PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 33, NO. 4, PP. 195-204 (1989)

ARCHITECTURE OF A MODERN PROCESSING OSCILLOSCOPE

H. FÜRST* AND H. FLECK**

*Institute of Electronic Measurement,
Technical University Vienna, Austria

**TRACE Elektronische Geräte GmbH, Vienna, Austria

Received June 15, 1988.

Abstract

For many applications the functions of a voltmeter, counter, power meter etc. can be provided by a single measurement instrument, the digitizing processing oscilloscope. An instrument featuring these capabilities has to be designed with respect to high speed data transfers and simultaneous calculation of processed waveforms as well as scalar measurement functions. This paper describes the architecture of a recently introduced modern processing oscilloscope which is able to perform algebraic and analytical processing on time domain signals as well as various amplitude or time measurements.

Keywords: digital oscilloscope, digitizer, data acquisition.

Introduction

Digitizing oscilloscopes combine analysis functions with excellent measurement fidelity to characterize single shot or repetitive signals, either simple waveshapes or complex modulated signals. These instruments contain a fast acquisition memory, where digitized amplitude data are stored in a consecutive manner by overwriting old data by those recently received from an analog to digital converter. Recording may be stopped by a trigger signal derived from the recorded waveform either instantaneously or after a certain delay time. Digitized data representing the time domain signal are available for processing algorithms.

The complexity of those processing algorithms is only limited by the performance of the oscilloscope calculation system . A variety of digital signal processors, numeric processors or microprocessors offer the possibility to tailor systems for a large field of application areas.

In principle there are two possible methods for signal processing:

a) Real Time Processing

Data are processed as they are digitized. The processing system has to meet the speed requirements of the digitizing rate. The signal may be

recorded continuously.
Examples: Filters, FFT-s etc.

b) Quasi Real Time Processing

One set of data is recorded and stored. A user definable processing algorithm is performed on that data after the recording stops. The processing speed is not related to the digitizing rate. The signal is not recorded continuously. The dead time of the system depends primarily upon the processing speed. If a repetitive recording sequence is selected, an unlimited number of recording and processing cycles may be performed without further user action.
Examples: Waveform arithmetics, scalar functions etc.

As the quasi real time method is independent from the digitizing rate, it is the only one that meets complex processing requirements at a reasonable price.

Architecture

A processing oscilloscope typically consists of the following building blocks (*Fig. 1.*).

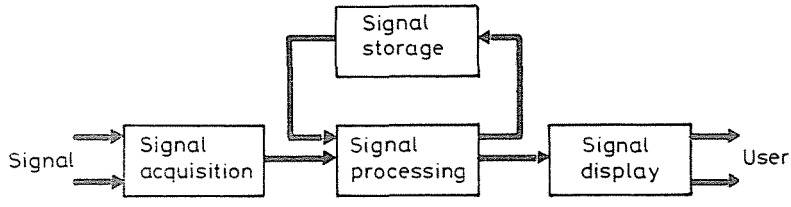

Fig. 1. Structure of a processing oscilloscope

The data flow starts at the acquisition section where the signal is digitized and the data are stored in a fast input memory.

After this memory is filled up, the data set called a 'trace' is transferred to the processing section, where it may be used together with other new recorded or stored traces for calculation of new traces or for calculation of scalar functions as frequency, rms etc.

Additionally, traces may be saved in the storage section for later use.

The last section provides a display of measured and calculated traces and scalars to the user.

The following description will focus primarily on the processing section and give only absolutely necessary descriptions for the other sections.

Acquisition Section

This section transforms the analog signal into a set of numeric values as they are needed by the following sections. One sampled amplitude is represented by an integer value and a scaling value that considers attenuation and gain of the analog circuits (*Fig. 2.*).

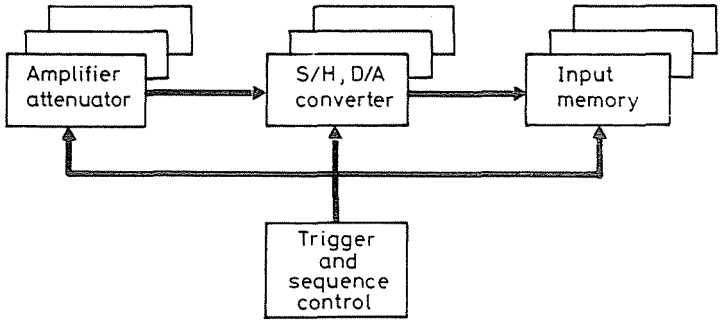

Fig. 2. Acquisition section

The signal to be measured is connected to the attenuator and amplifier section of one of various channels for conditioning the amplitude to the A/D converter window. A number of attenuation steps provides a sufficient signal amplitude to the converter.

