HP E1413C

High-Speed 64-Channel Scanning A/D Converter

E1413A

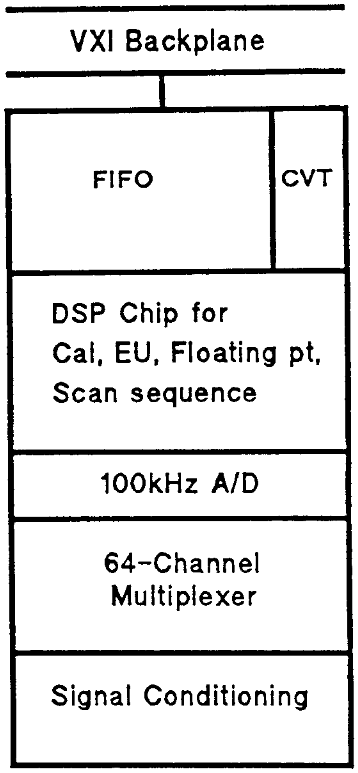
64 channel, 100 kHz

Scanning A/D

VXI Systems Division
Technical Support:TCevh:101292:1413_00

HEWLETT PACKARD

64-Channel Scanning A/D Subsystem



- Throughput
  - 100kHz at 16 bits
  - > 1.5kHz/channel
- Measurement Types
  - Temperature
  - DC Volts
  - Strain
  - Resistance
- Engineering Unit Conversion at full speed
- Plug-on Signal Conditioning Modules
- On-board Calibration Source
- 64K Readings RAM
  - FIFO
  - Current Value Table
- Scan List Management (up to 4)
- Tare Calibration Provided

VXI Systems Division
Technical Support:Tcevh:102492:1413_s01

HEWLETT PACKARD

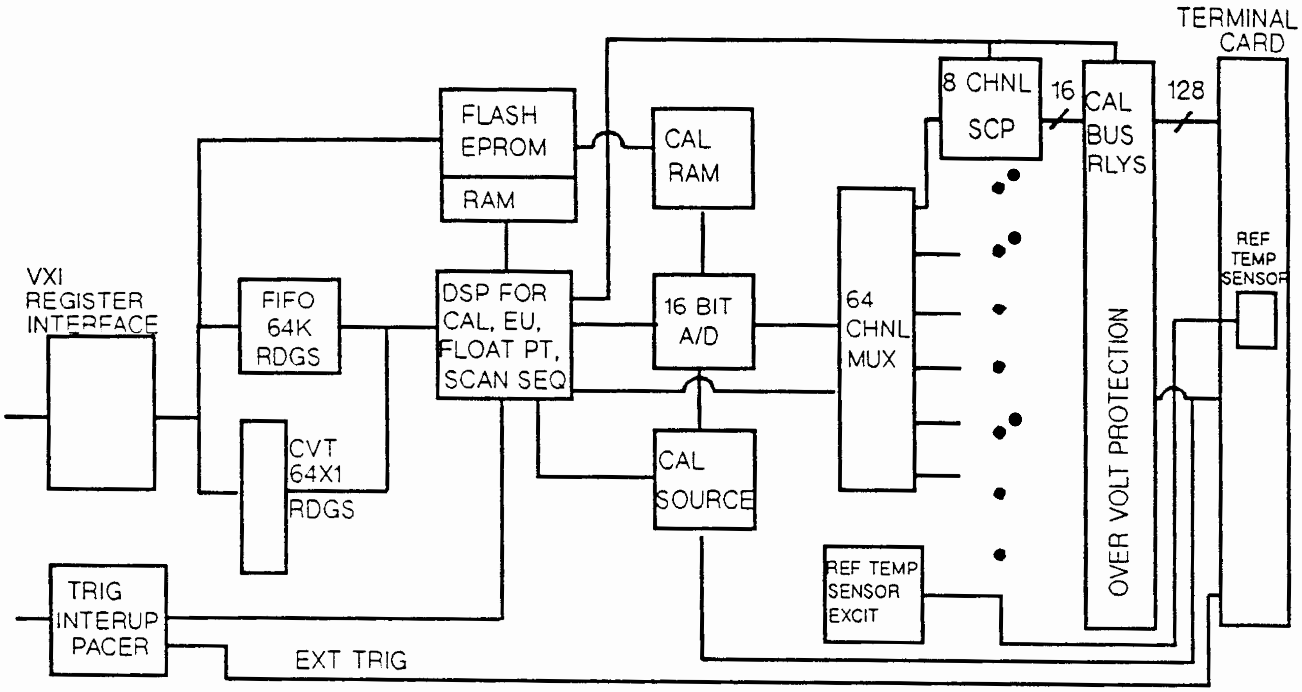
E1413A Background - Target Market - HiDATT System

Many of you with customers in the turbine and piston engine test and wind tunnel test monitoring businesses already know all about the E1413A (Mirrors) A/D. We have asked virtually everyone in these businesses exactly what they want and need in a data acquisition system and that is what HiDATT is intended to be. The E1413A is the A/D front end and it exceeds the general purpose, multi-channel requirements of the target market customers. It is a port of the HP 3852A's 16 bit, 100 kHz A/D with a 64 channel FET mux on-board and considerable additional functionality in a smaller space.

The E1413A is an extended device with registers in both A16 and A24 address space. It is designed for high speed continuous data acquisition to disk with multiple modules in a VXI card cage. It does IEEE floating point number conversion and engineering unit conversion of data at full speed. The Current Value Table (CVT) is a set of 64 channel data registers in A24 space that can be rapidly accessed to update displays, PID loops, etc. 8 channel signal conditioning plug in cards allow mixed signal measurements at the full 100 kHz speed.

Both SCPI and CSCPI drivers are provided, and direct high speed programming of the register interface can also be done.

HP E1413A 64-Channel Scanning A/D



VXI Systems Division
Technical Support:TCevh:102492:1413_s02

HEWLETT PACKARD

E1413A Features & Performance

The major contributions of the E1413A are:

1. Per channel signal conditioning allowing mixed functions, volts, temperature, strain, and resistance at speed. Pressure will be taken care of by the E1414A, PSI pressure scanner interface, which is a modified version of the E1413A, designed to directly interface with PSI Inc., 8400 series pressure scanners.

2. Amplification and Filter per channel.

3. 64 FET channels per module on board, and multiple module capability in VXI.

4. 32 bit IEEE Floating Point Data Format at Speed

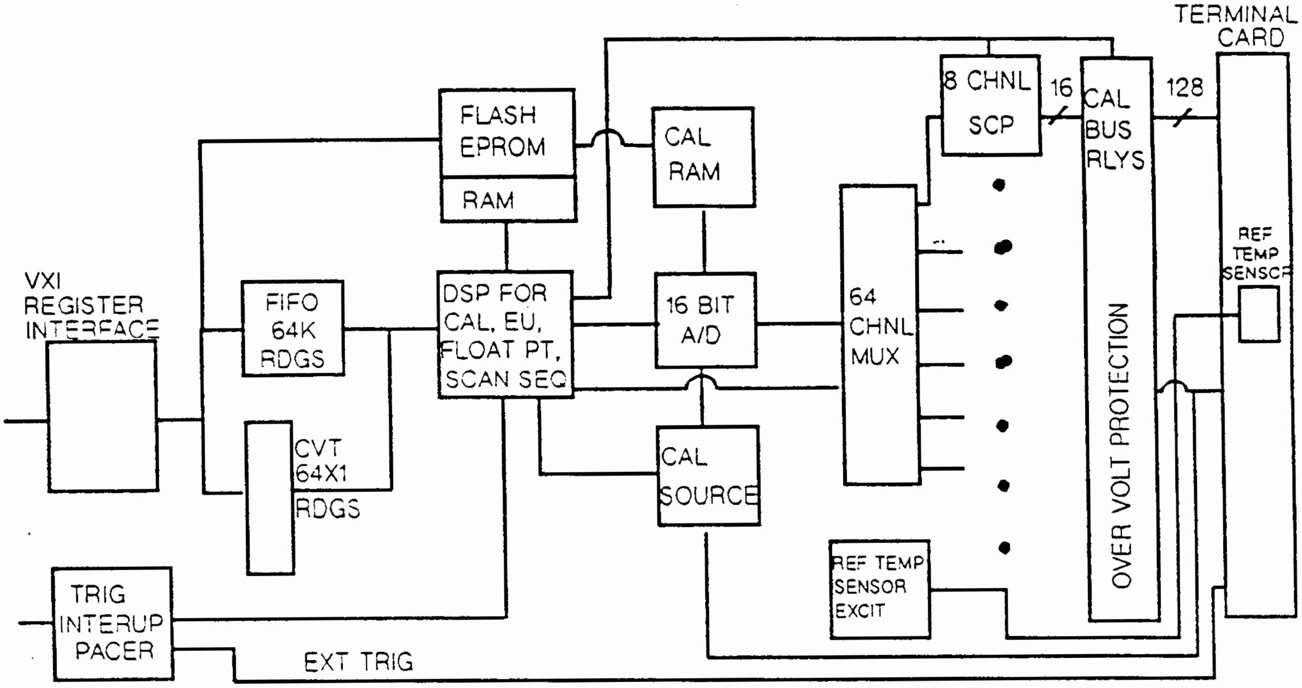
5. Engineering Unit (EU) Conversion at Speed

6. Up to 4 Scan Lists with different scan rates

7. 64 Kreading FIFO in A16 space and 64 register Current Value Table in A24. This allows you to continuously log all channels to disk while also sending all or a selected group of channels to a display, or to update PID loops, etc. The FIFO can be operated in Circular Buffer or Block mode.

