bit0 0 | 1 | 0 | Test TSINPUT2 low 0 | 1 | 1 | Test TSINPUT2 high 1 | 0 | 0 | Test TSSTROBE neg. edge 1 | 0 | 1 | Test TSSTROBE pos. edge 1 | 1 | 0 | Test for delay 1 | 1 | 1 | No test	
		Byte 4:	Bit 0 FLD1 direction contro 0 = internal 1 = external Bit 1 FLD1 direction if bit 0 = 0 0 = output 1 = input FLD1 input strobe control if bit 0 = 1 0 = internal 1 = external Bit 2 FLD1 output register 0 = OFF 1 = ON Bit 3 FLD1 output enable/strobe control 0 = internal 1 = external Bit 4 FLD2 direction control 0 = internal 1 = external Bit 5 FLD2 direction if bit 0 = 0 0 = output 1 = input FLD2 input strobe control if bit 0 = 1 0 = internal 1 = external Bit 6 FLD2 output register 0 = OFF 1 = ON Bit 7 FLD2 output enable/strobe control 0 = internal 1 = external	

5.9.4 :FCONtrol

The FCONtrol subtree allows the operator to set the FLD1/FLD2 field control settings for the specified timing set.

5.9.4.1 :DIRection

The DIRection command allows the operator to set the direction for FLD1 and FLD2 for the specified timing set.

SYNTAX

TIMing:FCONtrol:DIRection <timing_set>,<field>,<field-mode>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. See Section 5.2	Selects Timing set to program the direction for FLD1, and FLD2	NA
<field>	ASCII	FLD1, FLD2	Selects the Field.	
<field-mode>	ASCII	INPut	Sets the direction for FLD1 or FLD2.	
		OUTPut		
		EXTernal		

COMMENTS

None.

EXAMPLE

```
TIM:FCONtrol:DIR READ_IO,FLD1,OUTPut
```

{Sets the FLD1 direction to output for the READ_IO cycle.}

```
TIM:FCON:DIR READ_IO,FLD2,INPut
```

{Sets the FLD2 direction to input for the READ_IO cycle.}

5.9.4.2 :OREGister

The OREGister command allows the operator to set the field output registers on or off.

SYNTAX

TIMing:FCONtrol:OREGister <timing_set>,<field>, <state>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>,	Name	User-defined. See Section 5.2.	Selects the Timing Set.	NA
<field>	ASCII	FLD1, FLD2	Selects the Field.	
<state>	ASCII	ON	Turns the output Register on.	
		OFF	Turns the output register off.	

COMMENTS

1. The field output registers are only used when the field direction is set to OUTput or EXTernal.
2. Setting the field output registers off makes the output data transparent (i.e. no output strobe is required).

EXAMPLE

```
TIM:FCONtrol:OREG READ_IO,FLD1,ON
```

{Sets the FLD1 output register on for READ_IO timing set.}

5.9.4.3 :OCONtrol

The OCONtrol command allows the operator to select the field output strobe and enable signal for the specified timing set.

SYNTAX

TIMing:FCONtrol:OCONtrol <timing_set>,<field>,<field_control>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. See Section 5.2.	Selects the Timing Set.	NA
<field>	ASCII	FLD1, FLD2	Selects the Field.	
<field_control>	ASCII	INTernal	Selects the EN FLD and STR FLD timing set signals to control the field output registers.	
		EXTernal	Selects the FnENAB and STBFLD signals from the front connector.	

COMMENTS

1. The field output registers are only used when the field direction is set to OUTput or EXTernal.

EXAMPLE

```
TIM:FCON:OCON READ_MEM,FLD1,INT
```

{Sets the FLD1 output strobe and enable signals for READ_MEM timing to internal.}

5.9.4.4 :ISTRobe

The ISTRobe command allows the operator to select the field input strobe signal for the specified timing set.

SYNTAX

TIMing:FCONtrol:ISTRobe <timing_set>,<field>,<field_control>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. See Section 5.2.	Selects the Timing Set.	NA
<field>	ASCII	FLD1, FLD2	Selects the Field.	
<field_control>	ASCII	INTernal	Selects the STR FLD timing set signal to strobe the field input registers.	
		EXTernal	Selects the STBFLD signal.	

COMMENTS

None.

EXAMPLE

```
TIM:FCON:ISTR READ_MEM,FLD2,INT
```

{Sets the FLD2 input strobe to internal (STROBE FIELD2) for READ_MEM.}

5.9.5 :CELL

The CELL command allows the editing of timing cell data.

SYNTAX

TIMing:CELL <timing_set>,<cell-number>,<cell-data>

TIMing:CELL? <Timing_set>,<cell-number>

PARAMETER LIST

Parameter	Type	Values		Description	Default
<Timing set>	Name	User-defined. See Section 5.2.3	Selects the timing set .		NA
<cell-number>	Numeric	1-256	Selects the timing set cell number to be edited/queried		
<cell-data>	Numeric	0-32767 (0_h-$7FFF_h$)	BYTE 1:	Bit 0-7 Level of TSOUT1 - TSOUT8 0 = low. 1 = high.	#H3FFF
			BYTE 2:	Bit 0 Internal EN FLD1 enable state 0 = enabled. 1 = disabled. Bit 1 Internal EN FLD2 enable state 0 = enabled. 1 = disabled. Bit 2 Internal STR FLD1 strobe level. Field register will be clocked on a high to low transition. Bit 3 Internal STR FLD2 strobe level. Field register will be clocked on a high to low transition. Bit 4 PRB FLD1 signature clock level 0 = low. 1 = high. Bit 5 PRB FLD2 signature clock level 0 = low. 1 = high. Bit 6 TRIG signal level 0 = low. 1 = high. Bit 7 Not Used.	

QUERY RESPONSE

<cell-data>

Query Data	Type	Value	Description
<cell-data>	Numeric	0-32767 (0_h-$7FFF_h$)	Timing set cell programmed value. See description above to determine its characteristics.

COMMENTS

1. The timing set size is defined by the TIMing:DEFine command.

EXAMPLE

```
TIM:CELL READ_IO,16,#H0055
```

{Sets cell number 16 of READ_IO timing set to hex 0x55.}

5.9.6 :TEST

The TEST subtree allows the operator to insert a test cell into a timing cycle memory.

5.9.6.1 :DELay

The DELay command inserts a test delay into the specified timing set at the cell indicated by <cell-number>.

SYNTAX

TIMing:TEST:DELay <timing_set>,<cell-number>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. See Section 5.2	Selects the timing set .	NA
<cell-number>	Numeric	1-256	Selects the timing set cell number where Delay will be placed	

COMMENTS

1. The timing set size is defined by the TIMing:DEFine command. The amount of delay is programmed by the TIMing:SETup:DELay command.

EXAMPLE

```
TIMing:TEST:DEL  READ_IO,4
```

{Inserts a delay in cell 4 of READ_IO.}

5.9.6.2 :LEVel

The LEVel command inserts a HIGH or LOW level test into the specified timing set at the cell and signal indicated by <cell-number> and TSINput1 or TSINput2.

SYNTAX

TIMing:TEST:LEVel <timing_set>, <test-source>, <level>, <cell-number>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. SeeSection 5.2	Selects the Timing Set.	NA
<test-source>	ASCII	TSINput1, TSINPut2	Selects a Test Signal Sourceto be placed in selected <cell-number>.	
<level>	ASCII	HIGH/LOW	Selects a Level test for the selected <test-source>	
<cell-number>	Numeric	1-256	Selects the timing set cell.	

COMMENTS

1. The timing set size is defined by the TIMing:DEFine command.

EXAMPLE

```
TIM:TEST:LEVel READ_IO,TSINput1,HIGH,5
```

{Inserts a test high level in cell 5 of READ_IO on the TSINput1 input signal.}

5.9.6.3 :STRobe

The STRobe command inserts a HIGH or LOW strobe transition test cell into the specified timing set at the cell indicated by <cell-number>.

SYNTAX

TIMing:TEST:STRobe <timing_set>,<level>,<cell-number>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. SeeSection 5.2	Selects the Timing Set.	NA
<level>	ASCII	HIGH/LOW	Inserts a HIGH or LOW strobe transition test cell into the specified timing set <cell-number>	
<cell-number>	Numeric	1-256	Selects the timing set cell.	

COMMENTS

The timing set size is defined by the TIMing:DEFine command.

A strobe test is not valid in the last cell of a timing set.

EXAMPLE

```
TIM:TEST:STR READ_MEM,HIGH,6
```

{Inserts a low-high edge test in cell 6 of READ_MEM.}

5.9.6.4 :RESet

The RESet command removes a previously defined TEST delay, level or strobe from the cell indicated by <cell-number> in the specified timing set.

SYNTAX

TIMing:TEST:RESet <timing_set>,<cell-number>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. SeeSection 5.2	Selects the Timing Set.	NA
<cell-number>	Numeric	1-256	Selects the timing set cell.	

COMMENTS

1. The timing set size is defined by the TIMing:DEFine command.

EXAMPLE

```
TIM:TEST:RES READ_MEM,6
```

{Removes the strobe test set up in cell 6 by the example in section 5.4.6.3.}

5.9.7 :DELete

The DELete subtree allows the operator to delete timing sets and reset the memory to default.

5.9.7.1 [:NAME]

The NAME command deletes the specified timing set and resets the memory to the power-on default values.

SYNTAX

TIMing:DELete[:NAME] <timing_set>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timing_set>	Name	User-defined. SeeSection 5.2	Selects the Timing Set.	NA
<cell-number>	Numeric	1-256	Selects the timing set cell.	

COMMENTS

None.

EXAMPLE

```
TIM:DEL READ_MEM
```

{Deletes READ_MEM Timing set.}

5.9.7.2 :ALL

TIMing:DELete:ALL

The ALL command deletes all the defined timing cycles and resets the memory to the power-on defaults.

5.9.8 :DIRectory

Returns a directory listing of all the defined timing cycles followed by size and the A24 memory offset.

`<Timing_set>,<size>,<offset>;<timing_set>...`

SYNTAX

TIMing:DIRectory?

QUERY RESPONSE

<field>

Query Data	Type	Value	Description
<Timing_set>	Name	User-defined. See Section 5.2	Timing Set Name(s)
<size>	Numeric	2-256	Timing Set size.
<offset>	Numeric	Defined in Appendix D.	Timing Set A24 Memory offset location of first cell.

COMMENTS

1. May return multiple Timing sets parameters.

5.10 COMMON COMMANDS

The BE-64 responds to the following Common commands (for further information refer to the IEEE 488.2-1987 section number which follows the command):

5.10.1 *CLS - Clear Status (section 10.3)

The Clear Status command clears status data structures and forces the device to the Operation Complete Command Idle Status (OCIS) and the Operation Complete Query Idle Status (OQIS).

5.10.2 *ESE - Event Status Enable Command (section 10.10)

*ESE <numeric>

The Standard Event Status Enable command sets the Event Status Enable Register mask enabling the operator to select specific bits to be passed through to the Event Status summary Bit (ESB) of the Status Register. Refer to section 4.5 for a description of the status reporting structures of the BE-64.

The <numeric> can range from 0 through 255 and is the mask enable value (e.g. a value of “3" will enable bits 0 and 1).

5.10.3 *ESE? - Event Status Enable Query (section 10.11)

The Standard Event Status Enable query allows the programmer to determine the current contents of the Standard Event Status Enable Register.

The response to the query will be an integer between the range of 0 and 255 representing the mask value entered by the *ESE command.

5.10.4 *ESR? - Event Status Register Query (section 10.12)

The Standard Event Status Register query allows the programmer to determine the current contents of the Standard Event Status Register. Reading the Standard Event Status Register clears it.

Bit 0:	OPC, Operate Complete.
bit 1:	RQC, Request Control.
bit 2:	QYE, Query Error.
bit 3:	DDE, Device Dependent Error.
bit 4:	EXE, Execution Error.
bit 5:	CME, Command Error.
bit 6:	URQ, User Request.
bit 7:	PON, Power On.
bits 8 - 15	Reserved for future use.

The response to the query will be an integer between the range of 0 and 255 representing the value of the 8 bits in the event status register.

5.10.5 *IDN? - Identification Query (section 10.14)

The intent of the identification query is for the unique identification of devices over the system interface. The response will be:

TALON INSTRUMENTS,BE-64,0,x.xx

(x.xx is the firmware version number)

5.10.6 *OPC - Operation Complete Command (section 10.18)

The Operation Complete command causes the device to generate the operation complete message in the Standard Event Status Register when all pending selected operations have been finished.

5.10.7 *OPC? - Operation Complete Query (section 10.19)

The Operation Complete query places an ASCII character 1 into the device’s Output Queue when all pending selected device operations have been finished.

The response will be: 1.

5.10.8 *RST - Reset Command (section 10.32)

The Reset command performs a device reset. The Reset command is the third level of reset in a three level reset strategy.

The Reset does the following:

1. Sets the device-specific functions to a known state that is independent of the part-use history of the device (all subsystems to default values, parser position remains unchanged, memory is not tested).
2. Forces the device into the OCIS state (Operation complete Command Idle State).
3. Forces the device into the OQIS state (Operation complete Query Idle State).

5.10.9 *SRE - Service Request Enable Command (section 10.34)

*SRE <numeric>

The Service Request Enable command sets the Service Request Enable Register mask bits permitting the programmer to select which messages in the Status Byte Register may cause service requests.

The <numeric> can range between 0 and 255. The mask value of bit 6 will be ignored; it is a fixed logical 0.

5.10.10 *SRE? - Service Request Enable Query (section 10.35)

The Service Request Enable query allows the programmer to determine the current contents of the Service Request Enable Register.

The response will be the enable value from 0 through 63 or 128 through 191.

5.10.11 *STB? - Read Status Byte Query (section 10.36)

The Read Status Byte query allows the programmer to read the status byte and Master Summary Status bit.

bit 0:	BSY, Sequence Running.
bit 1:	TMO, Timeout.
bit 2:	IDLE, Idle Running.
bit 3:	QSB, Questionable Status summary Bit.
bit 4:	MAV, Message Available).
bit 5:	ESB, Event Status summary Bit.
bit 6:	MSS, Master Status Summary (Request Service).
bit 7:	OSB, Operation Status summary Bit.

The response will be a value from 0 through 255.

5.10.12 *TST? - Self-Test Query (section 10.38)

The Self-Test query causes an internal self-test and places a response into the Output Queue indicating whether or not the device completed the test without any detected errors. This query should be the first command after power-up. Device settings return to power up defaults.

The response results are defined as follows;

bit 0:	High indicates field memory failure.
bit 1:	High indicates byte enable/sync memory failure.
bit 2:	High indicates timing set memory failure.
bit 3:	High indicates master clock failure.
bits 4 - 31	Reserved for Extended Self Test fixture results.

5.10.13 *WAI - Wait-to-Continue Command (section 10.39)

The Wait-to-Continue command will prevent the device from executing any further commands or queries until the No-Operation-Pending flag is TRUE.

5.11 LOW LEVEL INTERFACE COMMANDS

The BE-64 responds to the following low level VXIbus interface commands. These are issued by its commander. Refer to the VXIbus specification for further information.

5.11.1 AMC - Asynchronous Mode Control

This command is used by the commander to direct the path of events from the BE-64.

5.11.2 ANOP - Abort Normal Operation

This command causes the BE-64 to cease its normal operation as fast as possible. It resets the ready bit in the Status register to 0. The unit is reconfigured to the default state; the power-up memory test is NOT performed. It has the same effect as *RST and *CLS.

5.11.3 BNOP - Begin Normal Operation

This commands the BE-64 to begin normal operation. The ready bit in the status register is set to 1 and the device is ready to receive data.

5.11.4 CEV - Control Event

This command is used by a Commander to selectively enable the generation of events by the BE-64.

5.11.5 CLE - Clear

Upon receiving the clear command, the BE-64 removes all output data in the VXI Data Low register. The BE-64 will discard all data due for output from operations currently in progress and become ready to receive a new command; the input and output queues are cleared. The read ready bit in the Response register will be reset to 0. The Err* bit of the Response register will be de-asserted and any pending read protocol error command responses will be canceled.

5.11.6 ENOP - End Normal Operation

The BE-64 ends its normal operation in an orderly manner without any time limit constraints. Incoming signals are no longer processed. The ready bit of the status register is reset to 0. The current BE-64 configuration is maintained.

5.11.7 RPR - Read Protocol

The VXIbus commander uses this command to find out what protocols, in addition to the Word Serial protocol, that the BE-64 supports. The BE-64 supports I4.

5.11.8 RPE - Read Protocol Error

The VXIbus commander uses this command to query the cause of the last protocol error.

5.11.9 RSTB - Read Status Byte

This command is used by the commander to read the status word from a servant device.

APPENDIX A SCPI BEGINNERS GUIDE

1 INTRODUCTION

SCPI (Standard Commands for Programmable Instruments) traces its lineage to IEEE 488.1 and IEEE 488.2. Although the IEEE 488.2 standard addressed some instrument measurements, it principally dealt with common commands and syntax or data formats. Please refer to the IEEE 488.2 and SCPI reference manuals for more information.