For accuracy and special sampling techniques, a sample and hold circuit may be necessary to freeze the signal during conversion time. The conversion rate and the number of bits of the digital result are the main specifications for this section.

The conversion result is stored in a fast input memory, built out of rams fast enough to meet conversion speed requirements or in case of high conversion rates built by means of bank switching circuits.

The trigger and sequencer section selects trigger sources and conditions. It performs recording sequences as roll mode, single shot, repetitive sampling or equivalent sampling. After a trace is completed, the data are made available to the following sections.

Processing Section

The main memory holds traces during processing time. Trace data is received from channels and storage devices. The results that are to be displayed to the user are sent to the display device. A numeric processing unit is optional to increase performance (*Fig. 3.*).

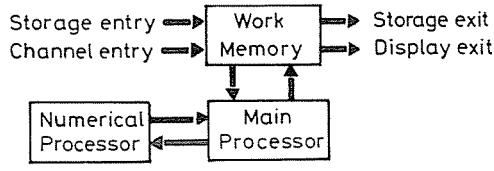

Fig. 3. Processing section

The processing section is dedicated to perform the following actions:

- — accept incoming data from devices (channel, interfaces, cursors, disk, user);
- — synchronize supply of calculation programs with data;
- — support both trace graph and scalar calculation functions;
- — conversion of data to graphic (traces) and numeric (real scalar list) files;
- — provide user definable trace and scalar expressions;
- — high speed transfer of calculated trace and scalar data to storage devices;
- — synchronize high speed transfer of calculated trace and scalar data to display devices;
- — synchronize recalculation of data if parameters change;
- — support standard buffered recording and roll mode;
- — support various recording sequences e.g. 'babysitting';
- — support fast data dumps for "I don't know what I'm looking for" applications.

In fact, the processing system performs synchronized read from a number of files, combination of data and synchronized write to destination files in a specified sequence.

Trace And Scalar Definitions

Trace processing should be definable by the user to meet his individual measurement requirements. By means of trace definitions supplied by the

user, the processing system dynamically creates a control block structure that enables the trace processing task to perform the requested operations.

For each trace and scalar that has to be calculated by the processing system a definition is necessary that covers the rules where the data have to be taken from, what processing has to be performed on the data and where the data have to be sent to.

Traces and scalars are handled with the same programs. In fact, scalar measurements like rms, frequency, cursor measurements are traces with a length of 1.

The origin expression defines where the data have to be taken from and what calculations are to be performed on the data. Using a mnemonic syntax for description only, we can write for example:

c1 · c2 defines the origin for a trace that is calculated as the product of channel 1 and channel 2;

c1 + e : t3 defines the origin for a trace that is calculated as the summation of channel 1 and trace 3 stored on e-disc;

rms(c1) defines the origin for a scalar that is equal to the rms value of the signal at channel 1.

Definition of a destination device enables the user to specify not only the display but storage devices, too, in order to perform fast signal dumps e.g. to a disk or an interface. The destination device definition is part of the trace name, for example:

$d : t1$	specifies a certain trace on the display;
$e : t2$	specifies a certain trace on an e-disc;
$d : s1$	specifies a certain scalar on the display.

By assigning the origin expression to the destination expression, the fundamental trace definition is completed:

$$d : t1 = c1 \cdot c2$$
$$e : t2 = c1 + e : t3$$
$$d : s1 = \mathrm{rms}(c1)$$

Trace Node Structure

A set of trace and scalar definitions is transformed to a control block structure that enables the trace processing task to perform the requested operations.

The control block structure is built from trace nodes. There is one node for each trace or scalar definition.