8. Autoranging at speed (20 bit settling at 100 kHz)

HP E1413A 64-Channel Scanning A/D



VXI Systems Division
Technical Support:TCevh:102492:1413_s02

HEWLETT PACKARD

More E1413A Features & Performance

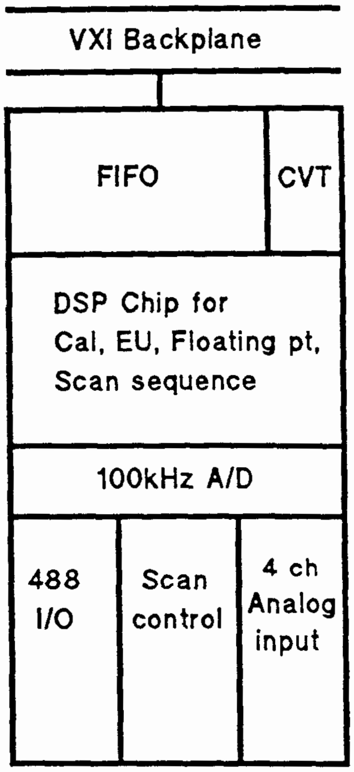
9. Individual Channel Gain & Offset Correction at Speed.

10. On Board, Per Channel Calibration Bus & Source. You can measure the internal calibration source at the terminal block. Store a calibration correction constant in the E1413A RAM, and then programatically calibrate all channels. This procedure stores offset and gain correction coefficients for each channel.

11. Programmable Open Channel Detection

12. Flash ROM for fast easy firmware enhancements

Pressure Interface to PSI Model 8400



- Pressure data in VXI
- Throughput
  - 50kHz at 16 bits
  - 512 channels
- On-board calibration routines
  - 488 control of pcu's
  - cal consistants in RAM
- PSI scanner sequencing using PSI channel numbering scheme
- Real-time EU conversion

VXI Systems Division
Technical Support:TCevh:102492:1413_s03

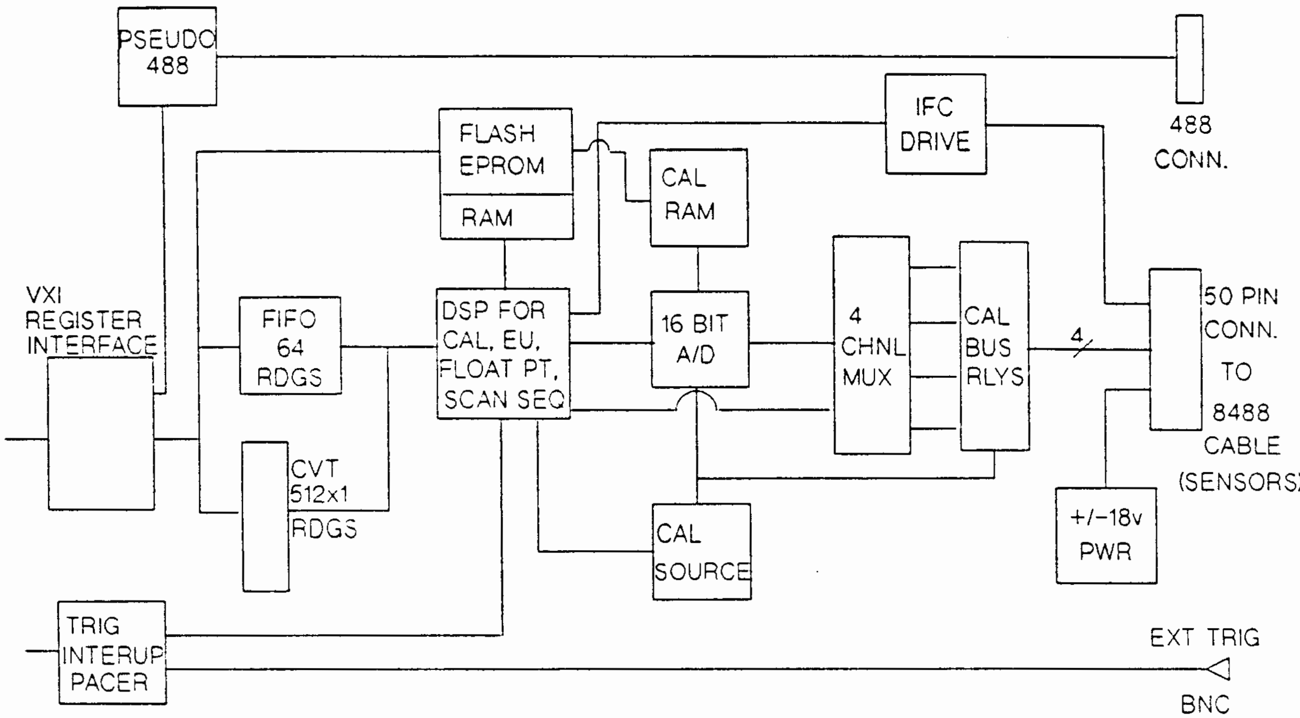
HEWLETT PACKARD

E1414A PSI Interface

Turbine and piston engine test and wind tunnel monitoring all require many channels of pressure measurements. The industry standards for these are PSI, Inc. and Scanivalve systems. Both of these companies use solid state pressure transducers for each channel and have FET multiplexers and an A/Ds in proprietary non-VXI systems. The E1414A provides a superior A/D and a VXI interface for the PSI 8400 series pressure scanners. Most importantly, because it is a major competitive advantage, the E1414A provides a calibration system interface for the pressure measurement system.

Kinetic Systems also provides a VXI interface for both PSI and Scanivalve, but they do not provide a means of calibrating the pressure system. This is a big deal because the solid state pressure transducers are very temperature sensitive and require frequent calibration. Customers are left on their own for calibration with the Kinetic Systems interface. A major part of the E1414A firmware development effort went into the calibration system. HP also suggested some changes to the PSI hardware that reduced system noise.

HP E1414A Pressure Scanner Interface



VXI Systems Division
Technical Support:TCevh:102492:1413_s04

HEWLETT PACKARD

E1414A PSI Interface Features

The block diagram above illustrates the modifications made on the E1414A to interface with the PSI 8400 series pressure scanners. The four FET channels on the E1414A allow each E1414A to support up to 512 pressure channels using PSI's parallel addressing mode (PAM). This channel sequencing mode pre settles the PSI sensors on one of 4 input channels while measurements are being made on another of the 4 channels. This is required for the full speed, 50 kHz scanning speed of the E1414A.

The other E1414A features are:

* Fixed 5 volt range and No SCPs

* Volts and Pressure functions only

* Individual calibration of the 4 E1414A FET channels

* Individual pressure channel sensor calibration at speed for up to 512 channels.

* Overvoltage protection (opens inputs at approximately +/- 22 volts)

* Switchable 5 kHz Low Pass Filter (same as E1413A)

* On board +/- 18 volt power supply for PSI 8400 scanners (uses PSI 8488 cable.

* 15 line digital interface to PSI scanners (also through PSI 8488 cable)

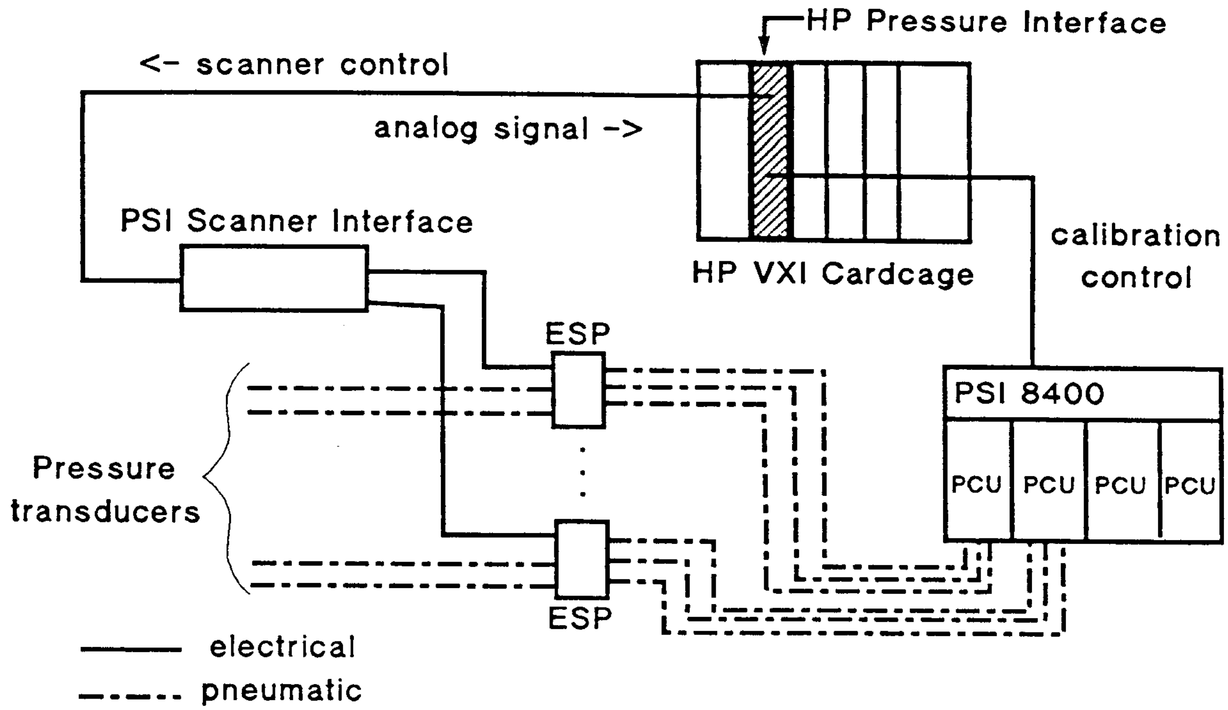
* Provides complete control of PSI 8400 scanners and Pressure Calibration Units (PCUs) for sensor calibration and linearization.

* External trigger input on front panel.

* Front panel LEDs for "Failed," "Access," and "PSI 8400 Control"

The PSI 8400 Control is used in large systems (> 512 pressure channels) where there are multiple E1414As. There will be only one PSI 8400 mainframe controlling the pressure calibration units (PCUs). One of the E1414As controls the PSI 8400, and all E1414As get the PCU setting values through the E1414A that controls the PSI 8400.