The IEEE 488.1 relates to the physical connection between the instrument and its remote control unit; how the data is transmitted between the two; and the method used to determine master and slave.

The IEEE 488.2 encompassed and built upon IEEE 488.1 by adding syntax and data requirements for the communication path. In addition, it also defined the commands which were to be common to any and all units, and the query format for data retrieval from the remote instrument.

With SCPI, a variety of modular instruments have a universal language, even though they have different functions and manufacturers. SCPI added the fine detail of instrument set-up by establishing a hierarchy of standard command formats and subsystem routing, reducing multiple ways to control similar functions. An example of vertical consistency (same instrument type) would be multimeters from different manufacturers implementing the command to measure a value of DC voltage in the same manner using the Measure sub-system. An example of horizontal consistency (different instrument types) would be different instruments using the same command to trigger a function using the Trigger subsystem. Refer to figure A-1. Queries of the instrument by the controller result in well defined status response and measurement data. By building on the IEEE 488.2 all of the earlier commands that it had defined have become a part of SCPI and, to the limit that an instrument can be operated by them, they are valid.

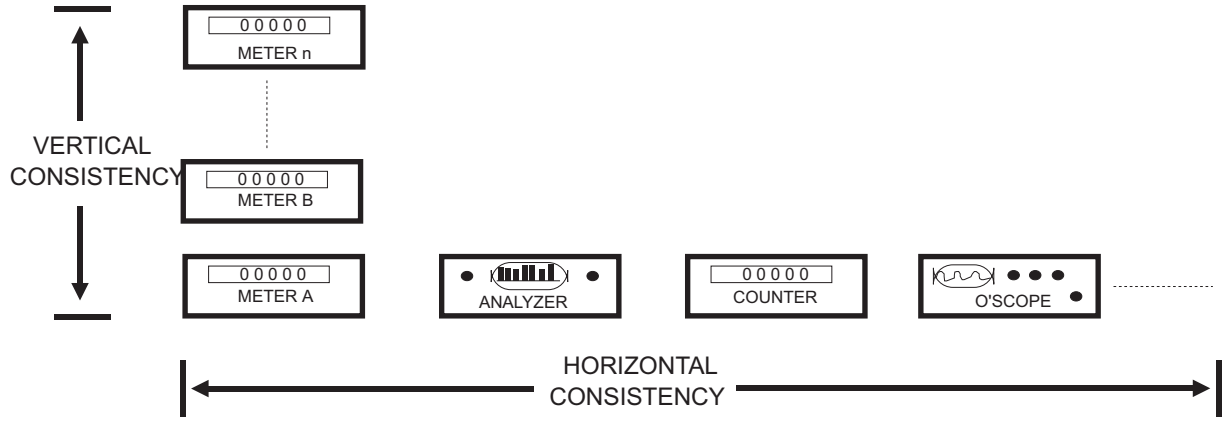


FIGURE A-1 COMMAND CONSISTENCY

The user need not be concerned about the interface commands, the common commands or the syntax and data structures, they are IEEE 488.1 and IEEE 488.2 definitions and have not changed; they are encompassed within SCPI. Not all instruments use all commands, but all instruments use the same command format. Using the language rules and the hierarchical nature of the command structure, new commands, parameters, and subsystems can be developed from the existing primitive elements and commands as new instruments are introduced.

The hierarchy of subsystem commands in SCPI is called a Command Tree (sometimes also called a Command Flow Chart). The SCPI Command Tree is up-side-down, the Root is at the top with branches extending downward, ending with the parameter required for the branch function. Refer to figure A-2. There is only one route to travel to reach the destination keyword or parameter on the selected branch. More than one command may have the same keyword, but on different branches. These commands

usually perform a similar function in the respective branches, however, each can only be reached by traversing a unique path.

The command hierarchy can also be represented by a Command Table and by Syntax Diagrams.

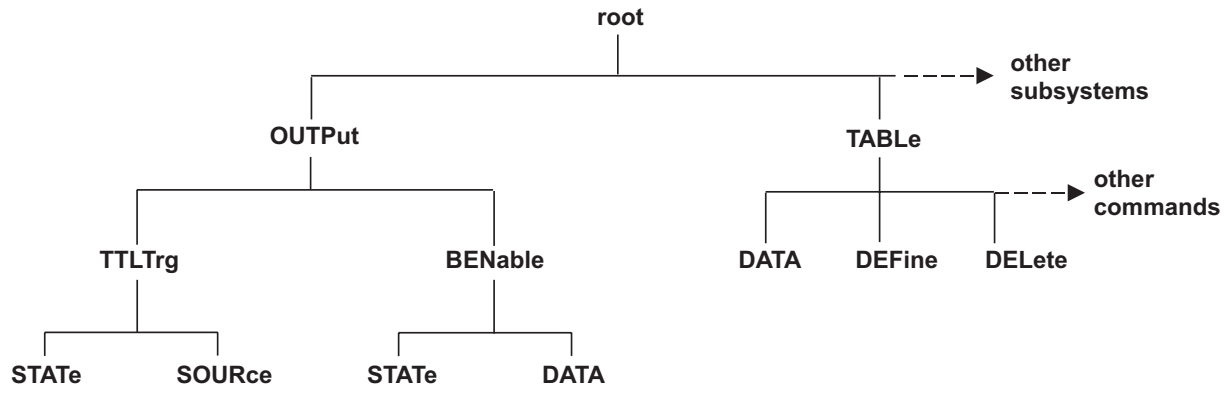


FIGURE A-2 SAME KEYWORDS IN DIFFERENT SUBSYSTEM

2 PARAMETERS

The parameter specifies the finest detail that is required in a branch. Most parameters have a defined default value at power-up or *RST. Parameter defaults are specified in the manual for each instrument. There are three major types of parameters: Character (Discrete), Decimal Numeric and Boolean.

Character parameters are a one word character label, usually one of a number of choices, that define a characteristic (e.g. EXTERNAL, the abbreviation external signal source).

Decimal Numeric parameters have an extended set that may be implemented. A decimal Numeric may not be used as a subset (e.g. 1 or n lines), but may be used as a value definition (e.g. start value = 10). Subsets are implemented by using a suffix with a discrete (e.g. FLD2). Decimal Numeric also covers the “label” type of parameter, the signal name (e.g. TSINPUT1).

Boolean parameters can have one of two values, usually ON or OFF. Boolean values can also be represented by “1" or ”0" on the command line. Queries requesting a Boolean value will always return either “1" or ”0".

A SCPI convention shows parameters in text and on the Command Tree with angle brackets <> if they are values or names to be entered by the user. Parameters with a character (discrete) keyword or Boolean parameters do not use the angle brackets.

3 QUERIES

Any command that sets a value can be queried about the current value of the setting. The query form of the command ends with a question mark (?). Some commands are defined as queries only and can be identified by the question mark used after them in the command tree. Some commands are events only and do not have a query form. These can usually be identified by their action nature such as DELete or RESet.

4 SCPI PUNCTUATION and SYNTAX

Keywords can be abbreviated or used in full. SCPI requires the exact abbreviation or the exact full spelling only; capital or lower case letters have equal weight. The long form of the keyword may be either a single word or a phrase which has been abbreviated to a single word. The SCPI convention is to use the entire keyword in any text or instructions with the accepted abbreviation shown in capital letters. For example,

EXECute

Common commands must start with an asterisk (*). SCPI commands start with an optional colon (:). Each time a colon is inserted in the command line, the “pointer” is instructed to move down the branch which has the keyword immediately following. A semi-colon separates a string of commands on one line. If a colon does not follow the semi-colon, the “pointer” remains at the same level. A colon following the semi-colon will set the “pointer” back to the root. The commands do not become effective until a “Program Message Terminator” is received at the end of the command line. An incorrect command line will generate an error message.

5 CONDENSED RULES:

- Power-on and Reset

After power is applied, the command “pointer” is set to the root and all parameters to default values.

- Command line termination

When the command line is terminated with a Program Message Terminator, the “pointer” is set to the root level.

- Colon

The optional leading colon in a command line indicates the keyword immediately following is at the root level (a subsystem).

A colon between command keywords indicates the pointer is to step down one level to the immediately following keyword.

- Semi-colon

The semi-colon separates commands within the same message without changing levels.

- Whitespace (space bar or tab)

Whitespace must be used to separate commands from parameters. Whitespace must not be used within a command keyword. Otherwise, SCPI usually ignores the whitespace.

- Comma

When a command requires a series of parameters, they must be separated by commas.

- Question Mark

The question mark is placed after the program header creating a query. A parameter may be placed after the question mark where appropriate (for example, TABLe? <table_ name>). Some event commands do not have a query form (for example, TABLe:DELete). Some commands are queries only (for example, TABLe:FREE?).

- Common Commands

Common commands (*RST, *TRG, etc.) are acted upon the same way regardless of which subsystem or into which level of the SCPI test program they are written. After execution of the common command, the SCPI command “pointer” will return to the point where it was interrupted (the exception is *CLS, the “pointer” is set back to the root). The *RST command will reset all subsystems to the default values.

6 TEXT SYMBOLS

- Square brackets

Commands or portions of parameters that are optional are enclosed by square brackets [ ].

- Angle brackets

Angle brackets < > enclose parameters that are to be entered by the user, usually either numeric label (names) or numeric data (levels, values).

- Curly brackets

Parameter(s) within curly brackets { } represent a repeating group.

- Parentheses and vertical line

The vertical (|) represents an “OR”, one of the values shown must be used.

7 CYCLE and DATA TABLE NAMES

The following restrictions govern parameter names (user names):

1. They cannot exceed 10 characters in length, preferred length is 4.
2. They must not have any embedded whitespace.
3. They must start with an alphabetic character. They cannot start with a numeric, a whitespace, or an underscore.
4. All alphanumeric characters may be used, both capital and lower case. The underscore (_) may also be used.
5. There is no abbreviation for a users name, it must be used as defined.

APPENDIX B COMMAND DESCRIPTION

1 SCPI CONFORMANCE

This instrument conforms to SCPI version l.0 published April, 1991.

Syntax of all SCPI confirmed commands implemented by the Talon Model BE-64 will be displayed in italic. There have been very few commands confirmed for digital testing so the majority of commands have not been confirmed at this date.

2 SCPI COMMAND SUMMARY

KEYWORD	PARAMETER FORM
CALCulate	
:CRC	<table_name>,<field_name>,<seed_value>[,<mask_value>]
:CRC32	<table_name>,<field_name>,<seed_value>[,<mask_value>]
:TCOMpare	<table_name>,<table_name>,<field_name>[,<mask_value>]
EXECute	
[:TIMing]	[<cycle_name>,(<table_name>|<fld1_data>,<fld2_data>,<byte_enab>)]
:SEQuence	[<cycle_name1>,<table_name>,<loop_value>,<cycle_name2>,...]
:FUNCtion	
:BUS	<fld1_mask>,<fld2_mask>
:BRAM	<start>,<stop>,<pattern>,<page_size>,<delay>,<byte_enab>
:EEPRom	<start>,<stop>,<pattern>,<page_size>,<delay>,<byte_enab>
:FLASh	<start>,<stop>,<pattern>,<write_delay>,<erase_delay>,<byte_enab>
:ROM	<start>,<stop>[,<byte_enab>]
:RAM	<start>,<stop>,<pattern>[,<byte_enab>]
:WIDTh	<byte_enab>
:FIELd	(PIO1 | PIO2 | OUT)[,<numeric_value>]
:MODE	<mode>[,<loop>]
OUTPut	
:BAENable	<boolean>
:BENable	
[:STATe]	<boolean>
:DATA	<byte_enab>,<numeric>,<numeric>,<numeric>,<numeric>
:FIELd	
:MODE	<field_name>,(OUTPut | OENabled | INPut | ISTRobed | COUNt)
:RESet	(HIGH | LOW | PULSE,<numeric_value>)
:SYNC	
[:STATe]	<boolean>
:POSition	<table_name>,<transfer_number>
:LOOP	<loop_value>
:TTLTrg<n>	
[:STATe]	<boolean>
:SOURce	(PFLD1 | PFLD2 | EXTernal | TRIGger)
:TTLTrg(A | B)	

KEYWORD	PARAMETER FORM
[:STATe]	<boolean>
:SOURce	TTLTrg<n>
TABLe	
[:DATA]	<table_name>,<block>
:BENable	<table_name>,<byte_enab>
:DEFine	<table_name>[,<number_words>|<table_name>]
:DELete	
[:NAME]	<table_name>
:ALL	

:DIRectory?	
:FIELd	
[:DATA]	<table_name>,<field_name>,<block>
:CEXPand	<table_name>,<field_name>,<channel>,<data_value>
:CHANnel	<table_name>,<field_name>,<channel>,<data_value>
:FILL	<table_name>,<field_name>,<fill_pattern>,<begin_word>[,<pattern_value]
:WIDTh	<table_name>,<field_name>,(BYTE | WORD | LONGword)
:WORD	<table_name>,<field_name>,<transfer_number>,<data_value>
:FREE?	
:WORD	<table_name>,<transfer_number>,<field1_data>,<field2_data>
TIMing	
[:DATA]	<cycle_name>,<block>
:CELL	<cycle_name>,<cell_number>,<numeric_value>
:DELete	
[:NAME]	<cycle_name>
:ALL	
:DIRectory?	
:DEFine	<cycle_name>,(<cycle_name>|<number_cells>)
:FCONtrol	
:DIRection	<cycle_name>,<field_name>,(INPut | OUTPut | EXTernal)
:ISTRobe	<cycle_name>,<field_name>,(INTernal | EXTernal)
:OCONtrol	<cycle_name>,<field_name>,(INTernal | EXTernal)
:OREGister	<cycle_name>,<field_name>,(ON | OFF)
:SETup	
:CLOCk	(50 | 20 | 10 | EXTernal)
:CTIMeout	<clock_count>
:DELay	<clock_delays>
:TEST	
:DELay	<cycle_name>,<cell_number>
:LEVel	<cycle_name>,(TSINput1 | TSINput2),(HIGH | LOW),<cell_number>
:STRobe	<cycle_name>,(HIGH | LOW),<cell_number>
:RESet	<cycle_name>,<cell_number>

3 SCPI MANDATED COMMANDS

KEYWORD	PARAMETER FORM
STATus	
:OPERation	
[:EVENt]?	
:CONDition?	
:ENABle	<mask_value>
:QUEStionable	
[:EVENt}	
:CONDition?	
:ENABle	<mask_value>
:PRESet	
SYSTem	
:ERRor?	
:VERSion?	

4 IEEE COMMON COMMANDS

*CLS	Clear Status Command
*ESE	Event Status Enable Command
*ESE?	Event Status Enable Query
*ESR?	Event Status Register Query
*IDN?	Identification Query

*OPC	Operation Complete Command
*OPC?	Operation Complete Query
*RST	Reset Command
*SRE	Service Request Enable Command
*SRE?	Service Request Enable Query
*STB?	Status Byte Query
*TST?	Self Test Query
*WAI	Wait to Continue Command

5 LOW LEVEL INTERFACE COMMANDS

AMC	Asynchronous Mode Control
ANO	Abort Normal Operation
BNO	Begin Normal Operation
CEV	Control Event
CLE	Clear
ENO	End Normal Operation
RPER	Read Protocol Error
RPR	Read Protocol
RSTB	Read Status Byte

APPENDIX C GUIDE TO BUS EMULATION

1 INTRODUCTION

Bus Emulation is a technology which enables an engineer to simulate all the signal characteristics of a given interface. Programmable bus emulation allows the user to software configure the programmable bus emulator to simulate literally an infinite number of digital interfaces.

The term 'bus emulation' generally implies microprocessor and/or bus structured interfaces. However, to infer that Talon's programmable bus emulator simulates only these categories of interfaces would be a substantial understatement. While the programmable bus emulator is ideally suited to simulate a VME bus, Multibus, SCSI bus, 80486, or 68030 bus structured interface, the emulator can also simulate almost any digital interface; interfaces such as serial communication interfaces, state sequencer interfaces or any user defined digital interface. Indeed, most digital interfaces can be considered to be a subset of a bus structured interface.

2 WORD GENERATOR OVERVIEW

Since "bus emulator" is a relatively new technology, this appendix will first describe the basic concept of a word generator. This will establish a baseline for the bus emulation description. The reader should note that the BE-64 module is capable of executing all the concepts which are described in this document.

A basic digital word or pattern generator is composed of a number of I/O channels where the group of channels define a single "word". Each channel is further described by a serial data stream. When the Talon BE-64 is looked at from this viewpoint, its word generator capacity is 64 I/O channels with a 32K word depth, see figure C-1.