The following definitions create a trace node structure that will be used in the following explanations:

$d : t1 := c1 + c2$	a trace to be displayed,
$d : t2 := c1 + t1$	a trace to be displayed,
$d : s1 := \text{freq}(t2)$	a scalar to be displayed,
$d : s2 := \text{rms}(c2)$	a scalar to be displayed,
$e : e1 := c1 - c2$	a trace to be stored,
$e : e2 := t1$	a trace to be stored.

The trace node structure shows the access paths from nodes to their operands (*Fig. 4.*).

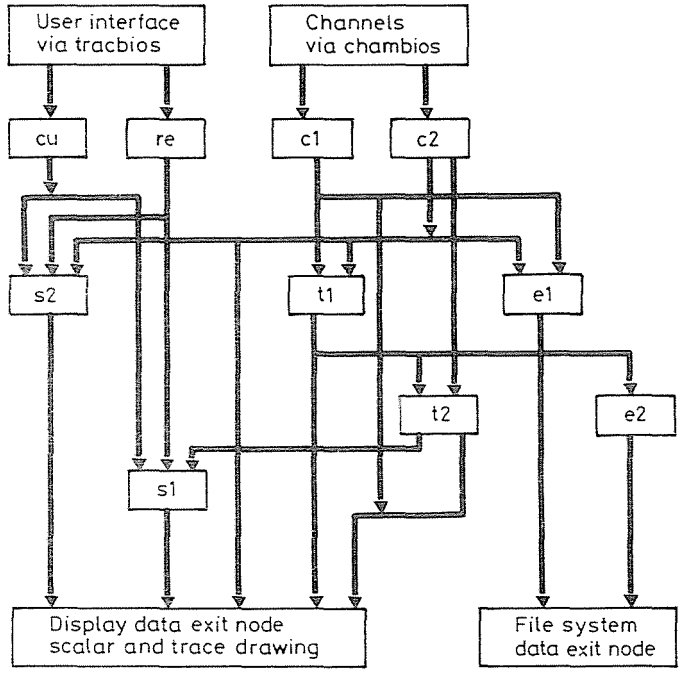

Fig. 4. Trace node structure

Each definition statement creates a symbol, a node and a node task. These tasks perform the following, in the system:

—execute calculation cycles;

—call calculation or input procedures;

—support calculation procedures with references to operand and result data buffers;

—communicate with other trace node tasks to synchronize execution.

—Trace nodes have one of several types:

a) Data Entry Nodes

get data from external devices instead of calling calculation procedures like calculation nodes. This tasks call driver routines that get data via e.g. channelbios calls or, in case of cursor and reference, via call of cursor bios.

Examples:

$$c1(= \text{channel1}),\ c2(= \text{channel2}),$$

$$C(= \text{cursor}),\ R(= \text{reference}).$$

b) Calculation Nodes

call calculation procedures that process data from operand nodes to result buffer. They have nearly the same data and task structure like data entry nodes.

Examples:

$$t1(= \text{trace 1}),\ t2(= \text{trace 2}).$$

c) Data Exit Nodes

have all the nodes as the operands that are attached to their device. We define two tasks, one for display and one for destinations accessible via file system calls to handle the data transfer.

Examples:

$$d(= \text{display}),\ e(= \text{edisc}).$$

Synchronization Requirements

Each time data from input comes in or cursor lines are moved, or parameters like zoom or position are changed, the user wants to have data on the screen updated. However, if inputs are triggered by the same trigger condition one only wants to get screen update as soon as all record synchronized

channels are ready. To obtain these features it is necessary to define three synchronization levels:

a) Recording Synchronization

All recording units with the same trigger condition must use a single 'new recorded data' signal. Channel bios has to manage this synchronization level.

b) Calculation Synchronization

A new calculation cycle starts like a calculation wave at the data entry trace nodes like channels and cursors, runs through the whole trace node structure and is completed at the data exit trace nodes like display or e-disc.

To achieve a simple synchronization mechanism, calculation waves start at all entry nodes whether they have new data or not. In order not to waste calculation time, trace nodes run synchronization only cycles if none of their operands has new data.

Calculation cycle starts if new data at channels is available, cursors are moved or calculation, storage or display parameters are changed.

c) Display Synchronization

Transformation of traces and scalars to graphs and readouts is synchronized by a display task to guarantee related trace and scalar data on screen.

Calculation Cycle Description

Completion of a channel recording or movement of a cursor shall start a new calculation cycle as soon as the predecessor cycle is finished. When a new cycle starts, one, some or all data entry nodes may have new data. A new data message is issued.