Pressure System Configuration



VXI Systems Division
Technical Support:TC:1413_04b

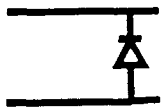
HEWLETT PACKARD

E1414A PSI System

The PSI pressure calibration units (PCUs) in the PSI 8400 mainframe provide standard pressure sources which are controlled through the E1414A pseudo HP-IB port. The E1414A controls the PSI Scanner Interface units through the 50 pin digital/analog connecter, which in turn control the actual pressure sensor blocks (ESPs). The ESPs contain a manifold and a pneumatic switch that can be controlled to switch from the unknown signals to the pressure standard ports.

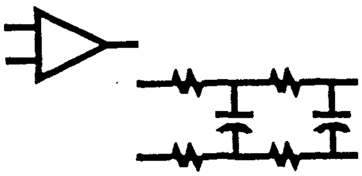
The Scanner Interface units multiplex the pressure transducers into the E1414A's 4 input channels and the E1414A provides scan control using the PSI channel numbering system. The PSI 8400 scan list is programmed through the E1414A via a quasi HP-IB interface on the E1414A through the E1414A firmware, then the E1414A controls and provides power for 8400 scanning via the scan control interface.

8-Channel Signal Conditioning Plug-ons



- **Direct Input**
  - over voltage protection
  - over current protection
  - common mode rejection (-110 dB)

- **Low Pass Filter**
  - Normal Mode Rejection
    6dB at 10Hz
    30dB at 60Hz
  - Passive 2-pole 10Hz filter w buffered output



- **Programmable Gain/Filter**
  - individually programmed channels
  - gains: 1, 8, 64
  - filters: 2 Hz, 10 Hz, 100 Hz

E1413A Signal Conditioning Plug-Ins

At Introduction

Straight Through
Passive RC Filter
Gain and Filter
Current Source (for 4 wire ohms)
Strain Bridge Completion
Breadboard

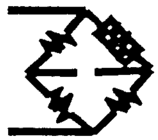
Future

Sample Hold
ACrms Converter

IBT

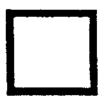
The VXD IBT has quoted a Sample & Hold SCP. Custom SCPs can be done as well as custom EU conversion at speed via special programming of the on board EU engine (DSP chip).

8-Channel Signal Conditioning Plug-ons



- **Strain**
  - 120 or 350 ohm
  - 1/4, 1/2 or full bridge
  - 4V excitation supply
  - auto shunt calibration

- **Current Source**
  - programmable source (488 uA or 30.5 uA)
  - 2 or 4-wire resistance



- **Custom (Breadboard)**
  - six square inches available

E1413A Signal Conditioning Plug-Ins - 2

Note that you can mix SCPs on the E1413A and that they work in conjunction with the EU conversion engine so that you can scan mixed measurement types and no voltmeter setup changes are required. The amplifier SCP can be used to match A/D input signal levels within the 20 bit dynamic settling range (at 100 kHz) of the E1413A, so you can have a 10 volt signal on a channel adjacent to a thermocouple channel.

A breadboard SCP will be available for customers who want to do their own SCPs. The can either just measure the voltage output of their custom SCP and do their own conversion as a postprocessing step, or they can contract with the AEO to provide a custom EU conversion that can be downloaded to the DSP RAM on the E1413A.

A program for building and downloading cusotm EU conversion routines will be provided to SEs.

E1413A Throughput

E1405B/06A: to RAM - 150 Krdgs/sec
HP-IB to external controller - 15 Krdgs/sec (estimated)

V382 to SCSI Disk - 200 Krdgs/sec (estimated)

Radysis EPC-VII - 200+ Krdgs/sec to internal hard disk
?? to external SCSII-2 hard disk

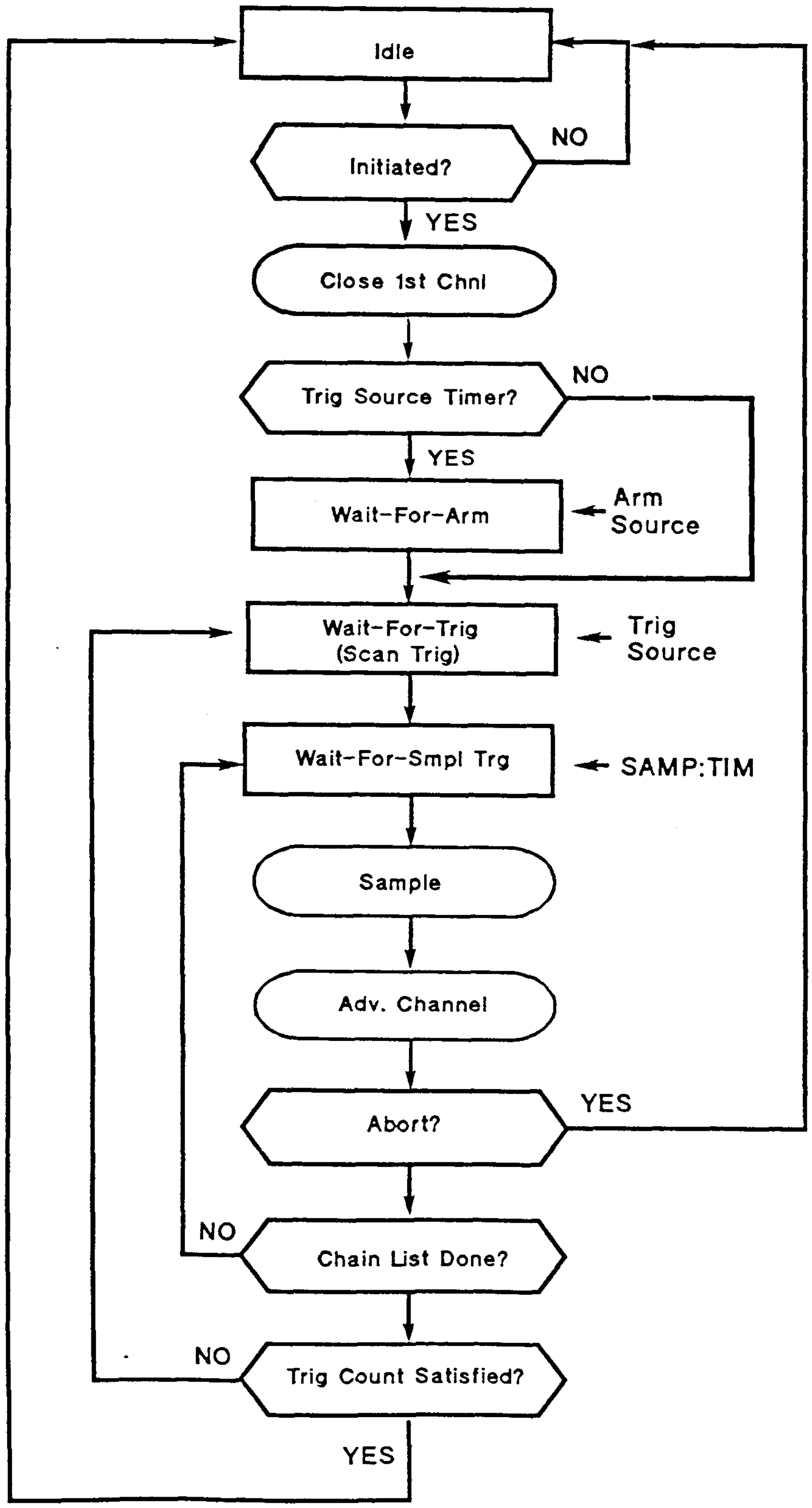
E1413A Throughput
----

400 ns average VXI data read supports 12 cards at maximum rate through the register based interface. This means that the E1413A hardware will not limit throughput to the VXI backplane for continuous data acquisition to disk.

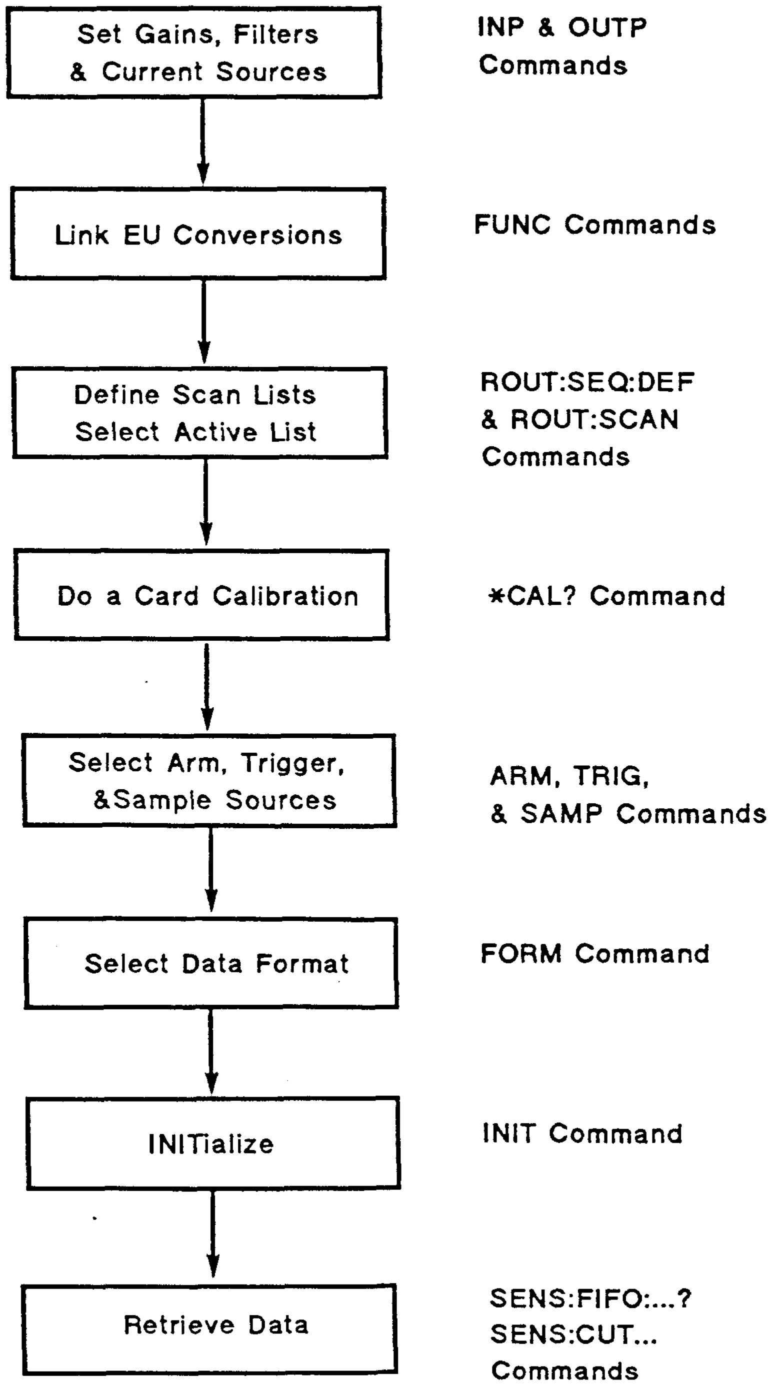
At present the Radusis EPC-VII is the controller of choice. A real time OS for the EPC-VII may be available and supported by VXD soon. The EPC-VII has a SCSII-2 interface and will write from memory to its internal hard disk at 900 KB/sec, so it can keep up with 2 E1413As running at speed.