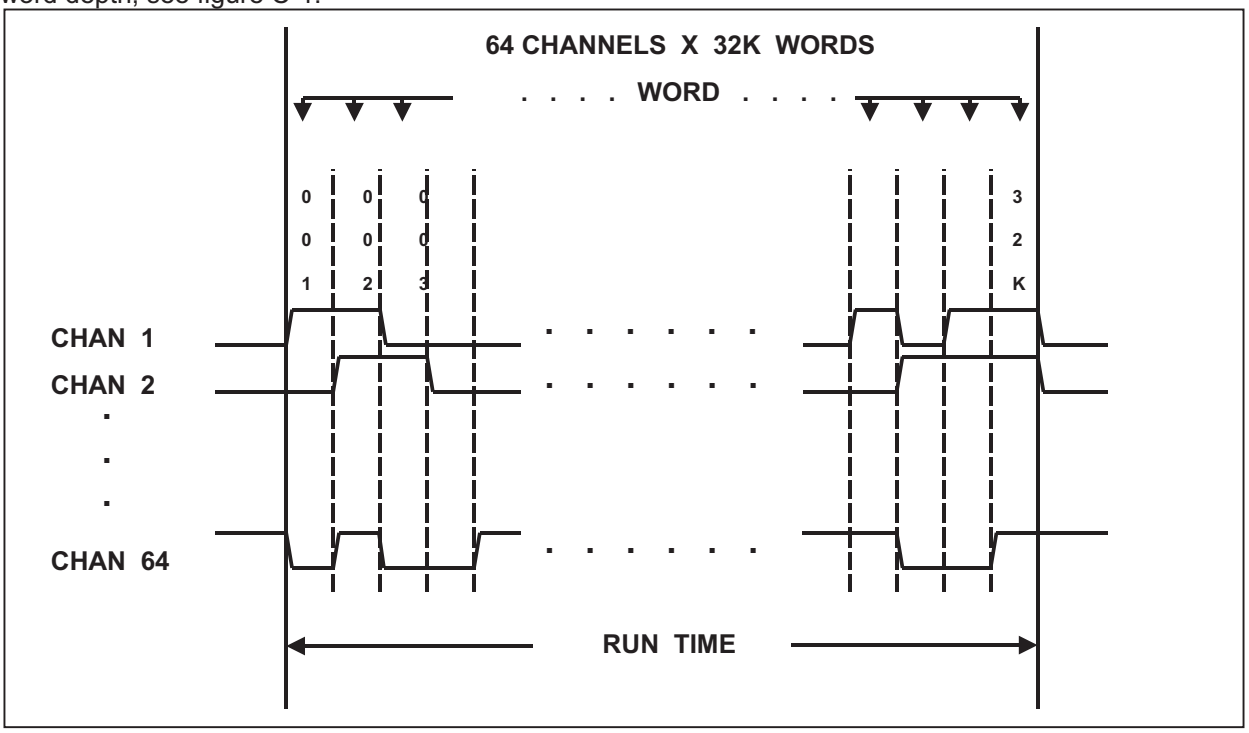


FIGURE C-1 64 CHANNELS BY 32K WORDS

In a run mode, the data would be output across the channels starting with word 1 and continuing through word 32K, generating a continuous stream of data. The time required to execute the entire table is referred to as RUN TIME.

For illustrations' sake we will consolidate figure C-1 into the abbreviated figure C-2 below.

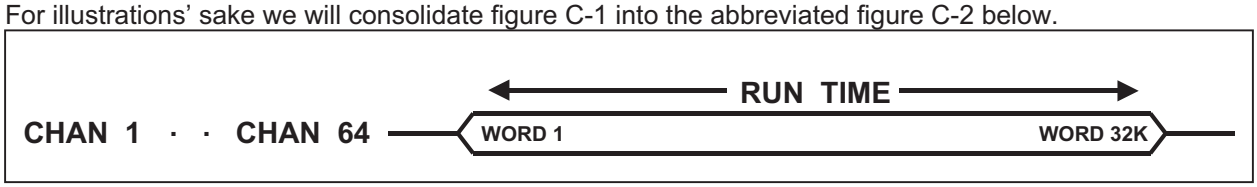


FIGURE C-5 64 CHANNELS CONSOLIDATED

2.1 Tables

When using a word generator, it is desirable to subdivide the entire word generator memory into smaller sections, each section dedicated to a specific UUT operation. The BE-64 refers to the divided memory as tables, see figure C-3. The BE-64 can be divided into multiple tables. The table length can be individually configured from 1 word to the full 32K memory depth. The RUN TIME is the time required to execute all the tables in the sequence. The execution of the tables is continuous with zero dead time between tables.

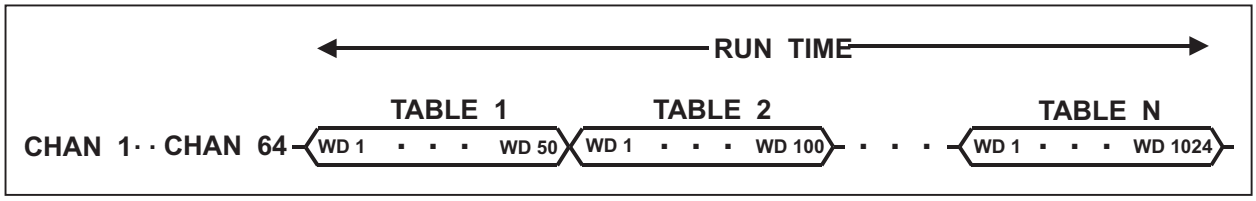


FIGURE C-2 TABLES

2.2 Table Looping

The ability to individually loop the tables is often desirable in order to create a repeating stimulus to the UUT. The BE-64 allows individual tables to be looped from 1 to 64K times, see figure C-4. In addition, tables can be looped continuously. A command from the VXI Slot 0 controller is required to exit the loop. Again, the RUN TIME is the amount of time to complete the entire sequence of tables.

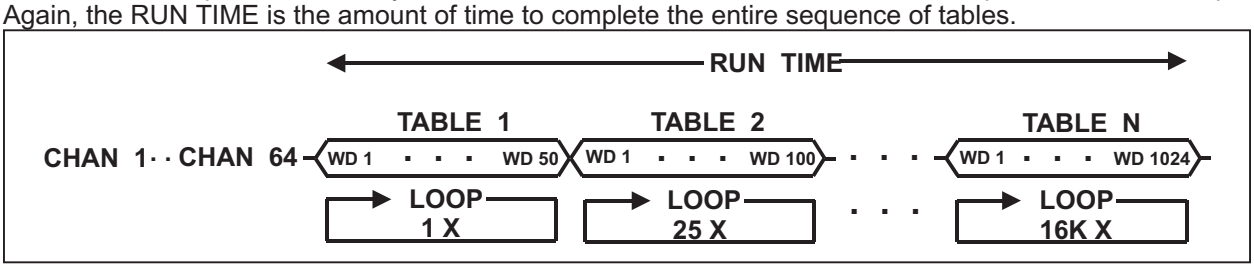


FIGURE C-3 TABLE LOOPING

2.3 Idle Cycle

The BE-64 incorporates a unique function which is not found on present data generators, this feature being the idle cycle. The idle cycle allows the user to define a timing and data sequence prior to and after the "run time", see figure C-5.

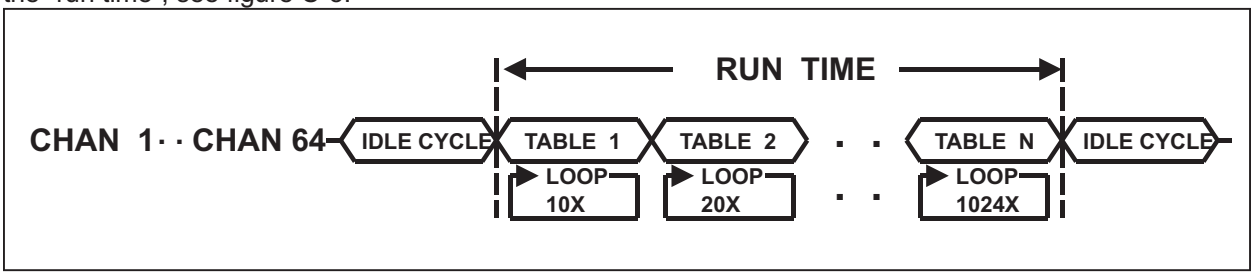


FIGURE C-4 IDLE CYCLE INCLUDED

This idle cycle can be programmed such that no signal activity is seen by the UUT, or a complete repetitive timing cycle with associated data patterns could be defined.

One of the key functions of the idle cycle is the ability of the VXI controller to download new data tables, loop count values, and sequence of operation while the idle cycle is running. This feature enables a UUT to continue to receive required timing signals while new data is defined for the next operation.

3 BUS EMULATION OVERVIEW

In general, word generator parameters describe the sequencing of data from one word to the next, and from one table of data to the next table. Bus emulation describes what happens “inside” each word. For example, let’s take a look at what’s happening inside word 2 of table 1, see figure C-6.

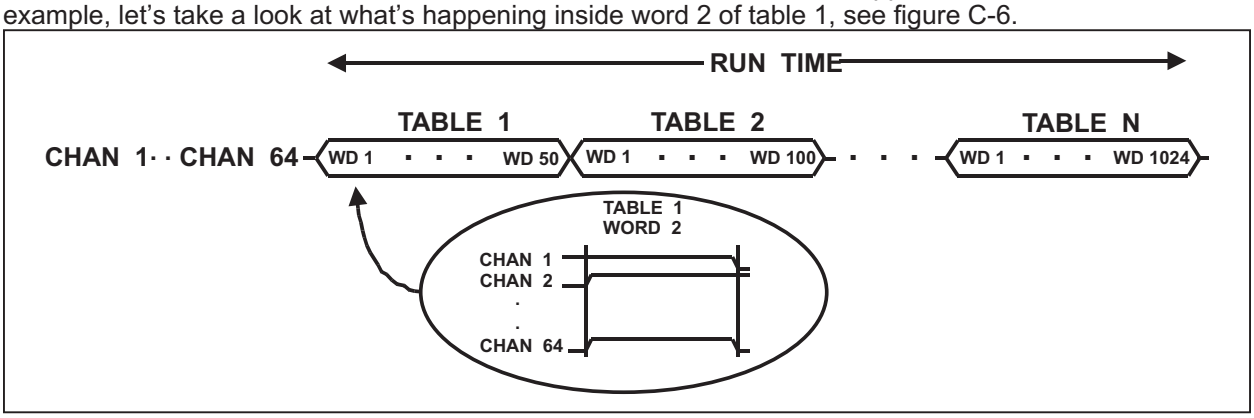


FIGURE C-6 “INSIDE” EACH WORD

The BE-64 allows an individual word to be divided into timing increments with resolution of 20 ns per increment. Figure C-7 depicts word 2 of table 1, where word 2 has a total period of 200ns.

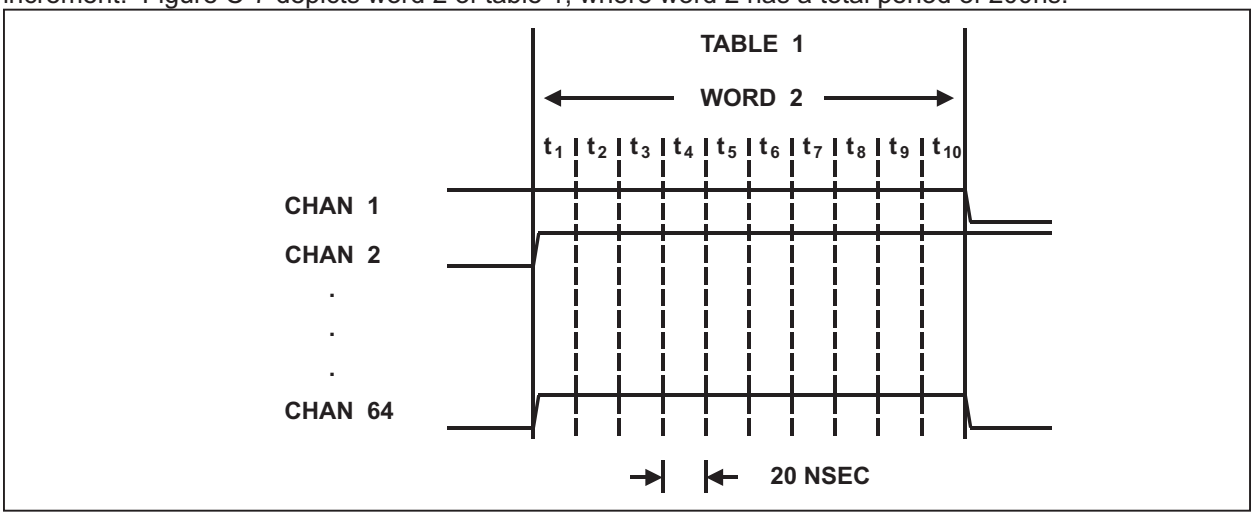


FIGURE C-7 WORD DIVIDED INTO TIMING INCREMENTS

3.1 Fields

figure C-10 again depicts word 2 of table 1. However, in this figure we will break up the 64 channels into

two fields, an address field and a data field. This configuration will allow our example to follow a more traditional bus structure interface.

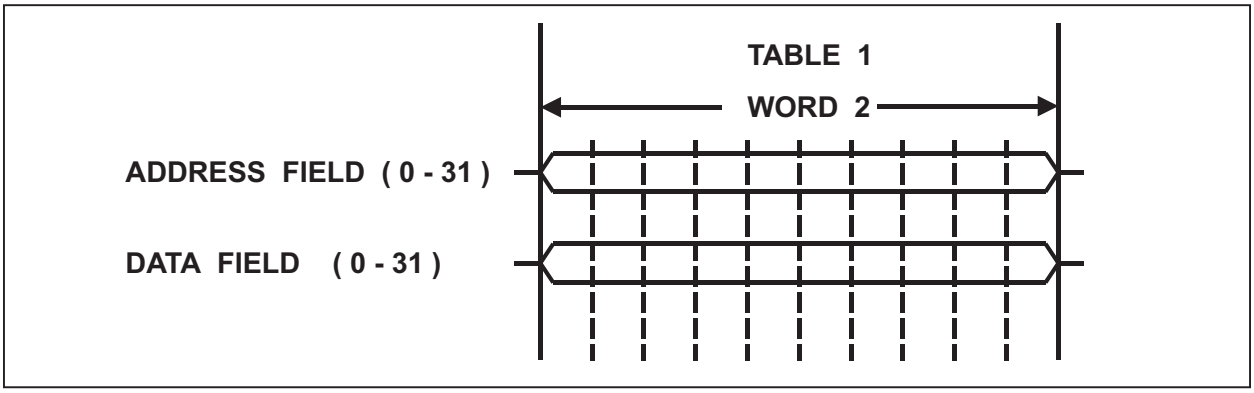


FIGURE C-10 BUS STRUCTURE WITH FIELDS

In most applications, the signals comprising a word in a field are not active during the entire word period. As shown in figure C-8, the address field period may be active from t2 (20ns) through t9 (180ns), while the data field is active from t4 (60ns) through t8 (160ns).

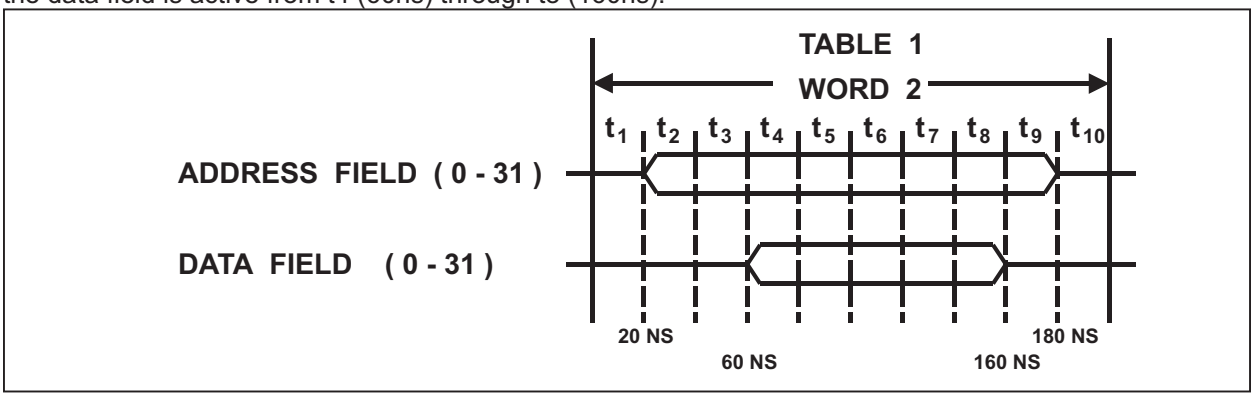


FIGURE C-8 IMPROVED BUS STRUCTURE

3.2 Field Timing

In order to achieve the “bus signals” depicted in figure C-10, the buses require control signals. In particular, they require an ADDRESS ENABLE and DATA ENABLE signal, figure C-9. These signals are generated from a separate timing generator, nomenclated the “timing set”.