Data entry nodes check whether successor calculation node tasks used previous data of data entry node. If so, new data may be filled into data buffer of data entry node.

To achieve a one-pass synchronization structure, all data entry nodes participate at a new cycle, whether they have new data or not. If there is no new data for an individual data entry node, it runs a synchronization-only cycle. Otherwise they call a channel or cursor driver that manages input of new data of their related device.

Data entry nodes inform successor tasks that new data are available. The calculation tasks now start their own cycle.

A calculation node may start a cycle as soon as all attached operand nodes issued a new data message and all attached clients signalled the use of

the former result. The calculation nodes then call calculation procedures that read the operand data and write the result of the new node. The next step is to send a new data message to all successor or client nodes, respectively. The loop is closed as the calculation node waits for a new data message from its operands.

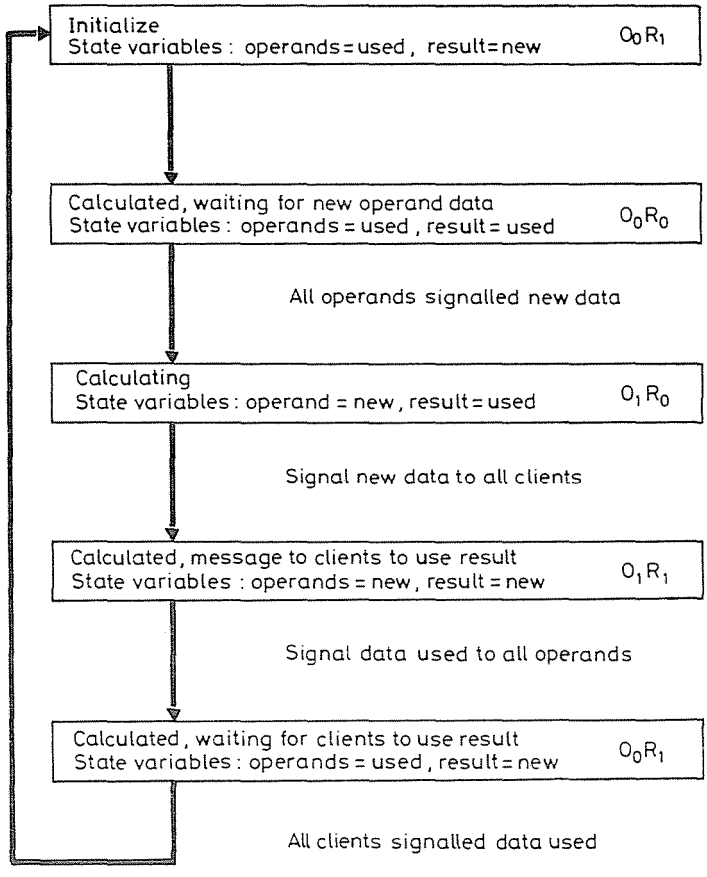

Fig. 5. Trace node task states

In fact, data entry node tasks and calculation node tasks are identical. The only difference is the kind of procedure they are calling for data processing.

Data exit nodes may handle operand data in a synchronous or in an asynchronous way. E-disc devices may copy operand result data as soon as

it is available, display devices may wait until all operand tasks are ready and then display traces and scalars all together.

Trace node tasks may be in one of four states. The actual state is reflected by two state variables: O, R representing the state of operand and result data, respectively. A loop through all states results in a gray-coded state variable pattern (*Fig. 5.*).

Intertask Communication

Task synchronization and communication are performed by means of real time operating system utilities. The task oriented structure is supported by modern microprocessors and guarantees high performance and software reliability.

Display Section

Display section is the interface for the user where he expects to examine the results almost immediately and in high quality.

Screen update rate and screen resolution are the main specifications in this section.

The traces may be manipulated by means of zoom and interpolation functions.

Addresses:

Dipl.-Ing. Dr. Hans FÜRST
Institut für Elektronische Messtechnik
Technische Universität Wien
Gusshausstr. 25, A–1040 Wien, Austria

Dipl.-Ing. Herbert FLECK
TRACE Elektronische Geräte GmbH
Tannhäuserpl. 2, A–1050 Wien, Austria