IEEE floating point format conversion on the fly at speed and conversion to engineering units like strain, temperature, resistance, etc. on board the E1413A at speed are a big deal for throughput issues. The engine test folks use this data real time to make test control decisions. One set of channels goes to a display graph for the hydraulics engineers. Another set goes to a display for the thermodynamic guys, and so on.

E1413 State Design



E1413 SCPI Programming Flow Chart



SCPI and CSCPI Drivers

- No MEAS or CONF commands
- FETCH? for E1405/06 only
- MEM:VME:xxx commands use E1405/06 up to send FIFO readings to shared RAM

The E1413A and E1414A are extended register based modules. This means that they have registers in both A16 and A24 address space. There are SCPI and CSCPI drivers for both modules.

The E1413A has no CONF or MEAS commands. These commands only allow a single function. The E1413A can do multiple mixed functions, so these commands are not appropriate.

There are a couple of commands listed in the E1413A Command Reference that work with the E1405/06 only. These commands are:

```
FETCH?
MEM:VME:xxx
```

The E1405/06 will probably only be used in systems that don't require continuous scanning or high speed data transfer to disk. For such applications the 64K reading memory on-board the E1413/14 is probably adequate. If not, the MEM:VME:xxx commands allow the E1405/06 driver to send data directly to shared memory or VMEmemory over the VXI backplane. The FETCH? command is used to retreive data from VME memory. FETCH? is not used to retreive data from the E1413/14 FIFO or CVT.

Both the E1405/06 and the CSCPI driver use the [SENS:]DATA:FIFO:xxx? and [SENS:]DATA:CVT:xxx? commands to retreive data directly from the E1413/14.

E1413A SCPI Programming – Step 1
SCP Setup

No programming for Straight Through & 10 Hz LP filter SCPs

INP:FILT[:LPAS]:FREQ <cutoff_freq>, (@ch_list>–

INP:FILT[:LPAS][STAT] <ON or OFF>, (@<ch_list>) – *RST is ON

INP:GAIN <gain>, (@<ch_list>) – (gain ratio not dB)

OUTP:CURR <HIGH or LOW>, (@<ch_list>) – (30 uA or 488 uA)

Programming the E1413A is fairly easy, but you do have to keep track of a few things, like which SCP is where and which current source SCP channel is connected to which input channel. There are only 4 main setup steps, a data format selection step, and a data retrieval step. The first step is to program the SCPs. This should be done first so that you know what voltage level the SCP is sending to the A/D for later selection of A/D voltage range.

For the fixed 10 Hz low pass filter SCP and the straight through SCP, no setup commands are used. For the programmable gain/filter SCPs,

```
INP:FILT[:LPAS]:FREQ <cutoff_freq>,(@<ch_list>)
INP:FILT[:LPAS][:STAT] <ON or OFF>,(@<ch_list>) (default is ON)
```

sets the cutoff frequency for the selected channels and enables or disables the filter. Each individual channel can have any available cutoff frequency and can be on or off. The cutoff frequency choices are 2, 10, and 100 Hz.

```
INP:GAIN <gain>,(@<ch_list>)
```

sets the gain to 1, 8, or 64 for each channel in the list. The gain setting is read by the processor at run time and is automatically used in the EU conversion.

The resistance function and the temperature function using RTDs or thermistors require current source SCPs as well as input SCPs. Thus these are two channel/4 wire functions. It does not matter which channel the current source is on, as long as you know which current source channel is connected to which input channel, so you can program the correct EU conversion with the FUNC:RES or FUNC:TEMP (RTD or THER) commands.

```
OUTP:CURR <HIGH or LOW>,(@<ch_list>)
```

sets the current source to either 30 uA (LOW for resistances => 8 Kohm) or 488uA (HIGH for resistances < 8 Kohm). Unfortunately, there is no way to query the current value actually connected to an input channel. You can query the current on the output channel (current source SCP), but you must be sure that you know which output channel is connected to which input channel.

E1413A SCPI Programming – Step 2a
Link EU Conversions

Voltage & Resistance:

[SENS:]FUNC:VOLT <range>, (@<ch_list>)

Use autorange (AUTO) unless you have a good reason not to

[SENS:]FUNC:RES <current_source>, (@<ch_list>)

Use 30 uA for R >= 8 Kohms, 488 uA for R < 8 Kohms

This must agree with the current source for this channel

VXI Systems Division
Technical Support:TCevh:102492:1413_s11

HEWLETT PACKARD

The [SENS:]FUNC:xxx commands set up the Engineering Unit (EU) DSP to perform the desired voltage to resistance, temperature, or strain conversion for each channel. They also set up the voltage range desired for straight voltage channels.

```
FUNC:VOLT AUTO,(@100:115)
```

selects autorange for channels 0 - 15. The channel modifier must always be "1" for the FUNC commands. The default range is is autorange and unless you want to see an overrange error when a fixed range is exceeded, there is no reason to select a fixed range. The E1413A can autorange and maintain all accuracy specifications at speed. FUNC:VOLT AUTO is the default setup for all channels following *RST or power up.

The resistance function requires 2 channels. Both a current source SCP and a voltage input SCP are required for resistance, as well as for thermistors and RTDs.

```
FUNC:RES <current_source>,(@<ch_list>)   -   current source - HIGH or LOW
```

tells the EU DSP to use the LOW or HIGH current source value in the R - V/I conversion (LOW corresponds to 30 uA for resistances > 8 Kohms. HIGH corresponds to 488 uA for resistances < 8 Kohms). This command must agree with the actual current being supplied to the channel. Since the current source can be supplied from any current source SCP channel that you select, or by an external current source, there is no way for the DSP to check to verify that you have used the correct value.

E1413A SCPI Programming - Step 2b
Link EU Conversions

Temperature:

[SENS:]FUNC:TEMP <type>,<sub_type> [ex_curr>],(@<ch_list>)

type, subtype:	TC	J, K, S, T, E, R, EXTended, or CUSTom	
	THER	2250, 5000, or 10000 ohms @ 25C	
	RTD	85 or 92 -	0.385 or 0.392 ohms/C & 100 ohms @ 25C

REF <type>,<sub_type>, (@<ch_list>) - THER,5000 RTD,85 RTD,92 or CUSTom

REF:TEMP <deg_C> - for external reference at a known temperature


```
FUNC:TEMP <type>,<sub_type>,[<ex_current>],(@ch_list)
```

covers thermocouples, RTDs, and thermistors, which is the <type> parameter (TC, THER, or RTD). The subtypes are:

For TC: J,K,S,T,E,R, EXTended or CUSTom
For THER: 2250, 5000, or 10000 ohms @ 25 C
For RTD: 85 or 92 - 0.00385 or 0.00392 ohms/ohm/C, 100 ohms @ 25C

The optional <ex_current> parameter is only used for RTDs and thermistors. This parameter tells the EU DSP which current value to use when converting from voltage to resistance to temperature. This value must agree with the current source SCP value setting done by the OUTP:CURRent command in the next programming step, (HIGH or LOW). The default range is LOW.

Note that you should not use just any nominal 2.25K, 5K, or 10K thermistor, because the temperature versus resistance curves for these can vary widely. The thermistor conversion curves used in the E1413A are based on the table values for the Omega 44000 series or equivalent thermistors which are selected to match the curve to 0.1 C or 0.2 C.

Custom thermocouple or other transducer conversion routines can be downloaded into the DSP memory and we have a program that can do this. It has not yet been decided whether this service will be offered by the VXD IBT or if the program will be made available to SEs.

For thermocouples, the additional command,

```
REF <type>[,<sub_type>], (@ch_list)
```

selects the reference temperature sensor(s) and the channel(s) where it (they) are connected. THER,5000; RTD,85; RTD,92; and CUSTom. are the sensor types supported by the REF command. You must use the 122 uA on-board current source for the thermistor or RTD. For standard E1413As CUSTom will have a conversion table for a type K thermocouple with an ice point reference (ie. two type K TC connected together, one on the isothermal reference connector and the other in an ice point reference).

Alternatively, if you have a well controlled external reference connector at a known temperature, you can just tell the EU DSP what that temperature is with REF:TEMP <deg_C>.

Any time a channel number from the REF ch_list is found in a scan list, that channel is subsequently used as the reference temperature for all following thermocouple channels until the next REF channel is found. The reference temperature reading can be updated as often as you wish by making use of this feature.