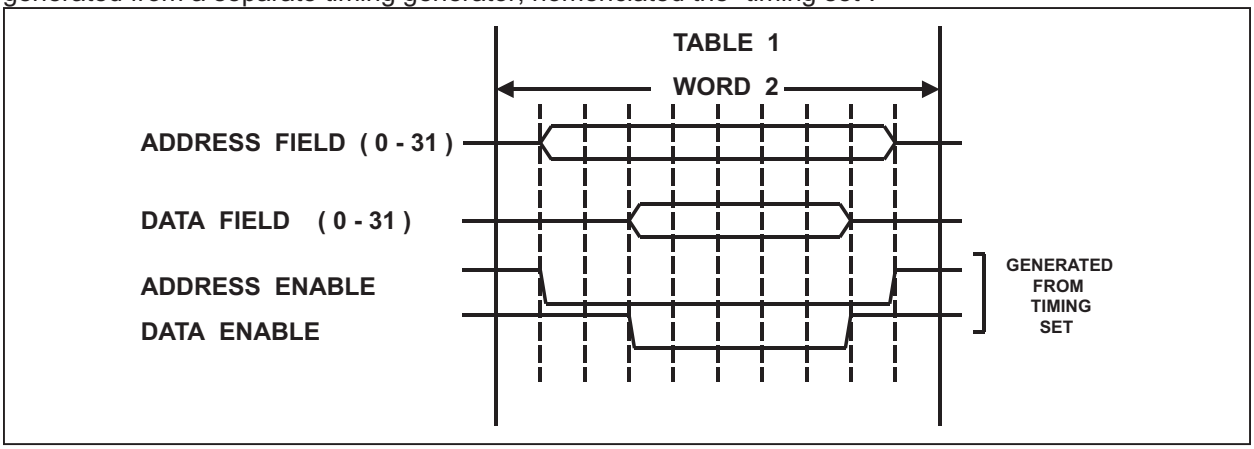


FIGURE C-9 FIELD TIMING SIGNALS

3.3 Timing Signals

Typically, the UUT requires additional control signals which further define the “bus cycle”. figure C-11 describes a UUT R/W-, a UUT Data Enable and a UUT Data Strobe signal. These signals are also generated by the BE-64 timing set.

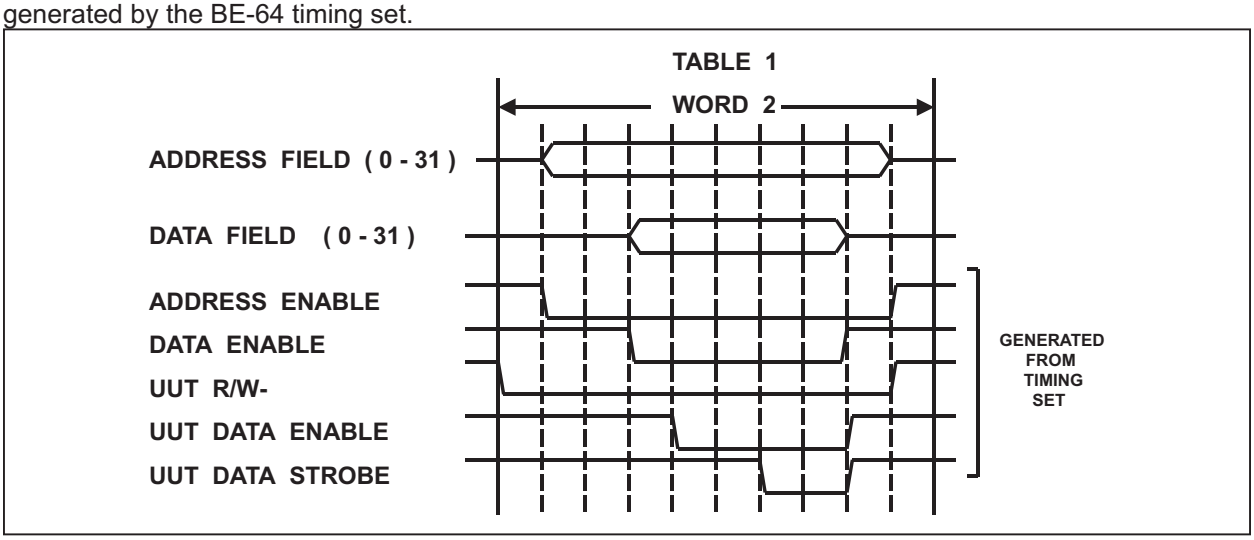


FIGURE C-11 “BUS CYCLE” TIMING SIGNALS

Once active data has been placed on the data bus and the data enable signal has been asserted, there must be a means to determine that the UUT is ready for data. The timing set must have the ability to test the UUT READY signal and enter a “WAIT” state until the test condition is true. This sequence is often referred to as a “handshake” sequence.

Figure C-12 indicates that the timing set will remain in state t5 until the UUT READY- signal goes low. Once the UUT READY- = Low condition is met, the timing set will continue. If the test condition does not occur in a specified time, a timeout condition will force the continuation of the timing set and inform the VXI controller that a timeout occurred. The timeout logic can be disabled, which in turn will cause the BE-64 to remain in state t5 until commanded to complete by the VXI controller.

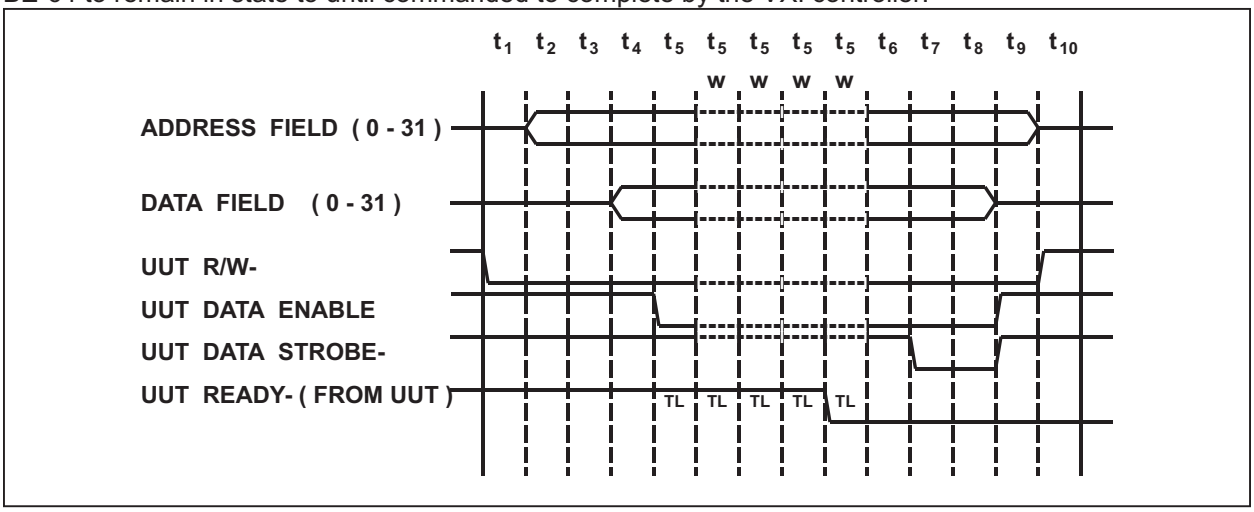


FIGURE C-12 TEST CONDITION AND “WAIT” STATE

The BE-64 incorporates 16 timing sets, each timing set programmed to simulate a particular bus cycle type or UUT timing cycle.

4 BUS EMULATION TESTING

Bus Emulation Testing is a test method in which the unit under test (UUT) is exercised, in real time, through all of its functions. This is accomplished by precisely simulating all the interfaces into and out of the UUT as well as transferring functional data to/from the UUT. Successful bus emulation testing requires independent bus emulators for each interface resident on the UUT.

figure C-13 depicts a typical UUT. The UUT has two interfaces, a VME interface and a SCSI bus interface. Successful test of the UUT requires two bus emulators, each providing a real time simulation of the address, data, and control lines of the respective bus, as well as the ability to transmit and receive functional data to/from the UUT.

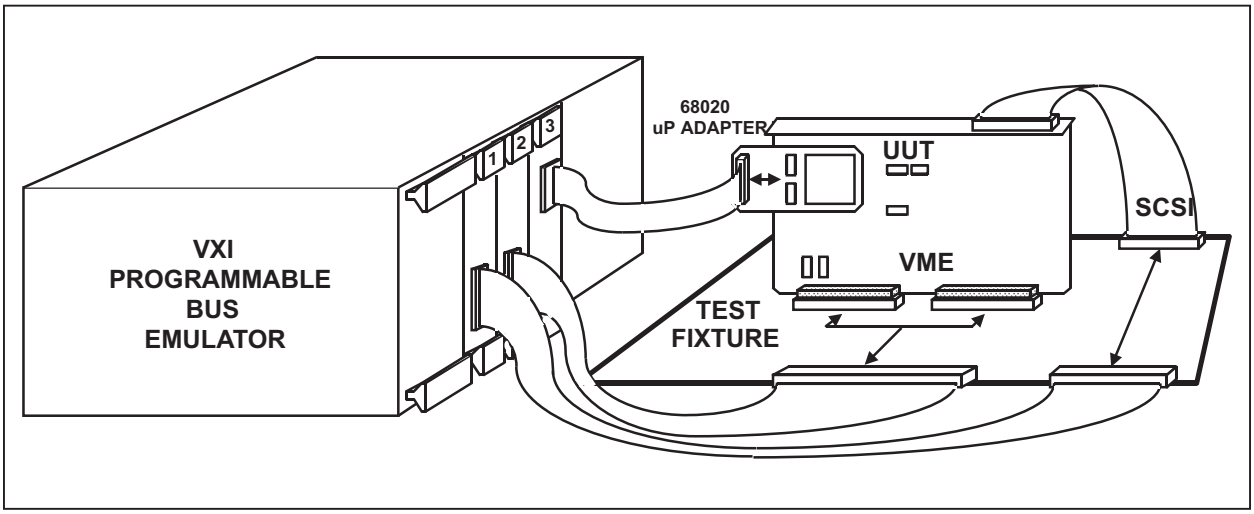


FIGURE C-13 SAMPLE UUT INTERFACE

This UUT also incorporates a 68020 microprocessor. If it is desirable to emulate the 68020, a third bus emulator would be installed to simulate this function. The UUT would then be exercised, in real time, through all its functions.

APPENDIX D BE-64 A16/A24 REGISTERS

1 INTRODUCTION

The BE-64, as with all VXI devices, exist within VME address space. This address space is divided into three types, A32, A24 and A16. The following sections describes how the BE-64 exists within this memory.

2 A16 MEMORY

Every VXI device is assigned a 64 word area of the A16 memory. The location of the 64 words is determined by the VXI devices "logical address". Figure D-1 is the A16 register map usage for the BE-64.

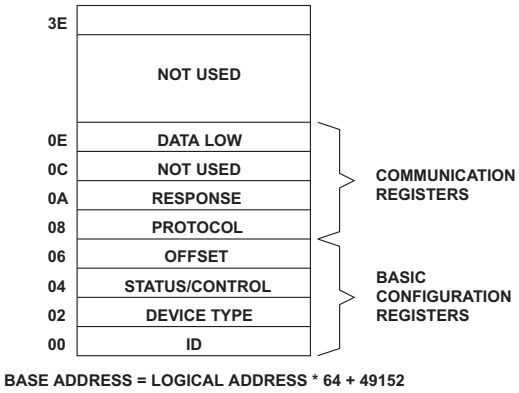


FIGURE D-1 A16 REGISTER MAP

The ID register provides information about the BE-64 to the controller. Read only.

ID Register:

Bit 0 - Bit 11	Manufacturer ID (3855/F0F hex).
Bit 12 - Bit 13	Address Space (0, A16/A24 addressing).
Bit 14 - Bit 15	Device Class (2, Message based)

The Device Type register contains device dependent information about the BE-64 to the controller. Read only.

Device Type Register:

Bit 0 - Bit 11	Model Code (256/100 hex).
Bit 12 - Bit 15	Required A24 Address Space (3, 1M A24 addressing).

The STATUS register contains bits that reflect the status of the BE-64. Read only.

Status Register:

Bit 0	PIO Strobed, reset by a PIO query.
Bit 1	UUT5V+ level.
Bit 2	Passed, 1 indicates self test passed.
Bit 3	Ready, 1 indicates BE-64 ready for operation.
Bit 4 - Bit 11	These eight bits are a copy of the STB byte defined in section 5.10.11 of the Operators manual
Bit 12 - Bit 13	Not Used.
Bit 14	Modid, 0 indicates the BE-64 is selected by the P2 MODID signal.
Bit 15	A24 Active, 1 indicates the the BE-64's A24 memory is enabled.

The CONTROL register contains several bits that affects the operation of the BE-64. Write only.

Control Register:

Bit 0	Reset, 1 causes BE-64 to go into the VXI reset state.

Bit 1	Sysfail Inhibit, 1 disable BE-64 from driving the SYSFAIL line.
Bit 2	Level of UUTRES.
Bit 3	Mode Reset, 0 sets the EXECute:MODE to RESet.
Bit 4	Mode Stop, 0 sets the EXECute:MODE to STOP.
Bit 5 - Bit 14	Not Used.
Bit 15	A24 Enable, 1 enables BE-64 A24 memory.

The OFFSET register provides base address of the BE-64's A24 memory. Read/Write.

Offset Register:

Bit 0 - Bit 11	Not Used
Bit 12 - Bit 15	Four most significant bits of the A24 base address.A24 Enable, 1 enables BE-64 A24 memory.

The PROTOCOL register indicates the VXI protocols the BE-64 supports. Read only.

Protocol Register:

Bit 0 - Bit 3	Not used
Bit 4 - Bit 9	Reserved by VXI.
Bit 10	A one indicates shared memory protocol not supported.
Bit 11	A one indicates fast handshake mode not supported.
Bit 12	A one indicates interrupter capability.
Bit 13	A one indicates no master capability.
Bit 14	A one indicates signal register not supported.
Bit 15	A one indicates servant only capability.

The RESPONSE register contains the status of the BE-64's communication registers. Read only.

Response Register:

Bit 0 - Bit 8	Not used
Bit 9	Write Ready, BE-64 ready for data transfer to DATA LOW register.
Bit 10	Read Ready, Data available in the DATA LOW register.
Bit 11	Err, zero indicates an error in the word serial protocol.
Bit 12	DIR, one indicates BE-64 ready for data using the byte transfer protocol.
Bit 13	DOR, one indicates BE-64 has message available using the byte transfer protocol.
Bit 14	Reserved by VXI.
Bit 15	Always zero.

The DATA LOW register is the register through which all word serial commands and messages are sent/received. A write to the DATA LOW register causes the BE-64 to perform some action based on the data written. A read of the DATA LOW register is one word of word serial data.

3 A24 MEMORY

Within the A16 register map certain registers tell the VXI controller how much, if any, A24 memory is required for each VXI device. The BE-64 requires 1Mbyte of A24 memory. The base address of the A24

memory is programmed by the VXI resource manager. Within the A24 memory of the BE-64 is the timing memory and field memory as shown in figure D-2.

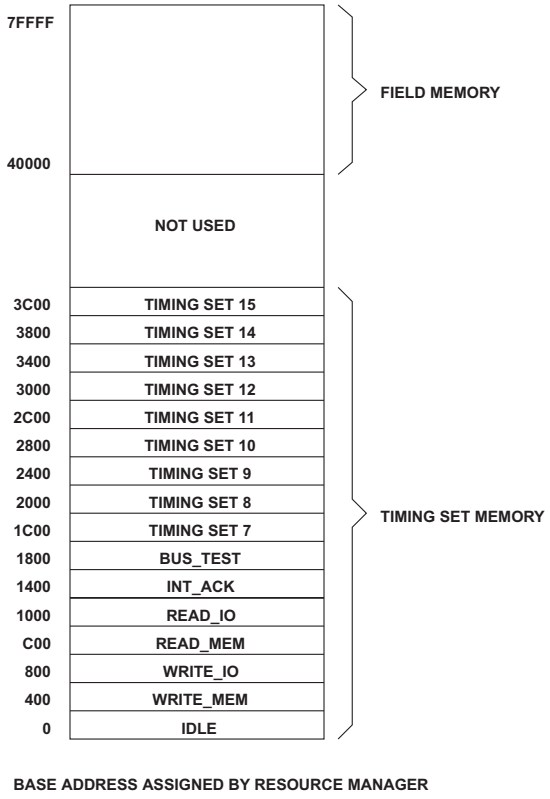


FIGURE D-2 A24 MEMORY MAP

3.1 Timing Memory Format

The timing memory of the BE-64 starts at a zero offset from the base address of the A24 memory and is accessed a word (sixteen bits) at a time.

The timing memory is divided into sixteen groups called timing sets. Each timing set contains 128 timing words. Each timing word is 64 bits and defines two consecutive cells, i.e., timing word one defines timing cell one and two.