E1413A SCPI Programming - Step 2c
Link EU Conversions

Strain:

[SENS:]FUNC:STRN <str_res>,<brdg_type>,<GF>,(@<ch_list>)

Step 3

★ Do a *CAL? (and CAL:TARE, if desired) after the SCP ★
★ setup and EU Conversion Links are done ★

The SCPI commands for strain were not yet available when this was written. The command to link the EUconversion will probably be the FUNC:STRN command shown above.

Strain is a two channel function when using the strain SCP. This is a bridge completion - excitation - shunt verification output SCP. An input SCP is required for reading the bridge output. The excitation voltage is read directly from the strain SCP channel. The EU strain conversion table will use the excitation voltage from the corresponding channel in the FUNC:STRN:VEX channel list to make the voltage to strain conversion. Thus these two channel lists must have a 1:1 relationship of strain input channel to SCP bridge completion channel. You determine how often the excitation voltage value is updated in the strain conversion by how often you measure these channels. Typically the VEX channels can be put in a separate scan list (LISTn) and read before and after (or a few times during) a test.

Since strain bridge outputs are typically on the order of uV for the desired strain resolution, an SCP with gain is generally required. The input SCP determines whether or not you are making what is referred to as a "dynamic" strain gage measurement. This typically requires high input bandwidth and anti-alias filtering. A future sample/hold SCP, or an IBT special high bandwidth amplifier SCP with anti-alias filtering will be required for dynamic strain (shock and vibration) measurements where frequency components over 100 Hz are of interest. The maximum recommended input frequency for the Opt. 13 Gain/Filter SCP is 100 Hz with a sample rate of 1 kHz/channel. (** preliminary information - not yet fully characterized **)

It is possible to do single channel strain measurements if you use external bridge excitation and completion. This will allow measurement of 64 channels of strain with the E1413A using only input SCPs. This will require a known stable excitation voltage source.

There will also be a command to balance the unstrained bridge output (null the bridge). This command is executed prior to the actual strain measurements to zero the bridge output by using a DAC on the strain SCP to offset the bridge imbalance. The maximum imbalance that can be nulled is approximately half the A/D voltage range.

Shunt verification is just a normal bridge measurement, with a 50K resistor shunting the strain gage. For a nominal 120 ohm strain gage, the parallel combination is 119.7 and the equivalent strain for a 1/4 bridge with 4V excitation and gage factor 2 is 1300 mocrostrain. This is used to verify the everything is hooked up and working correctly. A FET on the strain SCP automatically connects the shunt verification resistor when the command to read the shunt verification value is sent.

The E1413A will be able to do the EU conversion at speed, so there is no need for any post processing routines. The raw voltage readings are available if anyone want to write there own conversion routines.

E1413A SCPI Programming - Step 4
Define Scan Lists

ROUT:SEQ:DEF <scan_list>, (@<ch_list>)

example: ROUT:SEQ:DEF LIST1, (@1(00:15), 6(00:10))

Ch Modifier	Conversion and Destination
1	Do EU conversion & store in FIFO and CVT
2	Leave as raw voltage & store in FIFO and CVT
3	Do EU Conversion & store in CVT only
4	Leave as raw voltage & store in CVT only
5	Do EU conversion & store in FIFO only
6	Leave as raw voltage & store in FIFO only

ROUT:SEQ:DEF <scan_list>,(@<ch_list>) assigns a name (LIST1 - LIST4) and a scan sequence to a group of channels (up to 1024 channels). This command also assigns a data path and turns the EU conversion on or off for each channel. This is determined by the data modifier that you select for the channel or group of channels. For example,

```
ROUT:SEQ:DEF LIST1,(@1(00:15), 6(00:10))
```

defines LIST1 and specifies that the readings taken on channels 0 through 15 are to be converted to into the engineering units selected for these channels by the FUNC commands and these values are to be stored in both the FIFO and the CVT. In addition, the raw voltage values for the first 10 channels are also sent to the FIFO. You must keep track of this when reading data out of the CVT and FIFO.

Each E1413A is an individual instrument and has 64 channels. You can't combine cards to make a large instrument like you do with an E1411B and multiplexer modules. You wouldn't want to do this because it would defeat the ability of multiple E1413As to operate simultaneously. The E1413A may not be the best choice for systems with a large channel count where the individual channel reading rate is not very fast.

The channel modifier for the E1413A looks very much like the card number for an E1411B scanning voltmeter, but the function of the modifier is completely different. The E1413A channel modifier selections and they're functions are listed in the table above.

E1413A SCPI Programming - Step 5

Arm, Scan Trigger, and Sample Trigger Programming

Recommended sequence is inside out - lowest level first

1. SAMPL:TIM <LISTn or ALL>, <10 us to 32.768 ms>

2. TRIG:SOUR <BUS, EXT, HOLD, IMM, SCP, TIM, or TTLTn

3. If TRIG:SOUR TIM, then TRIG:TIM <100 us to 6.5536 sec>

★ TRIG:TIM setting must be > (SAMP:TIM) X (number of channels in the scan list) + 30 us + 1 sample interval ★

4. TRIG:COUN <0 to 65535 or INF> - *RST = 1 (0 or INF = continuous)

5. ARM:SOUR <BUS, EXT, HOLD, IMM, SCP, or TTLTn>

VXI Systems Division
Technical Support:TCevh:102492:1413_s15

HEWLETT PACKARD

The best way to program the E1413A trigger system is from the inside out, ie., the lowest level first. This is recommended because you need to know the lowest level sample interval before you can select the minimum scan interval, which is the sample interval times the number of channels in the active scan list.

SAMP:TIM <LISTn or ALL>,<10 us to 32.768 ms>

sets the sample rate. Each of the four possible scan lists can have a different sample rate. The *RST condition is 10 us, and this will usually be the value you want. The internal timer is the only sample source. You can't step through the scan list with an external sample trigger. There is no sample count setup command. To get multiple samples on individual channels within the same scan, you must enter the channel numbers multiple times in the scan list. This is useful if you must go from a high level signal to a low level signal.

NOTE: To get settling to 16 bits on the 62.5 mV range from a high level signal on the 16 volt range takes about 100 us. Much of this can be eliminated with the proper selection and setting of SCPs. Attenuation SCPs will be available soon.

TRIG:SOUR (BUS, EXT, HOLD, IMM, SCP, TIM, or TTLTn)

selects the source of the scan trigger. The *RST condition is HOLD. EXT selects the "Trig" port on the E1413A terminal block. IMMediate selects continuous triggering. In this mode the first channel of the scan list follows the last channel at the programmed sample interval. TRIG:SOUR SCP is for a future comparator SCP.

If the trigger source is set to TIMer, the TRIG:TIM <100 us to 6.5536 sec> command selects the scan trigger time interval. The scan trigger interval selected must be greater than the sample interval times the number of channels in the active scan list or you will get a "trigger too fast" error. The *RST trigger interval is 100 us. If you have fewer than 10 channels and this is not fast enough, just repeat the channels in the scan list.

TRIG:COUN <0 to 65535 or INF>

allows you to select a fixed number of triggers, or TRIG:COUN 0 or INF disables the trigger counter and allows the E1413A to accept an unlimited number of triggers. TRIG:COUN 1 is the *RST condition.

The trigger arming circuit is only in effect when TRIG:SOUR TIMer is specified.

ARM:SOUR <BUS, EXT, HOLD, IMM, SCP, or TTLTn>

The *RST condition is ARM:SOUR HOLD. When ARM:SOUR IMM, the INIT command will start the scan. The response will be a little faster if you first set ARM:SOUR <BUS or HOLD>, issue the INIT command to set up the E1413A and put it in the Wait-For-Arm state, then issue the ARM[:IMMediate] command. ARM:SOUR SCP will work with a future comparator SCP.

E1413A SCPI Programming - Step 6

Status System Programming - Interrupts

Standard Operation Status Register

STAT:OPER:ENABle <mask>		
Bit	**Decimal Weight**	**Condition**
8	256	Scan Complete
9	512	SCP Trigger
10	1024	FIFO Half Full

Questionable Data Register

STAT:QUES:ENABle <mask>		
Bit	**Decimal Weight**	**Condition**
8	256	Overvoltage
9	512	Trigger too fast
10	1024	FIFO Full

The STAT:OPER:ENABle <mask> and STAT:QUES:ENABle <mask> commands enable the E1413A to interrupt on the selected irq line. *SRE 8 allows any of the unmasked Questionable Data conditions to pull SRQ to interrupt a Basic/WS program. Due to the interrupt performance of UNIX, we don't recommend using interrupts to determine when to read data from the FIFO when the E1413A is being operated at any of the faster scan/sample speeds. For the V382, the interrupt response can be up to 60 ms. Using real time priority and locked processes, you can get 99% probability of 5ms response. If the scanning speed is such that you can live with this interrupt response, you should be OK. The important point is that you are aware of the interupt response of the controller you are using and have accounted for it. An example V382 C program using the SICL "I_INTR_VXI_SIGNAL" routine is included at the end of this section.

Normally you will not need an interrupt because the V382 will be dedicated to reading data from the FIFO and/or CVT and sending data to a disk file and updating displays. The SENS:DATA:FIFO:HALF? query will wait for 32K readings which can be dumped fast enough (400 ns/rdg) that you will have time to write to a disk file and update display buffers between FIFO accesses. There is also a SENS:DATA:FIFO:COUNT:HALF? that returns a "1" when the FIFO is half full or greater. This is the recommended command for polling to see when 32K readings are available.

If you do wish to query the status, the STAT:OPER:COND? and STAT:QUES:COND? query commands provide the status information.

E1413A SCPI Programming - Steps 7 & 9

FIFO Mode, Data Format and Retrieval

[SENS:]DATA:FIFO:MODE <BLOCk or OVERwrite>

FORMat[:DATA] <format>[,<size>] - REAL,32 REAL,64 ASCII,14

[SENS:]DATA:FIFO[:ALL]?

[SENS:]DATA:FIFO:HALF?

[SENS:]DATA:FIFO:PART? <1 to 32767>

[SENS:]DATA:FIFO:COUNT? - returns the number of readings in the FIFO

[SENS:]DATA:FIFO:COUNT:HALF? - returns "1" when FIFO count >= half full

[SENS:]DATA:CVT? (@<ch_list>)

VXI Systems Division
Technical Support:TCevh:102492:1413_s17

HEWLETT PACKARD

The E1413A FIFO has two operating modes selected with the [SENS:]DATA:FIFO:MODE <BLOCK or OVERWRITE> command. The *RST mode is BLOCK. In this mode, the A/D stops writing data into the FIFO when it is full. It keeps sampling, but the data is lost. The FIFO full bit is set and and error message is put in the error que. In the OVERWRITE mode, the FIFO is a circular buffer and old readings that have not been read are overwritten when the FIFO is full.

FORMat[:DATA] <format>[,<size>] sets the format for data returned using the [SENSe:]DATA:FIFO:xxx?, [SENS:]DATA:CVT?, and FETCH? commands The choices are REAL,32, REAL,64 or ASCII,14 (7 bit ASCII, 14 characters/rdg). REAL,32 is the FIFO format of the E1413A and is the *RST format. Since no conversion step is required, REAL,32 provides the highest data transfer performance. For Basic it is faster to enter REAL,32 data into two 16 bit integers over a binary path, than to use REAL,64 A Basic REAL,32 conversion program is included at the end of this section.

There are several SCPI command choices for retrieving data form the E1413A FIFO. The standard FETCH? command only works with the E1405B/06A to read data from VME (or shared) memory. The E1413A SCPI driver must first be set up to send readings to VME memory (or shared memory) using the MEM:VME:xxx setup commands.

These are the three commands for reading data from the E1413A FIFO. DATA:FIFO? can be used to acquire all readings (even while they are being made). This command does not complete until the scanning stops. The data retrieval command that will most often be used in a continuous data acquisition operation is DATA:FIFO:HALF? This command provides a fast means of acquiring 64 Kbyte (32 Krdg) blocks of readings. [SENS:]DATA:FIFO:COUNT?

returns the number of readings currently in the FIFO.

[SENS:]DATA:CVT? (@<ch_list>) reads the selected channels from the CVT. Remember that the channel data must have been previously directed to the CVT and/or the FIFO with the appropriate FUNC:xxx command.

E1413A Register Programming

- Well documented in manual
- Manual has flow charts for:
  - Reset
  - Scan Control & Trigger Register Programming
  - Command & Parameter Register Programming
  - Reading Data & Querying Settings
- Manual has Basic and C Register Programming Examples

The E1413A is a VXI Extended Device, which means that it is a register based device with additional registers in A24 space. There are 64 CVT registers, 1 for each channel, that are mapped into the A24 address space reserved for the E1413A by the VXI resource manager. The A24 base address is found by reading the Offset Register in A16 space, that is located at address base + 6 (49152+laddr*64+6). This is automaticaly done by the SICL "I_MAP_EXTEND" routine which will conveniently map all the registers into a data structure for you as shown in example ??? at the end of this section.

There are individual A16 registers for software trigger, the trigger timer setting, the trigger source, scan control, FIFO mode, and interrupt setup. In addition, you will find semi-synchronous and asynchronous trigger mode options which are defined in the VXI specification, but are not supported with the SCPI and CSCPI drivers. Programming each of these setup conditions only requires a simple register write operation.

All of the other setup programming (selecting the data destination, linking the conversion routines, seting SCP gain, filter, and current source, and programming the scan trigger timer) is done using the parameter and command registers following the flow chart above.

E1413A Register Programming
Program Sequence

1. Reset the card by writing to the Sysfail Inhibit and Reset bits in the Control Register.

2. Set up SCP gain, filter, and current source for each channel using the SCBWRITE <regaddr> <word> command opcode and parameters.

3. Do a channel calibration to set gain and offset corrections (up to SCP) using the CARDCAL command opcode.

4. Link conversions to the channels using the ASSIGN <channel> <conversion> command opcode and parameters.

5. Assign channel to a scan list, set voltage range, and set data destination using the APPENDn <channel> <range> <flags> command opcode and parameters. There are four APPEND opcodes, one four each of the four possible scan lists.

6. Clear the CVT (set to Not A Number) using the CVTINIT command opcode (no parameters).

7. Set channel advance rate using the ADVRATEn command opcode and interval parameter.

8. Select the trigger source by writing to the Trigger Mode Register and the Trigger Timer Register.

9. If a fixed number (=< 64K scans are desired, select the trigger count using the SEQCOUNT command opcode and count parameter.

10. Set up interrupts by writing to the Interrupt Configuration Register.

11. Arm the E1413A to accept triggers by writing to the Scan Control Register.

12. Start triggering or software trigger by writing to the Software Trigger Register.

13. Read data from the FIFO or CVT registers.

14. Disarm the E1413A if in continuous mode by writing to bit 7 of the Scan Control Register. This stops the E1413A at the end of a scan.

15. Disable interrupts by writing 0 to the Interrupt Configuration register.

16. Read the remaining data from the FIFO.

DELAY EQUALIZED SHARP CUTOFF

LOWPASS

Custom-Built LC Filters 1KHz to 15MHz

The **DELAY EQUALIZED SHARP CUT-OFF LOWPASS FILTERS** tabulated on this page are the result of many years of experience in the use of specialized computer programs for the design and optimization of Delay Equalized Filters. By using modern digital computers, the composite behavior of the filter and equalizer are optimized to yield the ultimate [illegible] amplitude and delay response.

This type of filter [illegible] deally suited for use as an Anti-Aliasing Filter [illegible] n analog to digital conversion. When used as a Post-Aliasing Filter in digital processing applications, the passband can be shaped [illegible] correct for sin x/x amplitude distortion.

The [illegible] ers tabulated below represent a widely used group. However, many other combinations of stopband ratio, impedance, delay distortion and size are possible. Two stopband ratios are listed in the table below, 1.22 @ 45dB and 1.32 @ 60dB.

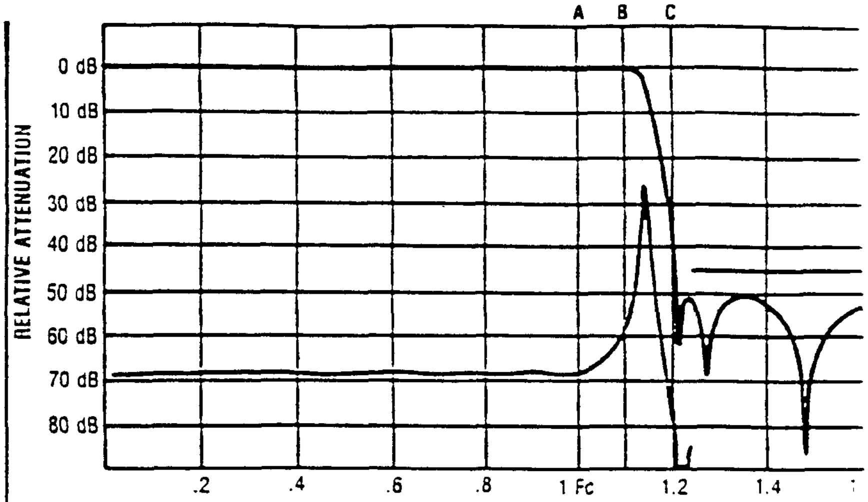
Units normally supplied in metal cans for printed circuit mounting. SMA connectors same size. BNC connectors may require larger cans.

Amplitude, phase and/or delay matching between filters is also available.

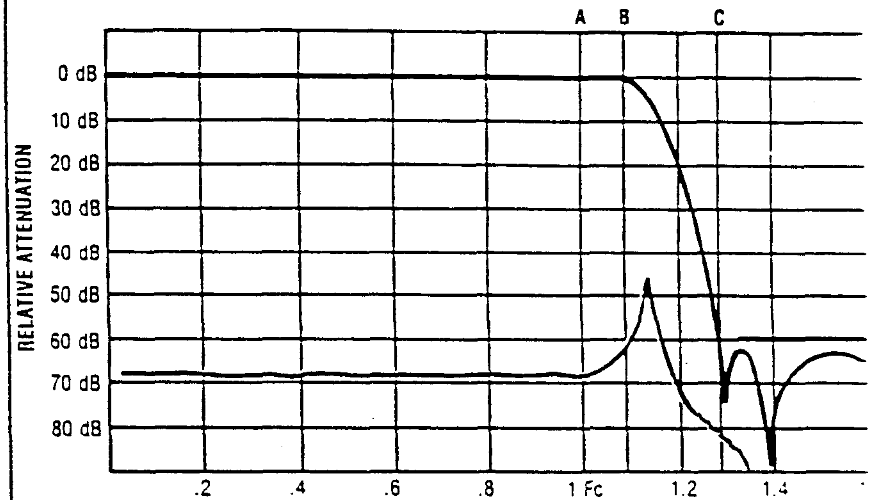
CALL FACTORY FOR SPECIAL SIZES AND DELIVERY INFORMATION.

ORDER ANY CUT-OFF FREQUENCY FROM 1KHz TO 15MHz. INTERPOLATION BETWEEN TABULATED DATA ALLOWABLE.

See page 33 for mechanical specifications.