The 64 bit timing word interlaces the two cells in the following manner:

First word:

Bit 0	TSOUT1 level for cell 1.
Bit 1	TSOUT1 level for cell 2.
Bit 2	TSOUT2 level for cell 1.
Bit 3	TSOUT2 level for cell 2.
Bit 4	TSOUT3 level for cell 1.
Bit 5	TSOUT3 level for cell 2.
Bit 6	TSOUT4 level for cell 1.
Bit 7	TSOUT4 level for cell 2.
Bit 8	TSOUT5 level for cell 1.
Bit 9	TSOUT5 level for cell 2.
Bit 10	TSOUT6 level for cell 1.
Bit 11	TSOUT6 level for cell 2.
Bit 12	TSOUT7 level for cell 1.
Bit 13	TSOUT7 level for cell 2.
Bit 14	TSOUT8 level for cell 1.
Bit 15	TSOUT8 level for cell 2.

Second Word:

Bit 0	EN FLD1 level for cell 1.
Bit 1	EN FLD1 level for cell 2.
Bit 2	EN FLD2 level for cell 1.
Bit 3	EN FLD2 level for cell 2.
Bit 4	STR FLD1 level for cell 1.
Bit 5	STR FLD1 level for cell 2.
Bit 6	STR FLD2 level for cell 1.
Bit 7	STR FLD2 level for cell 2.
Bit 8	PRB FLD1 level for cell 1.
Bit 9	PRB FLD1 level for cell 2.
Bit 10	PRB FLD2 level for cell 1.
Bit 11	PRB FLD2 level for cell 2.
Bit 12	TRIG level for cell 1.
Bit 13	TRIG level for cell 2.
Bit 14	Always 1.
Bit 15	Last Cell flag, 0 = last cell.

Third Word

Bit 0	T0 for cell 1.
Bit 1	T0 for cell 2.
Bit 2	T1 for cell 1.
Bit 3	T1 for cell 2.
Bit 4	T2 for cell 1.
Bit 5	T2 for cell 2.
Bit 6	Always 0.
Bit 7	Last timing word, 0 = last word, 1 for all previous words.
Bit 8	FLD1 direction control, 0 = internal.
Bit 9	If direction control internal: FLD1 OCONtrol/ISTRobe, 0 = internal. If direction control external: FLD1 OCONtrol, 0 = internal.
Bit 10	FLD1 OREGister, 0 = off.
Bit 11	If direction control internal: FLD1 direction, 0 = output. If direction control external: FLD1 OCONtrol, 0 = internal.
Bit 12	FLD2 direction control, 0 = internal.
Bit 13	If direction control internal: FLD2 OCONtrol/ISTRobe, 0 = internal. If direction control external: FLD2 OCONtrol, 0 = internal.
Bit 14	FLD2 OREGister, 0 = off.
Bit 15	If direction control internal: FLD2 direction, 0 = output. If direction control external: FLD2 OCONtrol, 0 = internal.

Fourth Word

NONE

Bits 8 through 15 of the third word must be programmed the same for every timing word.

3.2 Field Memory Format

The field memory of the BE-64 starts at offset 40000 hex from the base address of the A24 memory and is accessed a word (sixteen bits) at a time.

The field memory is divided into groups called tables. Each table set contains a variable number of table words. as defined in the TABLe:DEFine command. Each table word is 64 bits and defines two separate fields (FLD1 and FLD2).

Each table word is 64 bits.

The 64 bit word is formatted in the following manner:

Word 1

Bit 0	FLD1 channel 16
.	.
.	.
.	.

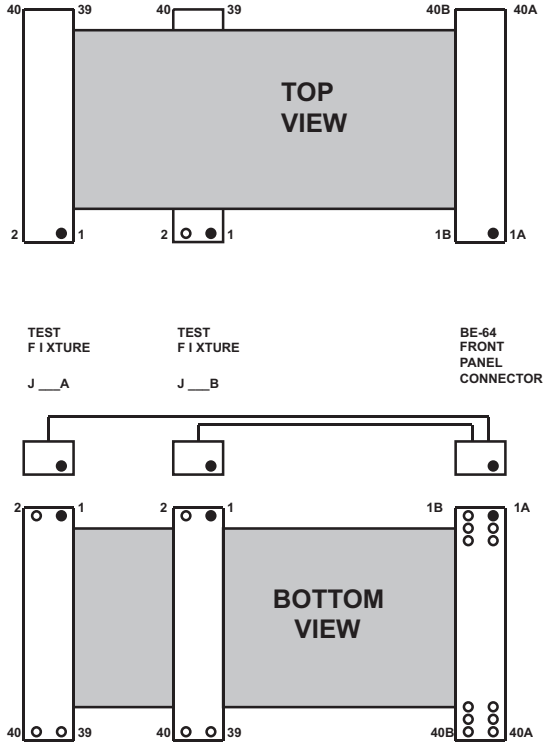
.	.
.	.
Bit 15	FLD1 channel 31
Word 2	
Bit 0	FLD1 channel 0
.	.
.	.
.	.
.	.
.	.
Bit 15	FLD1 channel 15
Word 3	
Bit 0	FLD2 channel 16
.	.
.	.
.	.
.	.
.	.
Bit 15	FLD2 channel 31
Word 4	
Bit 0	FLD2 channel 0
.	.
.	.
.	.
.	.
.	.
Bit 15	FLD1 channel 15

APPENDIX E CABLE DESIGN EXAMPLES

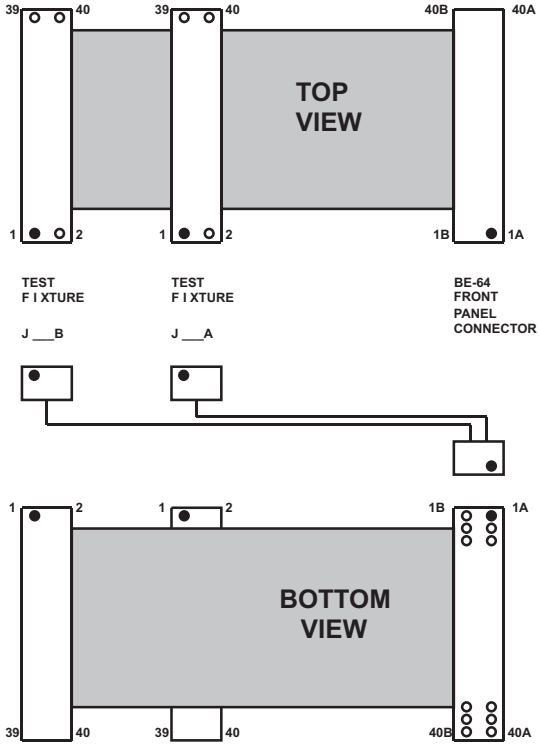
The following cable design examples are included:

(a) Typical cable for J1.
(b) Typical cable for J2.
(c) Typical cable for J3.
(d) Typical cable for J4.

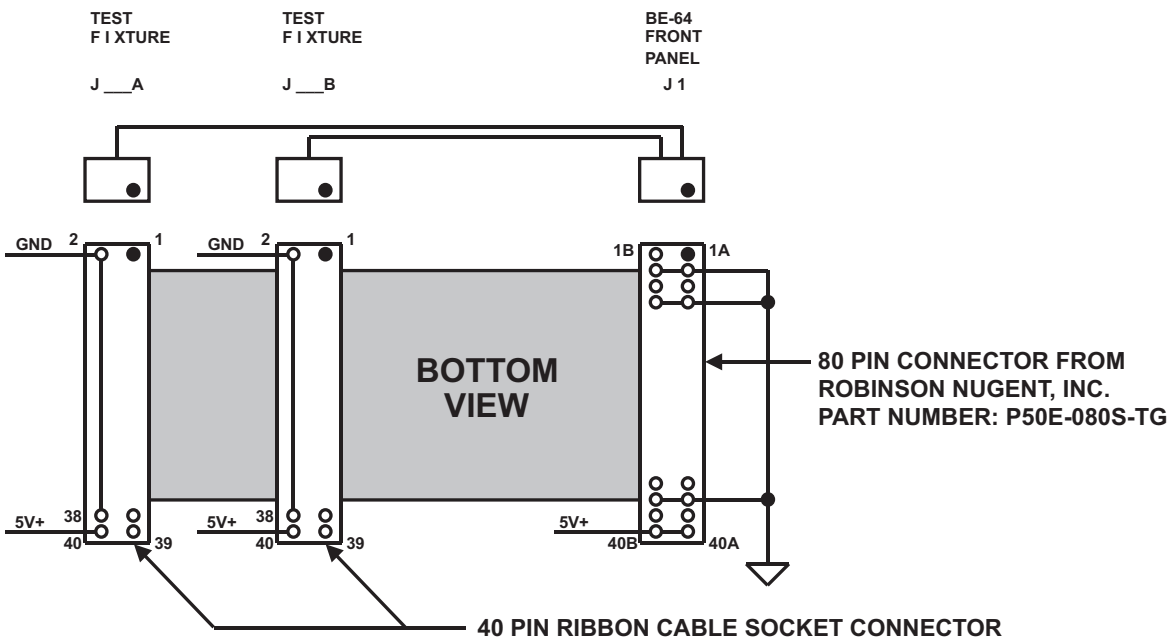
Talon offers two configurations of cables that mate with the BE-64. Both configurations include the 80 pin mating connector, three feet of ribbon cable and four forty pin headers, two spare and two crimped.