STOPBAND RATIO #1 = 1.22



STOPBAND RATIO #2 = 1.32

Normalized Plot of Amplitude & Delay Response of Delay Equalized Lowpass Filte

A = −.25dB Frequency **B** = −3dB Frequency **C** = −45dB Frequency

Stopband Ratio #1 = 1.22 @ 45dB
Delay (D) = 25.37

Stopband Radio #2 = 1.32 @ 60dB
Delay (D) = 25.03

$$\text{APPROXIMATE PASSBAND DELAY (Seconds)} = \frac{\text{Delay (D)}}{2 \times \pi \times \text{Frequency A(Hz)}}$$

± .25dB MAXIMUM RIPPLE—2dB MAXIMUM INSERTION LOSS
MAXIMUM DELAY VARIATION ± 3% TO −.25dB FREQUENCY

Maximum −.25dB Cut-Off Frequency (Last point that delay flatness is specified) (Graph location A)	Maximum 3dB Attenuation Frequency (Graph location B) B = 1.14 × A	Attenuation Frequency Graph Location C		Impedance Range (Ohms)	Approximate Passband Delay Micro-Seconds		Standard Size Inches	"Space-Saving" Size Inches
		45dB Ratio #1 C = 1.22 × A	60dB Ratio #2 C = 1.32 × A		Ratio #1	Ratio #2		
1.0 KHz	1.14 KHz	1.22 KHz	1.32 KHz	500-2.5K	4038	3984	6 × 3 × 1¼	———
2.5 KHz	2.85 KHz	3.05 KHz	3.30 KHz	500-2.5K	1615	1539	6 × 3 × 1¼	———
5.0 KHz	5.70 KHz	6.10 KHz	6.60 KHz	500-2.5K	808	797	6 × 3 × 1¼	———
10.0 KHz	11.40 KHz	12.22 KHz	13.20 KHz	500-2.5K	404	398	6 × 2 × 1¼	5 × 2 × 1¼
25.0 KHz	28.50 KHz	30.50 KHz	33.00 KHz	100-1.0K	162	159	5 × 3 × 1¼	5 × 2 × 1¼
50.0 KHz	57.00 KHz	61.00 KHz	66.00 KHz	50-500	81	79	5 × 3 × 1¼	5 × 2 × 1¼
100.0 KHz	114.00 KHz	122.00 KHz	132.00 KHz	50-200	40	39	5 × 2 × 1¼	4 × 2 × 1¼
250.0 KHz	285.00 KHz	305.00 KHz	330.00 KHz	50-100	16	15	5 × 2 × 1¼	4 × 2 × 1¼
500.0 KHz	570.00 KHz	610.00 KHz	660.00 KHz	50-100	8.08	7.96	5 × 2 × 1¼	4 × 2 × 1¼
1.0 MHz	1.14 MHz	1.22 MHz	1.32 MHz	50-75	4.04	3.98	5 × 2 × 1¼	4 × 2 × 1¼
2.5 MHz	2.85 MHz	3.05 MHz	3.30 MHz	50-75	1.62	1.59	5 × 2 × 1¼	4 × 2 × 1¼
5.0 MHz	5.70 MHz	6.10 MHz	6.60 MHz	50-75	.81	.79	5 × 2 × 1¼	4 × 2 × 1¼
7.5 MHz	8.55 MHz	9.15 MHz	9.90 MHz	50-75	.537	.531	5 × 2 × 1¼	4 × 2 × 1¼
10.0 MHz	11.40 MHz	12.20 MHz	13.20 MHz	50-75	.404	.398	5 × 2 × 1¼	4 × 2 × 1¼
*12.5 MHz	14.25 MHz	15.25 MHz	16.50 MHz	50-75	.322	.318	5 × 2 × 1¼	4 × 2 × 1¼
*15.0 MHz	17.10 MHz	18.30 MHz	19.80 MHz	50-75	.268	.264	5 × 2 × 1¼	———

*At frequencies above 10MHz, the maximum attenuation at location A becomes .5dB and the delay variation up to location A becomes ± 5%.

```
10  !  RE-SAVE "QK01"
20  !  This program ilustrates taking 10 bursts of 2000 readings at 20 MHz
30  !  with the E1429A/B.  20K readings are transferred in packed format
40  !  over a binary path to the V382 and converted to reals in 2.2 sec.
50  !
60     DIM A$[80],Ndig$[1],Count$[9],Res$[8],Rdgs(1:20000)
70     REAL Size,Range,Res,Port
80     INTEGER Irdgs(1:20000)
90     ASSIGN @Qk TO 1640
100    ASSIGN @Bin TO 1640;FORMAT OFF
110    CLEAR @Qk
120    OUTPUT @Qk:"*RST;*CLS"
130    OUTPUT @Qk;"*IDN?"
140    ENTER @Qk;A$
150    PRINT A$
160    !
170    OUTPUT @Qk;"*RST;*CLS"
180    OUTPUT @Qk;"CONF1:ARR:VOLT (2000),0.7,(@1)"
190    OUTPUT @Qk;"FORMAT PACKED"
200    OUTPUT @Qk;"ARM:COUN 10"
210    OUTPUT @Qk;"ARM:SOUR BUS"
220    OUTPUT @Qk;"TRIG:SOUR TIM1"
230    OUTPUT @Qk;"TRIG:TIM1 50 ns" ! 2000 * 50 ns = 100 us bursts
240    !
250    OUTPUT @Qk;"INIT"
260    FOR I=1 TO 10
270      OUTPUT @Qk;"ARM" !*TRG"
280 !    WAIT .011          ! 11 ms is the minimum wait to avoid "Trigger Ignored"
290    NEXT I              ! with *TRG.  10 ms internal delay to be removed.
300    !
310    OUTPUT @Qk;"FETCH1:COUN?"
320    ENTER @Qk;A
330    PRINT "COUNT: ";A
340    !
350    OUTPUT @Qk;"CONF1?"                ! Get the resolution for the range set
360    ENTER @Qk;Size,Range,Res,Port      ! with the CONF command.
370    REPEAT
380      OUTPUT @Qk;"SYST:ERR?"
390      ENTER @Qk;Ec,A$
400      PRINT Ec,A$
410    UNTIL Ec=0
420    !
430    T1=TIMEDATE
440    OUTPUT @Qk;"FETCH1?"
450    ENTER @Bin USING "#,X,K,K";Ndig$;Count$[1;VAL(Ndig$)]
460    ENTER @Bin;Irdgs(*)
470    ENTER @Qk;Lf$
480    FOR I=1 TO 20000
490      Rdgs(I)=SHIFT(Irdgs(I),4)*Res
500    NEXT I
510    T2=TIMEDATE-T1
520    PRINT "Time to convert and transfer 20,000 packed readings:";DROUND(T2,4)
530    PRINT "Readings:"
540    FOR I=1 TO 20 STEP 5
550      PRINT Rdgs(I),Rdgs(I+1),Rdgs(I+2),Rdgs(I+3),Rdgs(I+4)
560    NEXT I
570    END
```

```
HEWLETT-PACKARD,E1429A,0,A.00.06
COUNT:  20000
 0        "No error"
Time to convert and transfer 20,000 packed readings: 2.23
Readings:
 .2985     .2985     .2985     .298      .298
 .298      .2975     .298      .298      .298
```

```
10    !  RE-SAVE "QK018"
20    !  E1429A/B taking 10 bursts of 2000 readings at 20 MHz.  20K Real,64
30    !  reaindgs are transferred to the V382 with FORMAT OFF in 19 sec.
40    !
50       DIM A$[40],Ndig$[1],Count$[9],Rdgs(1:20000)
60       ASSIGN @Qk TO 1640
70       ASSIGN @Bin TO 1640;FORMAT OFF
80       CLEAR @Qk
90       OUTPUT @Qk;"*RST;*CLS"
100      OUTPUT @Qk;"*IDN?"
110      ENTER @Qk;A$
120      PRINT A$
130      !
140      OUTPUT @Qk;"*RST;*CLS"
150      OUTPUT @Qk;"CONF1:ARR:VOLT (2000),0.7,(@1)"
160      OUTPUT @Qk;"FORMAT REAL,64"
170      OUTPUT @Qk;"ARM:COUN 10"
180      OUTPUT @Qk;"ARM:SOUR BUS"
190.  .  OUTPUT @Qk;"TRIG:SOUR TIM1"
200      OUTPUT @Qk;"TRIG:TIM1 50 ns" ! 2000 * 50 ns = 100 us
210      !
220      OUTPUT @Qk;"INIT"
230      FOR I=1 TO 10
240        OUTPUT @Qk;"ARM"  !*TRG"
250 !      WAIT .011         ! 11 ms is the minimum wait to avoid "Trigger Ignored"
260      NEXT I              ! with *TRG.  No wait required with ARM ??????????
270      !
280      REPEAT
290        OUTPUT @Qk;"SYST:ERR?"
300        ENTER @Qk;Ec,A$
310        PRINT Ec,A$
320      UNTIL Ec=0
330      !
340      T1=TIMEDATE
350      OUTPUT @Qk;"FETCH1?"
360      ENTER @Bin USING "#,X,K,K";Ndig$;Count$[1;VAL(Ndig$)]
370      ENTER @Bin;Rdgs(*)
380      ENTER @Qk;Lf$
390      T2=TIMEDATE-T1
400      PRINT "Time to convert and transfer 20,000 REAL,64 readings:";DROUND(T2,4
)
410      PRINT "Readings:"
420      FOR I=1 TO 20 STEP 5
430        PRINT Rdgs(I),Rdgs(I+1),Rdgs(I+2),Rdgs(I+3),Rdgs(I+4)
440      NEXT I
450      END

HEWLETT-PACKARD,E1429A,0,A.00.06
 0          "No error"
Time to convert and transfer 20,000 REAL,64 readings: 18.88
Readings:
 .2975      .299       .2985      .299       .299
 .2985      .298       .298       .297       .298
 .2985      .2975      .298       .299       .298
 .298       .298       .298       .2985      .2985
```

```
10   !  RE-SAVE "QK05[illegible]"
20   !  TRIG:SOUR VME  E1429A trigger & transfer using READIO of A24 data register.
30   !
40      DIM A$[80],Ndig$[1],Count$[9],Res$[8],Rdgs(1:20000)
50      REAL Size,Range,Res,Port,A24_base,A24_data
60      INTEGER Laddr,Irdgs(1:20000),A24_stat,Req_mem
70      Laddr=40
80      ASSIGN @Qk TO 1600+Laddr
90      CLEAR @Qk
100     !
110     ! Map A16 space.
120     !
130     CONTROL 16,25;2
140     !
150     ! Read offset register for A24 base address of E1429A.
160     !
170     A24_base=READIO(-16,49152+64*Laddr+6)*256.
180     !
190     PRINT "A24 Base Address: ";A24_base
200     A24_data=A24_base+12
210     !
220     ! Map A24 space.
230     !
240     CONTROL 16,25;3
250     CONTROL 16,26;0    ! MAP PAGE 0
260     !
270     OUTPUT @Qk;"*RST;*CLS"
280     OUTPUT @Qk;"*IDN?"
290     ENTER @Qk;A$
300     PRINT A$
310     !
320     OUTPUT @Qk;"*RST;*CLS"
330     OUTPUT @Qk;"CONF1:ARR:VOLT (20000),0.7,(@1)"
340     OUTPUT @Qk;"ARM:SOUR IMM"
350     OUTPUT @Qk;"TRIG:SOUR VME"
360     !
370     OUTPUT @Qk;"INIT"
380     T1=TIMEDATE
390     FOR I=1 TO 20000
400       Irdgs(I)=READIO(-16,A24_data)
410       Ch2data=READIO(-16,A24_data)
420     NEXT I
430     T2=TIMEDATE-T1
440     PRINT "Time to read 20,000 readings with VME read: ";DROUND(T2,4)
450     !
460     OUTPUT @Qk;"CONF1?"                ! Get the resolution for the range set
470     ENTER @Qk;Size,Range,Res,Port     ! with the CONF command.
480     REPEAT
490       OUTPUT @Qk;"SYST:ERR?"
500       ENTER @Qk;Ec,A$
510       PRINT Ec,A$
520     UNTIL Ec=0
530     !
540     T1=TIMEDATE
550     FOR I=1 TO 20000
560       Rdgs(I)=Irdgs(I)*Res/16
570     NEXT I
580     T2=TIMEDATE-T1
590     PRINT "Time to convert 20,000 packed readings to reals:";DROUND(T2,4)
```

```
600     PRINT "Readings:"
610     FOR I=1 TO 20 STEP 5
620       PRINT Rdgs(I),Rdgs(I+1),Rdgs(I+2),Rdgs(I+3),Rdgs(I+4)
630     NEXT I
640     END
```

```
A24 Base Address:  262144
HEWLETT-PACKARD,E1429A,0,A.00.06
Time to read 20,000 readings with VME read:  2.99   — 6.7 KSa/sec.
 0          "No error"
Time to convert 20,000 packed readings to reals: 1.61
Readings:
 .2985     .2985     .2985     .2985     .2985
 .2985     .2985     .2985     .2985     .2985
 .2985     .2985     .2985     .2985     .2985
 .2985     .2985     .2985     .2985     .2985
```

```
10  ! RE-SAVE "QK05B"
20  ! Fast E1429A memory read using A24 data register and READIO.
30  !
40    DIM A$[80],Ndig$[1],Count$[9],Res$[8],Rdgs(1:20000)
50    REAL Size,Range,Res,Port,A24_base,A24_data
60    INTEGER Laddr,Irdgs(1:20000),A24_stat,Req_mem
70    Laddr=40
80    ASSIGN @Qk TO 1600+Laddr
90    CLEAR @Qk
100   !
110   ! Map A16 space.
120   !
130   CONTROL 16,25;2
140   !
150   ! Read offset register for A24 base address of E1429A.
160   !
170   A24_base=READIO(-16,49152+64*Laddr+6)*256.
180   !
190   PRINT "A24 Base Address: ";A24_base
200   A24_data=A24_base+12
210   !
220   ! Map A24 space.
230   !
240   CONTROL 16,25;3
250   CONTROL 16,26;0    ! MAP PAGE 0
260   !
270   OUTPUT @Qk;"*RST;*CLS"
280   OUTPUT @Qk;"*IDN?"
290   ENTER @Qk;A$
300   PRINT A$
310   !
320   OUTPUT @Qk;"*RST;*CLS"
330   OUTPUT @Qk;"CONF1:ARR:VOLT (20000),0.7,(@1)"
340   OUTPUT @Qk;"ARM:SOUR IMM"
350   OUTPUT @Qk;"TRIG:SOUR TIM1"
360   OUTPUT @Qk;"TRIG:TIM1 50 ns"
370   !
380   OUTPUT @Qk;"INIT;*OPC?"
390   ENTER @Qk;Cp
400   T1=TIMEDATE
410   FOR I=1 TO 20000
420     Irdgs(I)=READIO(-16,A24_data)
430     Ch2data=READIO(-16,A24_data)
440   NEXT I
450   T2=TIMEDATE-T1
460   PRINT "Time to read 20,000 readings with VME read: ";DROUND(T2,4)
470   !
480   OUTPUT @Qk;"CONF1?"             ! Get the resolution for the range set
490   ENTER @Qk;Size,Range,Res,Port   ! with the CONF command.
500   REPEAT
510     OUTPUT @Qk;"SYST:ERR?"
520     ENTER @Qk;Ec,A$
530     PRINT Ec,A$
540   UNTIL Ec=0
550   !
560   T1=TIMEDATE
570   FOR I=1 TO 20000
580     Rdgs(I)=Irdgs(I)*Res/16
590   NEXT I
```

```
600    T2=TIMEDATE-T1
610    PRINT "Time to convert 20,000 packed readings to reals:";DROUND(T2,4)
620    PRINT "Readings:"
630    FOR I=1 TO 20 STEP 5
640      PRINT Rdgs(I),Rdgs(I+1),Rdgs(I+2),Rdgs(I+3),Rdgs(I+4)
650    NEXT I
660    END
```

```
A24 Base Address:  262144
HEWLETT-PACKARD,E1429A,0,A.00.06
Time to read 20,000 readings with VME read:  2.93
 0        "No error"
Time to convert 20,000 packed readings to reals: 1.59
Readings:
 .2995     .2985     .2985     .2985     .299
 .298      .2985     .299      .298      .298
 .2985     .299      .299      .298      .298
 .298      .2985     .298      .2985     .2985
```