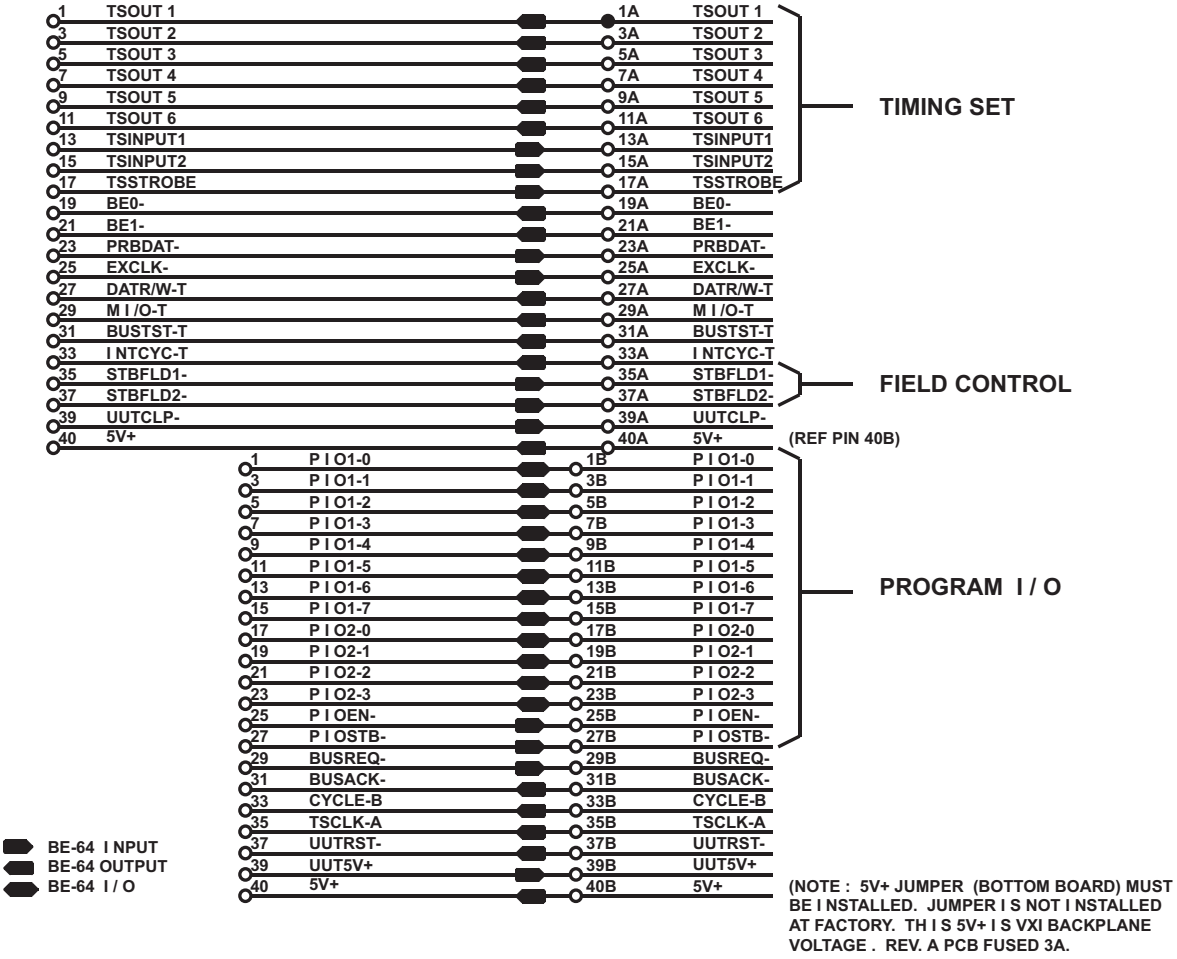


STANDARD MATING CABLE P/N BE64-300

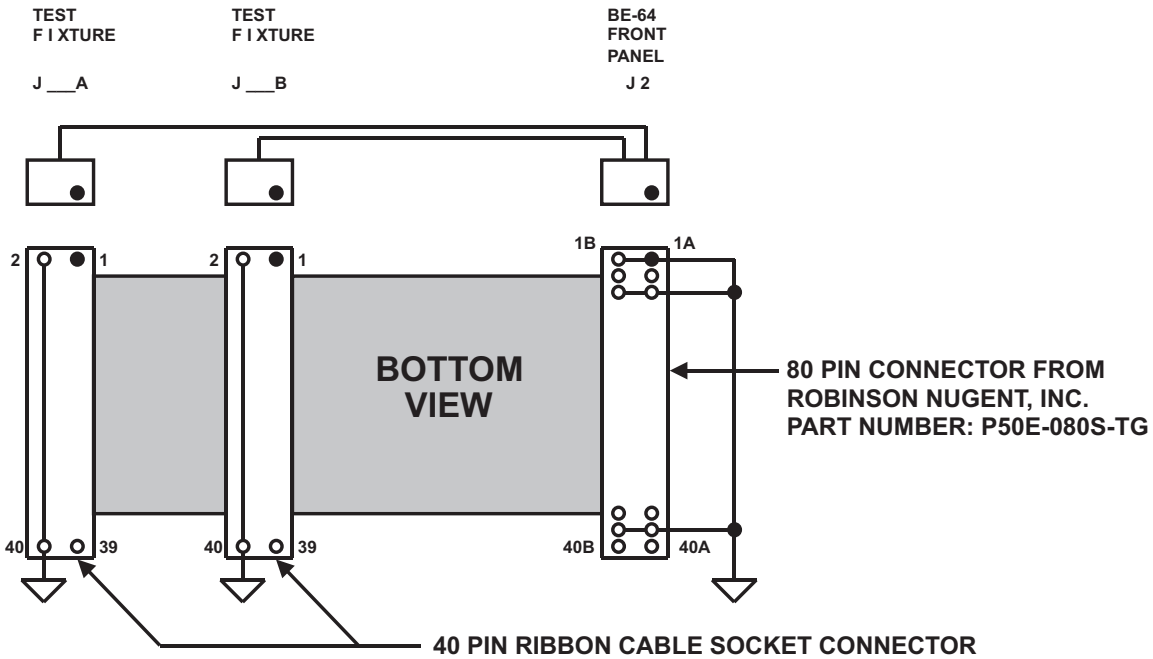


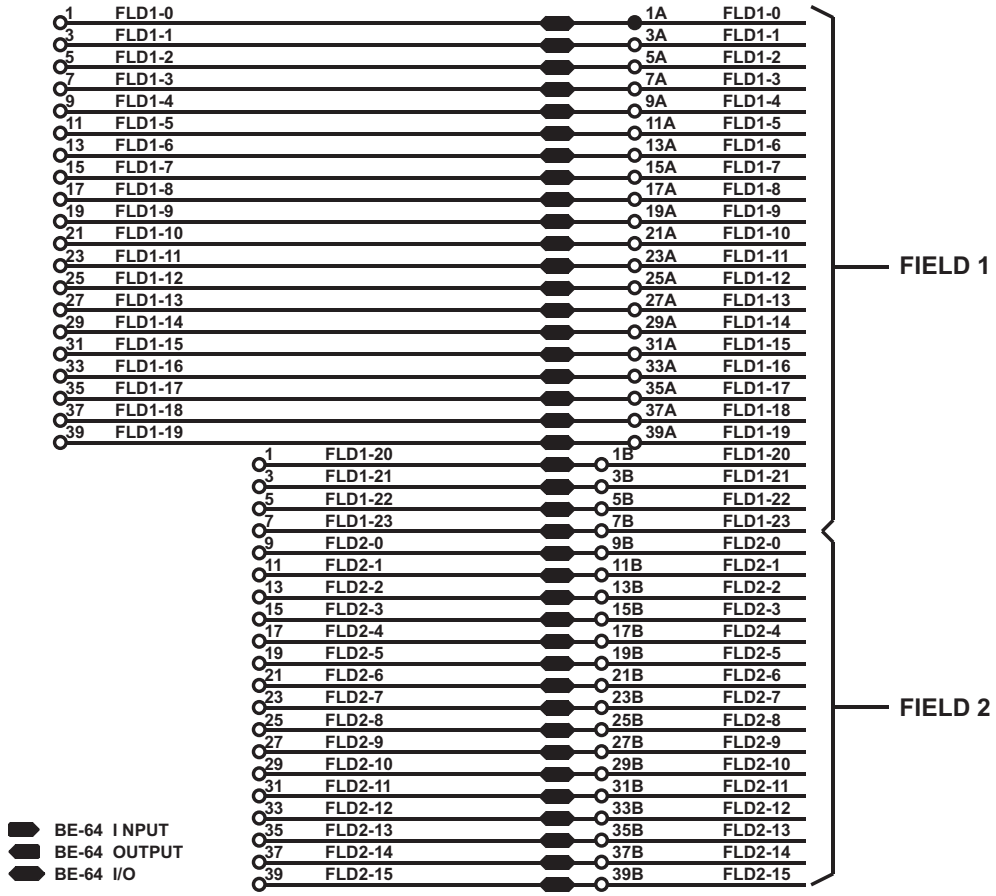
REVERSED MATING CABLE P/N BE64-300R



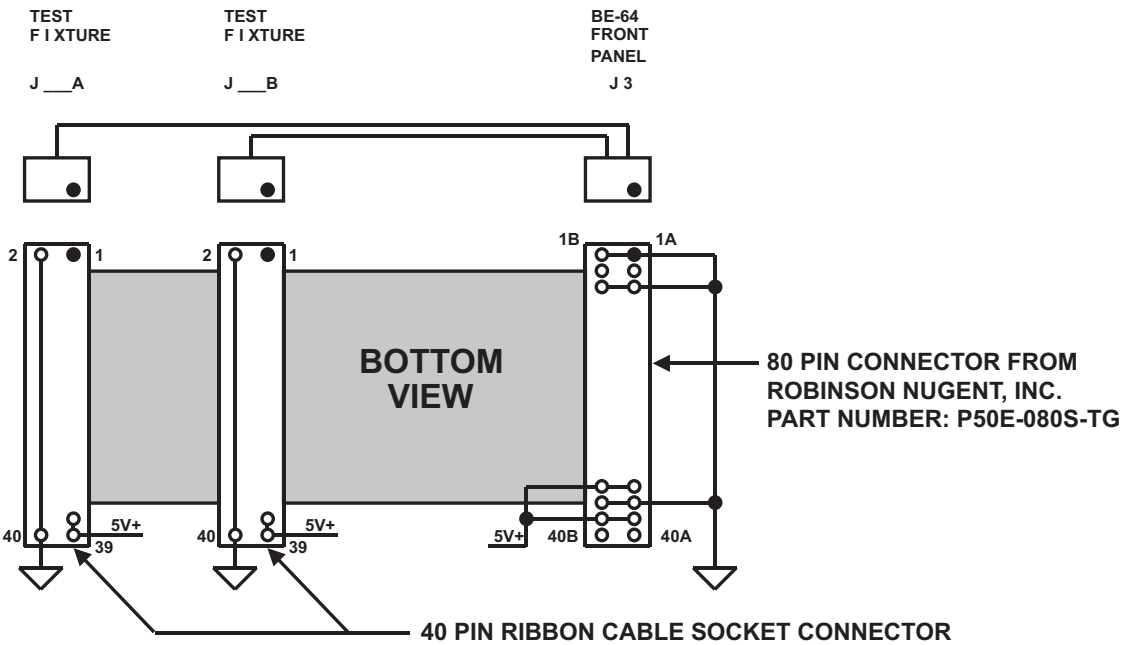


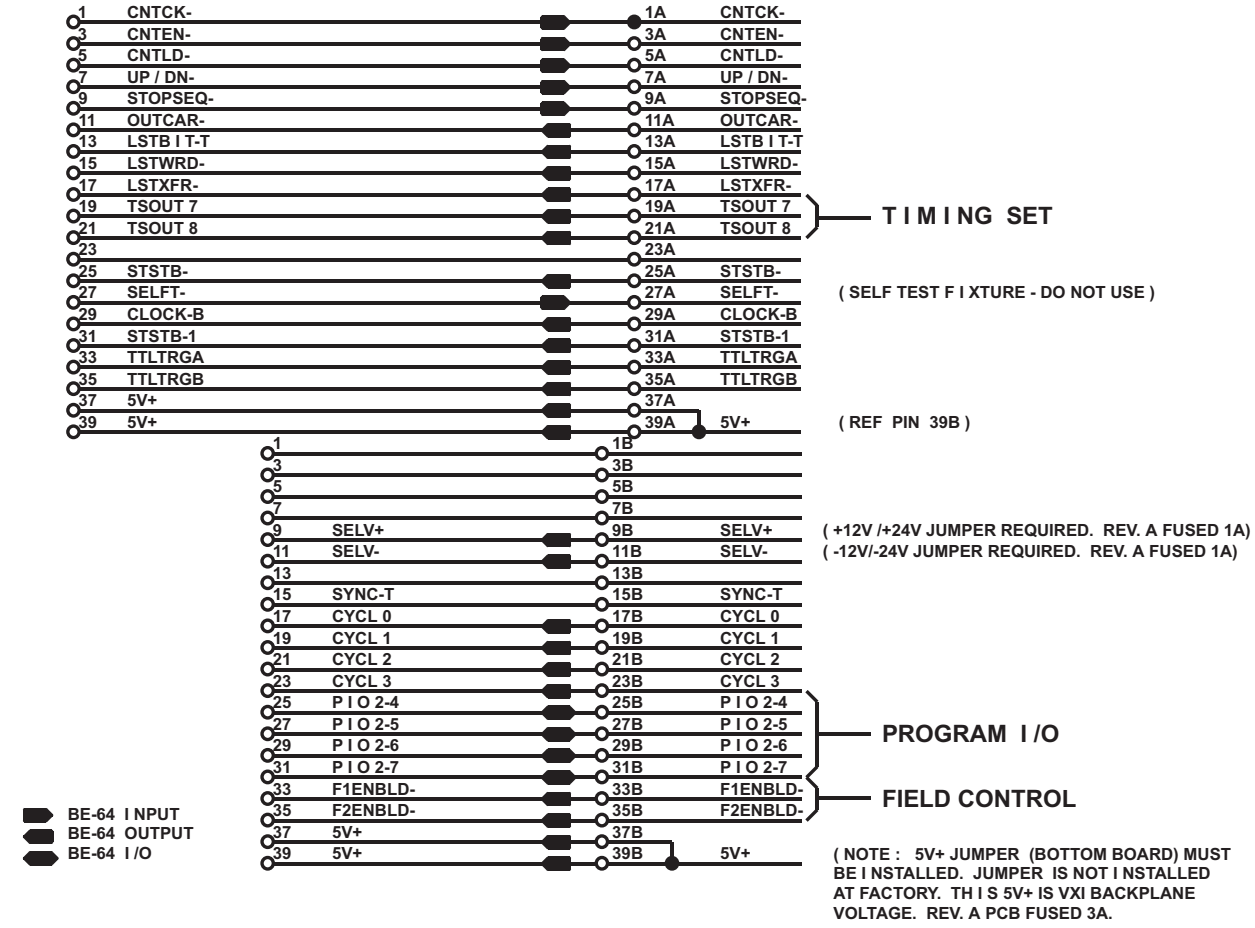
BE-64 I/O CABLE - J1



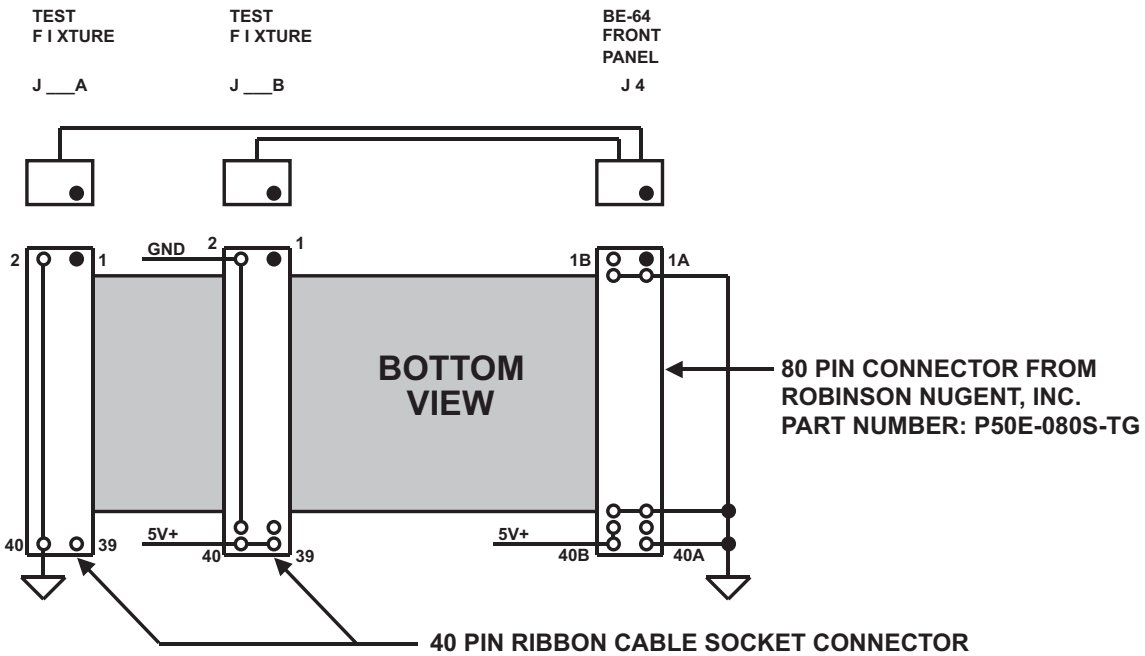


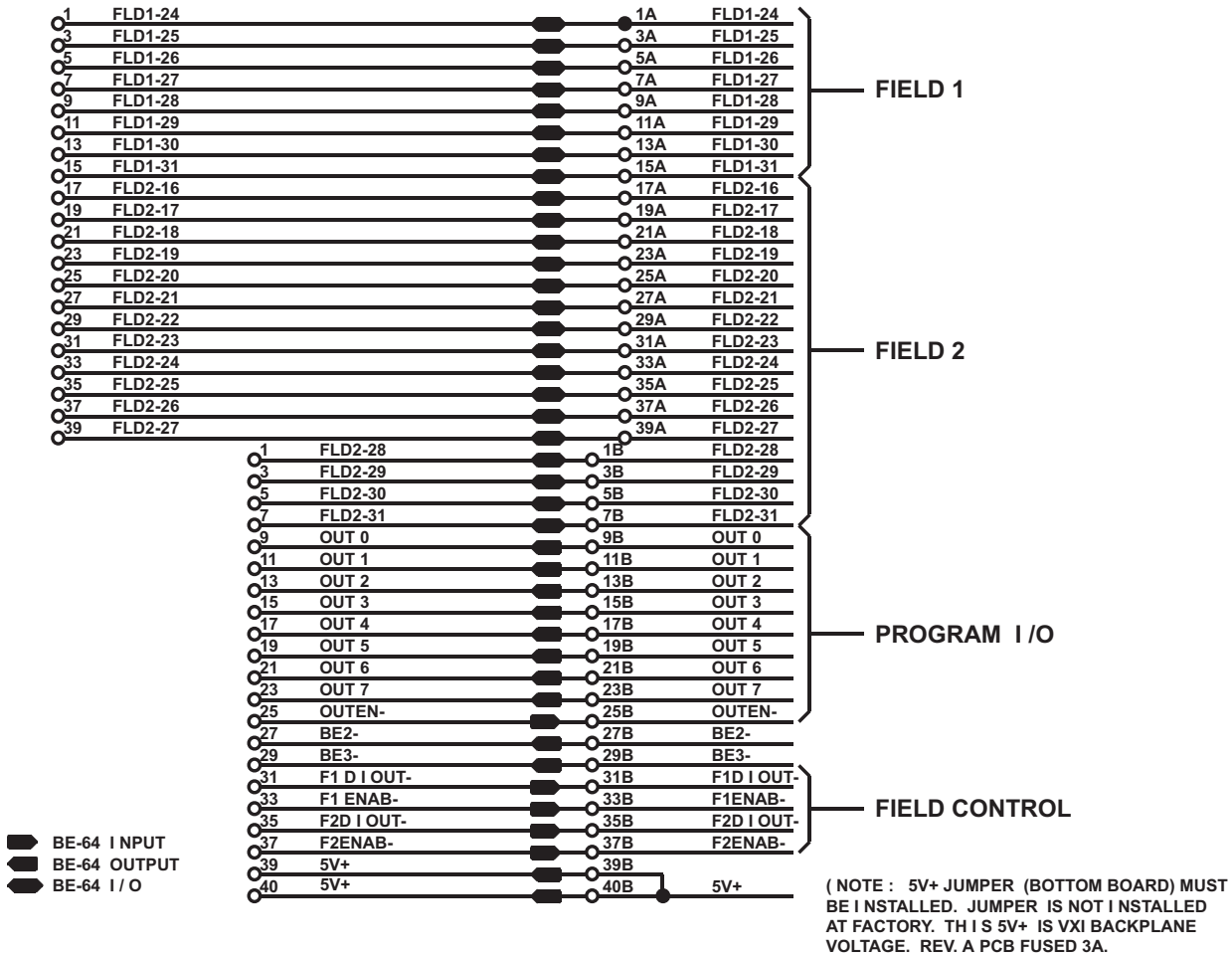
BE-64 I/O CABLE - J2





BE-64 I/O CABLE - J3





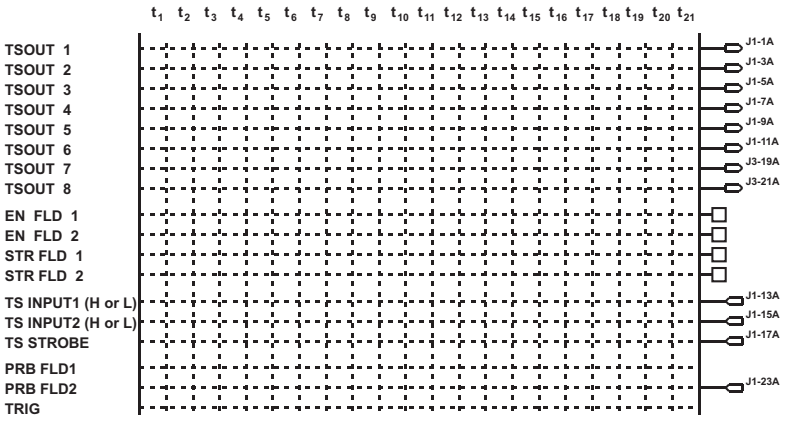
BE-64 I/O CABLE - J4

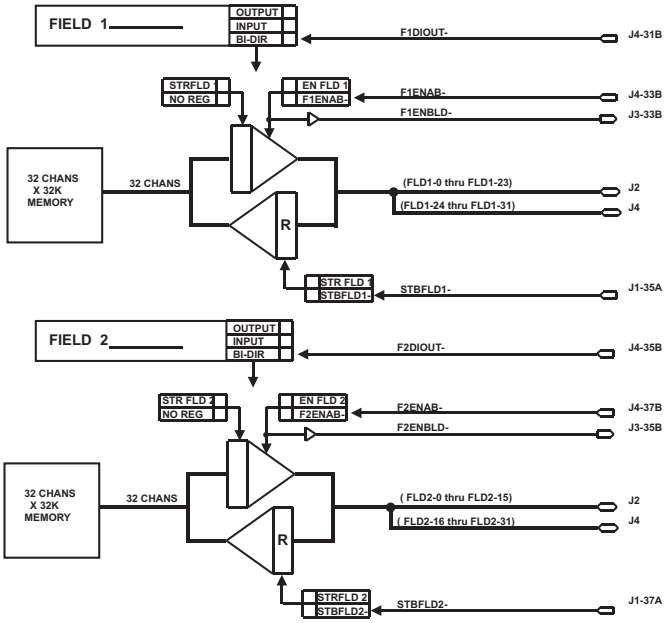
APPENDIX F WORKSHEETS

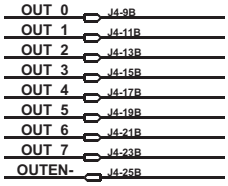
The following worksheets are included:

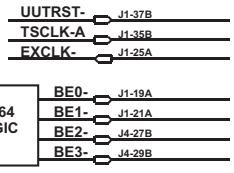
(a) Interface Simulation Worksheet
(b) Timing Set Worksheet.
(c) J1 Pinouts.
(d) J2 Pinouts.
(e) J3 Pinouts.
(f) J4 Pinouts.