— 6.8 KSa/sec for 1 ch

13.65 for 2 ch

qk01 - 1/3

```
/* qk01.c
 *
 * E1429A SE315 example program 1, using packed format.
 * ompile with "cc -g -Aa qk01.c -o qk01 -lcscpi -lsicl -lm"
 *    libraries must be linked in the order above.    11-04-92 TC
 */

#include        <math.h>
#include        <string.h>
#include        <stdio.h>
#include        <stdlib.h>
#include        <fcntl.h>
#include        <sicl.h>
#include        <sys/lock.h>
#include        <sys/rtprio.h>
#include        <time.h>

#define ARM_CNT         10
#define TRG_CNT         2000
#define Res             .0005

/* Function for timing                                  */

        unsigned long   start_sec, start_usec;
        unsigned long   end_sec, end_usec;

double time_diff( start_sec, start_usec, end_sec, end_usec)
    {
    return (0.001 * (1000000 * (end_sec - start_sec) + (end_usec
                                        - start_usec)));
    }

/***************************************************************************
 *****************   MAIN PROGRAM   ****************************************
 ***************************************************************************/

main(){
        INST            e1429a;
        int             i, j, Cp, ercode;
        char            msg[255];
        float           Rdgs[ARM_CNT * TRG_CNT];
        unsigned short  raw_data[ARM_CNT * TRG_CNT];
        int             number, count;
        double          cmd_time, ohd_time;

        ionerror(I_ERROR_EXIT);

        /* Open a VXI device session for the E1429A at LA 40.   */

        e1429a = iopen("vxi,40");

        /* Setup the E1429A using SCPI commands and WSP  */

        iprintf(e1429a, "*RST;*CLS\n");
        ipromptf(e1429a, "*IDN?\n", "%s\n", msg);
        printf("IDN QUERY: %s\n",msg);

        iprintf(e1429a, "CONF1:ARR:VOLT (2000),0.7,(@1)\n");
        iprintf(e1429a, "FORM PACK\n");
        iprintf(e1429a, "ARM:COUN 10\n");
```

gk01-

```
iprintf(el429a, "ARM:SOUR BUS\n");
iprintf(el429a, "TRIG:SOUR TIM1\n");
iprintf(el429a, "TRIG:TIM1 50 ns\n");
iprintf(el429a, "INIT\n");

/* Arm the E1429A 10 times using the ARM command.                    */

for (i=0; i<10; i++){
    for (j=0; j<1000; j++);
    iprintf(el429a, "ARM\n");
                  }

ipromptf(el429a, "*OPC?\n", "%d", &Cp);
printf("OPC?: %d\n", Cp);
ipromptf(el429a, "FETCH1:COUN?\n", "%d", &count);
/*      printf("COUNT = %d\n", count);   */

/* Get the waveform data.                                       */
/* Begin timing sequence.                                       */
(void) os_time( &start_sec, &start_usec);

ipromptf(el429a, "FETCH?\n", "%#wb%*t", &count, raw_data);
/* "count" is updated with actual number of readings (words) read. */
/* printf("count: %d\n", count); */

for (i=0; i<(ARM_CNT * TRG_CNT); i++)
   Rdgs[i] = (float) (raw_data[i]>>4) * Res;

/* End timing and compute execution time                        */
(void) os_time( &end_sec, &end_usec);
cmd_time = time_diff(start_sec, start_usec, end_sec, end_usec);

/* Find overhead time for "for" loop.                           */
/* Begin timing sequence.                                       */
(void) os_time( &start_sec, &start_usec);

for (i=0; i<(ARM_CNT * TRG_CNT); i++);

/* End timing and compute execution time                        */
(void) os_time( &end_sec, &end_usec);
ohd_time = time_diff(start_sec, start_usec, end_sec, end_usec);
printf("Execution time: %f msec\n",cmd_time - ohd_time);

for (i=0; i<20; i+=5){
  printf(" %f %f %f %f %f \n",Rdgs[i],Rdgs[i+1],Rdgs[i+2],Rdgs[i+3],
                               Rdgs[i+4]);
                               }

while (1) {
  ipromptf(el429a, "SYST:ERR?\n", "%d%t", &ercode, msg);
  printf("%d %s\n", ercode, msg);
  if (ercode == 0)
    break;
    }

iclose(el429a);
exit(0);

/*
```