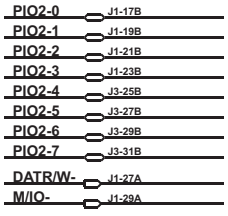
TIMING SET # ____ NAME _____ TIMEOUT ENABLED ☐ _____ CLOCKS DELAY _____ CLOCKS

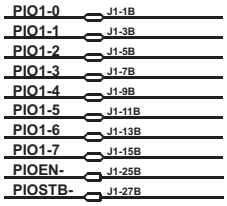












VXI MODEL BE-64 FUNCTION _____

DRAWN BY _____ DATE _____ DRAWING NO. _____

Timing Set Name_____**Sheet__of__**

TIMING SET WORKSHEET

CELL	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
TSOUT 1																				
BIT 0																				
TSOUT 2																				
BIT 1																				
TSOUT 3																				
BIT 2																				
TSOUT 4																				
BIT 3																				
TSOUT 5																				
BIT 4																				
TSOUT 6																				
BIT 5																				
TSOUT 7																				
BIT 6																				
TSOUT 8																				
BIT 7																				
BYTE LOW																				
EN FLD 1																				
BIT 8																				
EN FLD 2																				
BIT 9																				
STR FLD 1																				
BIT 10																				
STR FLD 2																				
BIT 11																				
PRB FLD 1																				
BIT 12																				
PRB FLD 2																				
BIT 13																				
TRIG																				
BIT 14																				
BIT 15 (unused)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BYTE HIGH																				
TS INPUT 1																				
TS INPUT 2																				
TS STROBE																				
DELAY _____ CLOCKS																				

BIN---HEX

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

User Connector Pin Designation	User Mnemonic	Talon Mnemonic	CONNECTOR B	CONNECTOR A	Talon Mnemonic	User Mnemonic	User Connector Pin Designation
		5V+	40B	40A	5V+		
		UUT5V+	39B	39A	UUTCLP-		
		GND	38B	38A	GND		
		UUTRST-	37B	37A	STBFLD2-		
		GND	36B	36A	GND		
		TSCLK-A	35B	35A	STBFLD1-		
		GND	34B	34A	GND		
		CYCLE-B	33B	33A	INTCYC-T		
		GND	32B	32A	GND		
		BUSACK-	31B	31A	BUSTST-T		
		GND	30B	30A	GND		
		BUSREQ-	29B	29A	MI/O-T		
		GND	28B	28A	GND		
		PIOSTB-	27B	27A	DATR/W-T		
		GND	26B	26A	GND		
		PIOEN-	25B	25A	EXCLK-		
		GND	24B	24A	GND		
		PIO2-3	23B	23A	PRBDAT-		
		GND	22B	22A	GND		
		PIO2-2	21B	21A	BE1-		
		GND	20B	20A	GND		
		PIO2-1	19B	19A	BE0-		
		GND	18B	18A	GND		
		PIO2-0	17B	17A	TSSTROBE		
		GND	16B	16A	GND		
		PIO1-7	15B	15A	TSINPUT2		
		GND	14B	14A	GND		
		PIO1-6	13B	13A	TSINPUT1		
		GND	12B	12A	GND		
		PIO1-5	11B	11A	TSOUT6		
		GND	10B	10A	GND		
		PIO1-4	9B	9A	TSOUT5		
		GND	8B	8A	GND		
		PIO1-3	7B	7A	TSOUT4		
		GND	6B	6A	GND		
		PIO1-2	5B	5A	TSOUT3		
		GND	4B	4A	GND		
		PIO1-1	3B	3A	TSOUT2		
		GND	2B	2A	GND		
		PIO1-0	1B	1A	TSOUT1		

DESIGNATES SIGNAL PAIRS

BE-64 I/O CONNECTOR J1 PINOUT

USER WORKSHEET

CONNECTOR B				CONNECTOR A			
User Connector Pin Designation	User Mnemonic	Talon Mnemonic			Talon Mnemonic	User Mnemonic	User Connector Pin Designation
		GND	40B	40A	GND		
		FLD2-15	39B	39A	FLD1-19		
		GND	38B	38A	GND		
		FLD2-14	37B	37A	FLD1-18		
		GND	36B	36A	GND		
		FLD2-13	35B	35A	FLD1-17		
		GND	34B	34A	GND		
		FLD2-12	33B	33A	FLD1-16		
		GND	32B	32A	GND		
		FLD2-11	31B	31A	FLD1-15		
		GND	30B	30A	GND		
		FLD2-10	29B	29A	FLD1-14		
		GND	28B	28A	GND		
		FLD2-9	27B	27A	FLD1-13		
		GND	26B	26A	GND		
		FLD2-8	25B	25A	FLD1-12		
		GND	24B	24A	GND		
		FLD2-7	23B	23A	FLD1-11		
		GND	22B	22A	GND		
		FLD2-6	21B	21A	FLD1-10		
		GND	20B	20A	GND		
		FLD2-5	19B	19A	FLD1-9		
		GND	18B	18A	GND		
		FLD2-4	17B	17A	FLD1-8		
		GND	16B	16A	GND		
		FLD2-3	15B	15A	FLD1-7		
		GND	14B	14A	GND		
		FLD2-2	13B	13A	FLD1-6		
		GND	12B	12A	GND		
		FLD2-1	11B	11A	FLD1-5		
		GND	10B	10A	GND		
		FLD2-0	9B	9A	FLD1-4		
		GND	8B	8A	GND		
		FLD1-23	7B	7A	FLD1-3		
		GND	6B	6A	GND		
		FLD1-22	5B	5A	FLD1-2		
		GND	4B	4A	GND		
		FLD1-21	3B	3A	FLD1-1		
		GND	2B	2A	GND		
		FLD1-20	1B	1A	FLD1-0		

DESIGNATES SIGNAL PAIRS

BE-64 I/O CONNECTOR J2 PINOUT

USER WORKSHEET

User Connector Pin Designation	User Mnemonic	Talon Mnemonic	CONNECTOR B	CONNECTOR A	Talon Mnemonic	User Mnemonic	User Connector Pin Designation
		GND	40B	40A	GND		
		5V+	39B	39A	5V+		
		GND	38B	38A	GND		
		5V+	37B	37A	5V+		
		GND	36B	36A	GND		
		F2ENBLD-	35B	35A	TTLTRGB		
		GND	34B	34A	GND		
		F1ENBLD-	33B	33A	TTLTRGA		
		GND	32B	32A	GND		
		PIO2-7	31B	31A	STSTB-1		
		GND	30B	30A	GND		
		PIO2-6	29B	29A	CLOCK-B		
		GND	28B	28A	GND		
		PIO2-5	27B	27A	SELFT-		
		GND	26B	26A	GND		
		PIO2-4	25B	25A	STSTB-		
		GND	24B	24A	GND		
		CYCL3	23B	23A			
		GND	22B	22A	GND		
		CYCL2	21B	21A	TSOUT8		
		GND	20B	20A	GND		
		CYCL1	19B	19A	TSOUT7		
		GND	18B	18A	GND		
		CYCL0	17B	17A	LSTXFR-		
		GND	16B	16A	GND		
		SYNC-T	15B	15A	LSTWRD-		
		GND	14B	14A	GND		
			13B	13A	LSTBIT-T		
		GND	12B	12A	GND		
		SELV-	11B	11A	OUTCAR-		
		GND	10B	10A	GND		
		SELV+	9B	9A	STOPSEQ-		
		GND	8B	8A	GND		
			7B	7A	UP/DN-		
		GND	6B	6A	GND		
			5B	5A	CNTLD-		
		GND	4B	4A	GND		
			3B	3A	CNTEN-		
		GND	2B	2A	GND		
			1B	1A	CNTCK-		

DESIGNATES SIGNAL PAIRS

BE-64 I/O CONNECTOR J3 PINOUT

USER WORKSHEET

User Connector Pin Designation	User Mnemonic	CONNECTOR B Talon Mnemonic		CONNECTOR A Talon Mnemonic		User Mnemonic	User Connector Pin Designation
		5V+	40B	40A	GND		
		5V+	39B	39A	FLD2-27		
		GND	38B	38A	GND		
		F2ENAB-	37B	37A	FLD2-26		
		GND	36B	36A	GND		
		F2DIOUT-	35B	35A	FLD2-25		
		GND	34B	34A	GND		
		F1ENAB-	33B	33A	FLD2-24		
		GND	32B	32A	GND		
		F1DIOUT-	31B	31A	FLD2-23		
		GND	30B	30A	GND		
		BE3-	29B	29A	FLD2-22		
		GND	28B	28A	GND		
		BE2-	27B	27A	FLD2-21		
		GND	26B	26A	GND		
		OUTEN-	25B	25A	FLD2-20		
		GND	24B	24A	GND		
		OUT7	23B	23A	FLD2-19		
		GND	22B	22A	GND		
		OUT6	21B	21A	FLD2-18		
		GND	20B	20A	GND		
		OUT5	19B	19A	FLD2-17		
		GND	18B	18A	GND		
		OUT4	17B	17A	FLD2-16		
		GND	16B	16A	GND		
		OUT3	15B	15A	FLD1-31		
		GND	14B	14A	GND		
		OUT2	13B	13A	FLD1-30		
		GND	12B	12A	GND		
		OUT1	11B	11A	FLD1-29		
		GND	10B	10A	GND		
		OUT0	9B	9A	FLD1-28		
		GND	8B	8A	GND		
		FLD2-31	7B	7A	FLD1-27		
		GND	6B	6A	GND		
		FLD2-30	5B	5A	FLD1-26		
		GND	4B	4A	GND		
		FLD2-29	3B	3A	FLD1-25		
		GND	2B	2A	GND		
		FLD2-28	1B	1A	FLD1-24		

DESIGNATES SIGNAL PAIRS

BE-64 I/O CONNECTOR J4 PINOUT

USER WORKSHEET

APPENDIX G SIGNAL DESCRIPTION

1 PROGRAM I/O

1.1 PIO DATA LINES

PIO1 (0-7): (8 lines) [J1-(1,3,5,7,9,11,13,15)B]: The PIO1 lines are set to be either input or output in an eight bit group. Each line can be either set or reset by the VXI controller when set to output mode, or may be tested for a high or low condition when in input mode. On power up or system reset, the PIO1 lines are set to the input mode. 74ALS652, 47W series terminated.

PIO2 (0-7): (8 lines) [J1-(17,19,21,23)B, J3-(25,27,29,31)B]: Same function as the PIO1 lines.

1.2 PIO CONTROL LINES

PIOEN- [J1-25B]: The PIOEN- input line enables the PIO1 and PIO2 lines when they are set to the OENabled mode (PIOEN- = Low = Enable, PIOEN- = High = Tristate). 3.3KW pull-up. This signal is inactive when PIO1 and PIO2 are in the OUTPUT mode.

PIOSTB- [J1-27B]: The PIOSTB- input line strobes the PIO1 and/or PIO2 input register when they are set to the input STRBed mode (strobe occurs on positive to negative edge). This signal is inactive when PIO1 and PIO2 are in the INPut mode.

2 COUNTER OUTPUT

2.1 OUT DATA LINES

OUT (0-7) (8 lines) [J4-(9,11,13,15,17,19,21,23)B]: The OUT 0-7 lines can be set to be either an output register (identical to the PIO lines when in output mode) or they can be set up to be an eight bit counter. When in the counter mode, the control lines are compatible with the field memory control lines (see Proprietary Control Signals, section 4.2 of this appendix). With proper connection the OUT 0-7 lines can be used to simulate the MSB's of an address bus. This allows memory testing of cards with 16 megabyte of RAM, with zero delay between continuous 16 MEG address boundaries. 74ALS245, 47W series terminated.

2.2 OUT CONTROL LINES

OUTEN- [J4-25B]: The OUTEN- input line enables the OUT 0 through 7 lines (OUTEN- = Low = Enable, OUTEN- = High = Tristate). 3.3KW pull-up

CNTCK- [J3-1A]: The CNTCK- input signal clocks the output counter when in counter mode (clock occurs on positive to negative edge). 3.3KW pull-up

CNTEN- [J3-3A]: The CNTEN- input signal enables the output counter to advance the count at the next clock (counter advances on CNTEN- = Low). 3.3KW pull-up

CNTLD- [J3-5A]: THE CNTLD- input signal enables the counter to load with the initial value at the next CNTEN- signal and CNTCK-clock signal. (counter loads when CNTLD- = Low). 3.3KW pull-up

UP/DN- [J3-7A]: The UP/DN- input signal defines the counter to be either an UP or DOWN counter (UP/DN- = Low = Down Count). 3.3KW pull-up

OUTCAR- [J3-11A]: The OUTCAR- output signal indicates the count value of hex FF (UP/DN- = 1) or hex OO (UP/DN- = 0) 74AS244, 47W series terminated.

3 TIMING SET SIGNALS

3.1 Timing Set Output Signals

TSOUT(1-8): (8 lines) [J1-(1,3,5,7,9,11)A, J3-(19,21)A]: The eight TSOUT output signals are general purpose programmable signals available for the UUT or test fixture inter-

face. They are programmed to a high or low level with 20 nsec resolution. 74AS244, 68W series terminated.

3.2 Timing Set Test Signals

TSINPUT1,2: [J1-(13,15)A]: The two TSINPUT input signals can be sampled by the timing set. They can be tested for either a logic “0" or a logic ”1" state.
If the tested condition is true, the timing set advances to the next state. If the tested condition is false, the timing set enters a “WAIT” state. It remains in the “WAIT” state until either the tested condition becomes true or until the programmable timeout function is detected.

The TSINPUT signals allow the timing set to “handshake” with the UUT.

TSSTROBE: [J1-17A]: The TSSTROBE input signal is similar in function to the TSINPUT signals with the exception that the timing set tests for the occurrence of either a positive or negative edge on the TSSTROBE input signal.
The edge detection logic is reset for the following conditions:

1) during an IDLE cycle.
2) immediately after a tested true condition is detected.
3) at the last cell position.

Therefore, an edge can be detected after the start of the timing set and multiple edges can be detected.
The detection of a positive edge does not affect the negative edge logic, and vice versa.
TSSTROBE can also be jumpered so that an internal signal, identical to TTLTRGA, will be the source.

3.3 Timing Set Field Control Signals

EN FLD1: The EN FLD1 signal programmed in the timing set is not routed to the front panel I/O lines.
When FLD1 output control is set to internal and the field direction is output, the EN FLD1 signal forces FLD1 from tri-state to an active state.

STR FLD1: The STR FLD1 signal, programmed in the timing set, is not routed to the front panel I/O lines.
When FLD1 output control is set to internal and the field direction is output, the STR FLD1 signal strobes the data from the 32K x 32 bit memory into the output register. The data transfer occurs on the positive to negative transition of the STR FLD1 signal.
When FLD1 input strobe is set to internal and the field direction is input, the STR FLD1 signal strobes the data from FLD1 into the input field register. The data transfer occurs on the positive to negative transition of the STR FLD1 signal.

EN FLD2: The EN FLD2 signal programmed in the timing set is not routed to the front panel I/O lines.
When FLD2 output control is set to internal and the field direction is output, the EN FLD2 signal forces FLD2 channels from tri-state to an active state.

STR FLD2: The STR FLD2 signal, programmed in the timing set, is not routed to the front panel I/O lines.
When FLD2 output control is set to internal and the field direction is output, the STR FLD2 signal strobes the data from the 32K x 32 bit memory into the output register. The data transfer occurs on the positive to negative transition of the STR FLD2 signal.
When FLD2 input strobe is set to internal and the field direction is input, the STROBE FIELD2 signal strobes the data from FLD2 into the input field register. The data transfer occurs on the positive to negative transition of the STROBE FIELD2 signal.

3.4 Timing Set Probe Signals

PRB FLD1: The PRB FLD1 signal generates a pulse compatible with Talon’s VXI signature/logic analyzer probe. The user should program this signal within the enable time of the

EN FLD1 signal. The PRB FLD1 signal can be routed to the TTLTRG signals on the VXI bus.

PRB FLD2: The PRB FLD2 signal generates a pulse compatible with Talon's VXI signature/logic analyzer probe. The user should program this signal within the enable time of the EN FLD2 signal. The PROBE FLD2 signal can be routed to the TTLTRG signals on the VXI bus.

TRIG: The TRIG signal is gated with the memory table sync signal. This combinatorial signal, compatible with Talon's VXI signature/logic analyzer probe, generates the trigger signal used by the logic analyzer function. The trigger function occurs at the detection of a logic "0" on the TRIG signal. This combinatorial signal is available as SYNC-T and is also available on the front panel BNC (SYNC). This signal can be routed to the TTLTRG signals on the VXI bus.
To ensure a single sync pulse, the TRIG signal should never be programmed with a low in the first cell and a high in the last cell. If a sync pulse at the beginning of cell 1 is desired, then it is recommended to program TRIG low for the entire timing set.

3.5 Timing Set Misc Control Signals

CYCL 0,1,2,3 (4 lines) [J3-(17,19,21,23)B]: The CYCL 0,1,2,3 output signals are a binary value defining the present timing set. Hex zero indicates the idle timing set. PAL20R4-7, 47W series terminated.

TIMING SET	TIMING SET NAME	CYCL3	CYCL2	CYCL1	CYCL0
0	IDLE	0	0	0	0
1	WRITE_MEM	0	0	0	1
2	WRITE_IO	0	0	1	0
3	READ_MEM	0	0	1	1
4	READ_IO	0	1	0	0
5	INT_ACK	0	1	0	1
6	BUS_TEST	0	1	1	0
7	user defined	0	1	1	1
8	user defined	1	0	0	0
9	user defined	1	0	0	1
10	user defined	1	0	1	0
11	user defined	1	0	1	1
12	user defined	1	1	0	0
13	user defined	1	1	0	1
14	user defined	1	1	1	0
15	user defined	1	1	1	1

CYCLE-B (1 line) [J1-33B]: The CYCLE-B output signal, when low, indicates a non-idle cycle is being executed. 74ALS244, 47W series terminated.

DATR/W-T (1 line) [J1-27A]: A low on the DATR/W-T output signal indicates an active timing set #1 (WRITE MEMORY) or timing set #2 (WRITE I/O). This signal may be used to generate the R/W- control line of a UUT interface. PAL20L8B-2, 47W series terminated.

MI/O-T (1 line) [J1-29A]: A low on the MI/O-T output signal indicates an active timing set #2 (WRITE I/O) or timing set #4 (READ I/O). This signal may be used to generate the memory/I/O control signal for Intel microprocessors. PAL20L8B-2, 47W series terminated.

INTCYC-T (1 line) [J1-33A]: A low on the INTCYC-T output signal indicates an active timing set #5 (INTERRUPT CYCLE). This signal indicates an active interrupt acknowledge cycle is being executed. PAL20L8B-2, 47W series terminated.

BUSTST-T (1 line) [J1-31A]: A low on the BUSTST-T output signal indicates an active timing set #6 (BUS TEST). This signal may be used to indicate to the UUT that a bus test cycle is being executed. PAL20L8B-2, 47W series terminated.

TSCLK-A (1 line) [J1-35B]: The TSCLK-A output signal is the present clock driving the timing set. Timing set signals are activated at the high to low transition of this signal. 74AS244, 47W series terminated.

3.6 Bus Arbitration Signals

BUSREQ- (1 line) [J1-29B]: The BUSREQ- input signal requests the bus emulator to enter a tri-state condition with an active Low state. The bus emulator will enter the tri-state condition when BUSREQ- is active (and enabled by a VXI command) and immediately after the last bit of the present timing set cycle. 3.3KW pull-up.

BUSACK- (1 line) [J1-31B]: The BUSACK- output signal goes active Low indicating the bus emulator has acknowledged the present bus request (BUSREQ-). The bus emulator enters a hold state and appropriate output signals are tri-stated. BUSACK- goes not true (high) immediately after the BUSREQ- goes not true (high) and after syncing up to the timing set clock. PAL16R4-7 no termination.

UUTCLP- (1 line) [J1-39A]: A low on the UUTCLP- input signal indicates a test connector has been clipped over the UUT microprocessor. 3.3KW pull-up.

3.7 Timing Set External Clock

EXCLK- (1 line) [J1-25A]: The EXCLK- input signal is the user defined external clock which may be selected to drive the timing sets. The maximum frequency is 50MHz. The timing set signals are activated at the high to low transition of the EXCLK- clock. 180W pull-up, 390W pull-down.

3.8 Timing Set SYNC

SYNC-T (1 line) [J3-15B]: The SYNC-T output signal is the combination of the timing set TRIG signal and the memory table sync signal. See TRIG in section 3.4 of this appendix. 74AS1034, 47W series terminated.

SYNC (1 line) [Front Panel BNC]: Same signal as SYNC-T. This signal is intended to trigger a scope. 74ALS244A, 47W series terminated.

4 PROPRIETARY CONTROL SIGNALS

The BE-64 board has been designed to simulate any digital interface and in particular bus structured interfaces. To effectively simulate the latest Intel and Motorola microprocessors at clock speeds of 50 Mhz, several signals are required which Talon considers proprietary.

The BE-64 can drive a parallel to serial converter card which achieves data rates to 100 Mhz. All the necessary control signals for the conversion card reside in these proprietary control signals.

In addition, several control lines from the field memory address logic are output which are compatible with the OUT 0-7 counter function.

4.1 External Sequence Stop Control

STOPSEQ- (1 line) [J3-9A]: An active low pulse, greater than 50 ns, on input signal STOPSEQ- will cause a sequence to stop. The VXI controller can verify the sequence properly stopped. Use with caution if the user is uncertain where in the sequence STOPSEQ- is activated. 3.3KW pull-up.

4.2 Sequence State Output Signals

LSTBIT-T (1 line) [J3-13A]: The LSTBIT-T output signal is active low during the last cell of the current timing set. Note: Active during the idle timing set. 74AS1034, 68W series terminated.

LSTWRD- (1 line) [J3-15A]: The LSTWRD- output signal is active low during the last word of the current table. LSTWRD- stays low for as long as one execution of the timing set. Note: Driven high during the idle timing set. PAL20R8-7, no termination

LSTXFR- (1 line) [J3-17A]: The LSTXFR- output signal is active low during the last word of the last loop of a table. LSTXFR- stays low during the entire timing set. Note: Driven high during the idle timing set. PAL20R8-7, no termination.

5 FIELD CONTROL SIGNALS

5.1 Field Signals

FLD1 (0-31) (32 lines) [J2(1,3,...,39)A, J2(1,3,5,7)B, J4(1,3,...,15)A ALL ODD]: The FLD1-0 through FLD1-31 signals are bi-directional signals with 32K bits behind each channel. The maximum data rate for this field is 25 Mhz. This field may be used to simulate an address bus of the UUT. However, it is general purpose and may be used for any UUT function. 74AS652, 47W series terminated.

F1DIOUT- (1 line) [J4-31B]: The F1DIOUT- input signal defines the direction of FLD1 when FLD1 is set in external mode. FLD1 is set to the OUTPUT direction when F1DIOUT- = Low. No termination.

F1ENAB- (1 line) [J4-33B]: The F1ENAB- input signal enables FLD1 data when FLD1 output control is set in the external mode. FLD1 data is enabled when F1ENAB- = Low. FLD1 data is enabled only if the direction is set to be output. No termination.

STBFLD1- (1 line) [J1-35A]: When input strobe is set to external, the STBFLD1- input signal strobes FLD1 data into FLD1 input register. When output control is set to external and output register is on, the STBFLD1- signal strobes FLD1 output data into the output register. The data is strobed on the high to low transition of the STBFLD1- signal. No termination

F1ENBLD- (1 line) [J3-33B]: The F1ENBLD- output signal is active low whenever FLD1 is in the output mode and the enable signal is true. 74ALS244, 47W series terminated.

FLD2 (0-31) (32 lines) [J2(9,11,...,39)B, J4(17,19,...39)A, J4(1,3,5,7)B ALL ODD]: The FLD2-0 through FLD2-31 signals are bi-directional signals with 32K bits behind each channel. The maximum data rate for this field is 25 Mhz. This field may be used to simulate a data bus of the UUT. However, it is general purpose and may be used for any UUT function. 74AS244, 68W series terminated.

F2DIOUT- (1 line) [J4-35B]: The F2DIOUT- input signal defines the direction of FLD2 when FLD2 is set in external mode. FLD2 is set to the OUTPUT direction when F2DIOUT- = Low. No termination.

F2ENAB- (1 line) [J4-37B]: The F2ENAB- input signal enables FLD2 data when FLD2 output control is set to external. FLD2 data is enabled when F2ENAB- = Low. FLD2 data is only enabled if the direction is set to be output. No termination.

STBFLD2- (1 line) [J1-37A]: When input strobe is set to external, the STBFLD2- input signal strobes FIELD 2 data into FLD2 input register. When output control is set to external and output register is on, the STBFLD2- signal strobes FLD2 output data into the output register. The data is strobed on the high to low transition of the STBFLD2- signal. No termination.

F2ENBLD- (1 line) [J3-35B]: The F2ENBLD- output signal is active low whenever FIELD 2 (UUT DATA BUS) is in the output mode and the enable signal is true. 74ALS244, 47W series terminated.

5.2 Byte Enable Signals

BEO-,1-,2-,3- (4 lines) [J1-(19,21)A, J4-(27,29)B]: Most bus and microprocessor structured interfaces have separate control lines to indicate the present size of the data bus (byte, word, or longword). Moreover, byte and word transfers may be defined to be in the LSB or MSB positions. The byte enable output signals are defined by respective VXI compatible commands. The active condition of the BE0- through BE3- is defined by the present transfer type (byte, word, or longword), as well as Bit 0 and Bit 1 of Field 1 (which represents A0 and A1 of the address bus). These signals are further qualified by timing set signal #8 (TSOUT8). 74AS258, 47W series terminated.

5.3 Bus Emulator Output Clock

CLOCK-B (1 line) [J3-29A]: The CLOCK-B output signal is the currently selected clock rate for the bus emulator. 74AS244, 47W series terminated.

6 VXI TTL TRIGGER SIGNALS

6.1 TTL Trigger Source Selection

TTLTRGA (1 line) [J3-33A]: The TTLTRGA output signal is user defined to select any one of the TTLTRG (0-7) lines as the source. TTLTRGA can also be defined to be in a tri-state condition. PAL22P10AL, no termination

TTLTRGB (1 line) [J3-35A]: The TTLTRGB output signal is similar in function to the TTLTRGA signal. TTLTRGB can also be jumpered to TSINPUT2. This jumper is not normally installed. PAL22P10AL, no termination

6.2 TTL Trigger Output Selection

PRBDAT- (1 line) [J1-23A]: The PRBDAT- input signal is a user defined probe signal. PRBDAT- can be routed to the VXI trigger lines TTLTRG (4-7). No termination.

7 SELF TEST FIXTURE SIGNALS

7.1 Self Test Input Signal

SELFT- (1 line) [J3-27A]: The SELFT- input signal is used by the self test fixture. Do not use.

7.2 Self Test Output Signals

STSTB- (1 line) [J3-25A]: The STSTB- output signal is used to clock the self test fixture. Not to be used by the user.

STSTB-1 (1 line) [J3-31A]: Same function as STSTB-.

8 MISCELLANEOUS SIGNALS

8.1 Reset Signal

UUTRST- (1 line) [J1-37B]: A low on the UUTRST- output signal is used to reset the UUT. UUTRST- is low for a minimum of 50 milliseconds and is asynchronous. 3.3KW pull-up.

8.2 UUT Power On Signal

UUT5V+ (1 line) [J1-39B]: A TTL logic “0" on input line UUT5V+ indicates there is no power on the UUT. A TTL logic ”1" on UUT5V+ indicates there is power on the UUT. UUT5V+ is pulled low through a 10K resistor.

8.3 Power Lines From VXI Backplane

5V+ (8 lines) [J1-40A,J1-40B, J3-(39,37)A, J3-(39,37)B, J4-(40,39)B]: Output lines connected via jumpers to the VXI 5V backplane (fused 3A). Use with caution.

SELV+ (1 line) [J3-9B]: The SELV+ output line is user defined to connect to either the +12V or +24V of the VXI backplane via jumpers on the bottom board (fused 1/4A). Use with caution.

SELV- (1 line) [J3-11B]: The SELV- output line is user defined to connect to either the -12V or -24V of the VXI backplane via jumpers on the bottom board (fused 1/4A). Use with caution.

APPENDIX H IMPROVING BE-64 ACCESS

1 VXI COMMUNICATION LAYERS

The following figure describes the different communication layers that can be used to program the BE-64's field and timing memories.

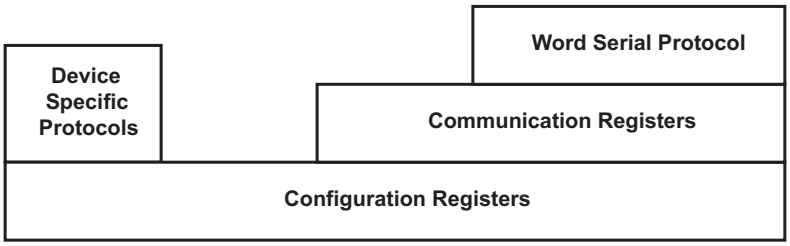


FIGURE H-1 VXI COMMUNICATION LAYERS

The lowest level of communication of any VXIbus device is the VME bus protocol. The VME bus protocol is defined by generic bus cycles (READ, WRITE, etc.) that are controlled by hardware on each device.

1.1 Register Based Devices

The VXIbus protocol defines a specific addressing method as well as a standard set of registers called the Configuration Registers. VXIbus devices who only support these registers are called "Register Based Devices". Data/commands are sent to these registers via VMEbus WRITE cycles. Data/results are received from these registers via VMEbus READ cycles. Register based devices can be programmed quickly because their control functions are closer to the hardware. Register based devices generally require a custom driver because there is no standard set of rules governing programming or setup.

1.2 Message Based Devices

An optional device class that the BE-64 supports is called "Message Based". Message based devices define an additional set of registers called the Communication Registers. As with the Configuration Registers data is sent/received to these registers via VMEbus WRITE/READ cycles. Message Based devices are required to support a minimum set of rules and communication capability. Additionally, several manufacturers have developed a command language called SCPI (Standard Command for Programmable Instruments) in order to standardize commands for instruments with similar functionality. Message based devices can be programmed with drivers supplied with the Slot 0 resource manager, however, Message Based device communication is performed a character at a time and involves several protocols as described below.

2 THE WORD SERIAL PROTOCOL

All Message Based device must support the Word Serial protocol. The Word Serial protocol defines a set of commands (WS Commands) and device functionality. Two of the message based communication registers are of particular interest, the Data Low and Response registers. These registers are used to send/receive WS commands and data.

The Response Register contains two bits, Read Ready (RRDY) and Write Ready (WRDY), that handshakes WS commands/responses to the Data Low register. The Response Register also contains two additional bits, Data Input Ready (DIR) and Data Output Ready (DOR), that are used to handshake message strings to/from the BE-64's command parser buffer via the Byte Transfer protocol.

The following figure illustrates the processing involved with a Slot 0 resource manager sending an EXEC command to a BE-64.

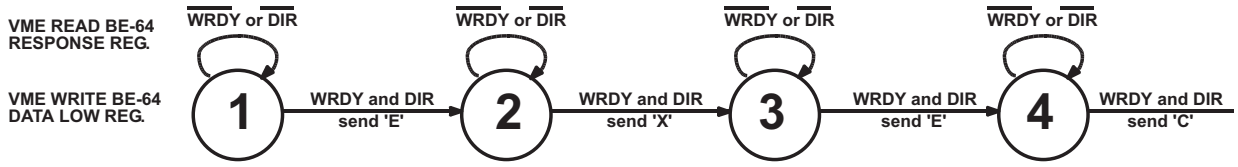


FIGURE H-2 BYTE TRANSFER PROTOCOL EXAMPLE

Step 1 Resource manager reads the BE-64's RESPONSE register until both WRDY and DIR are both true. When WRDY and DIR are both true, the resource manager sends the 'E' to the BE-64 by sending a Byte Available Word Serial command to the DATA LOW register.

Step 2 Resource manager reads the BE-64's RESPONSE register until both WRDY and DIR are both true. When WRDY and DIR are both true, the resource manager sends the 'X' to the BE-64 by sending a Byte Available Word Serial command to the DATA LOW register.

Step 3 Resource manager reads the BE-64's RESPONSE register until both WRDY and DIR are both true. When WRDY and DIR are both true, the resource manager sends the 'E' to the BE-64 by sending a Byte Available Word Serial command to the DATA LOW register.

Step 4 Resource manager reads the BE-64's RESPONSE register until both WRDY and DIR are both true. When WRDY and DIR are both true, the resource manager sends the 'C' to the BE-64 by sending a Byte Available Word Serial command to the DATA LOW register.

2.1 The Word Serial Bottleneck

As can be seen in the figure above there are two layers of protocol as well as a command parser when sending Word Serial messages. This overhead averages to about 2 milliseconds per eight bit byte for the BE-64. Binary block data transfers have a 500 microsecond overhead because they bypass the command parser.

Talon recognized the possible bottleneck of data that could occur with the Word Serial Protocol. Therefore the Field and Timing memory were placed within the VXIbus A24 address space so that they could be programmed without using the Word Serial Protocol. Using the A24 memory, sixteen bit words can be read/written approximately every 1 microsecond using direct VME bus READ and WRITE cycles.

3 SPEEDING UP TABLE TRANSFERS.

Using the preceding numbers above, the following rules can be used to speed up data throughput to/from the BE-64.

1. Program the field memory using the A24 memory. 32K 64 bit words works out to 128K 16 bit words. 128K words X 1 microsecond/word = 128 milliseconds (best case).
2. Use the fill functions to define the data. The BE-64 can program the field memory faster than the SCPI commands. 32K field increment = ~2 seconds.
3. Program the fields separately if one field can be filled using a fill pattern or if one field is set to input.
4. Use the DATA commands instead of the WORD commands. The DATA command requires fewer bytes and bypasses the parser for the data block. TABL:DATA using a 32K table = ~ 256K bytes X 500 microseconds/byte = ~128 seconds (2 min.). TABL:WORD using a 32K table = ~ 1.4Mbytes X 2 millisecond/byte = ~2,800 seconds (46 min).
5. Use the CALC:CRC command to verify large tables results.