Powerful timing generator using mono-chip timers: An application to pulsed nuclear magnetic resonance

Hervé Saint-Jalmes

Université Paris XI-Institut d'Electronique Fondamentale Bâtiment 220, 91405, Orsay, France

Yves Barjhoux

Thomson-CSF-Groupe Activités Médicales 48, rue Camille Desmoulins, 92130, Issy-Les-Moulineaux, France

(Received 5 March 1981; accepted for publication 13 June 1981)

We present a 10-line–7-MHz timing generator built on a single board around two LSI timer chips interfaced to a 16-bit microcomputer. Once programmed from the host computer, this device is able to generate elaborate logic sequences on its 10 output lines without further interventions from the CPU. Powerful architecture introduces new possibilities over conventional memory-based timing simulators and word generators. Loop control on a given sequence of events, loop nesting, and various logic combinations can easily be implemented through a software interface, using a symbolic command language. Typical applications of such a device range from development, emulation, and test of integrated circuits, circuit boards, and communication systems to pulse-controlled instrumentation (radar, ultrasonic systems). A particular application to a pulsed Nuclear Magnetic Resonance (NMR) spectrometer is presented, along with customization of the device for generating four-channel radio-frequency pulses and the necessary sequence for subsequent data acquisition.

PACS numbers: 84.30.Wp

INTRODUCTION

As the complexity of digitally-controlled systems increases, a growing need appears for real-time command and control tools such as automatic generators of elaborate logic sequences. Computer-controlled pulse-sequence generators and timing simulators are now used in many applications, e.g., development and test of circuit boards and integrated circuits, waveform generation, disk controllers, radar and ultrasonic systems. Analytical instrumentation also makes a wide use of these devices, in particular pulsed spectrometers for Nuclear Magnetic Resonance (NMR).

A sequence generator can be defined as a system providing n output logic (bistate) lines controlled in real time, and referenced to a master clock. This kind of generator is primarily defined by its speed (i.e., maximum frequency of master clock) and synchronism or accuracy. Many applications also require versatility and logic power (i.e., a great number of output lines, triggering, and loop facilities, etc.).

The designer may choose different approaches to a timing problem when a computer is readily available. First, he can use some output lines of his computer and change their level by program, thus realizing a software timing generator. This method is simple but puts a considerable load on the CPU. Time accuracy is poor (>1 μs) because of the limited speed and lack of synchronism of the computer operation.

Another solution is to use a device with one counter, some programmable memory, and control[1–5] as shown on Fig. 1. The memory holds the events of the sequence to be generated. Each event is characterized by its duration and logic state (i.e., state of the n output lines). This method provides high-speed generators with excellent time resolution ($\sim$20 ns). But as the complexity of the desired sequence increases, so does the amount of required memory and control circuitry. These systems are limited in flexibility and usually offer rudimentary triggering and loop control, if any.

A third way is now offered to the designer with the development of large scale integration (LSI) timer chips. A number of LSI programmable timers are now commercially available with counting frequencies of up to 10 MHz.[6–10] These chips offer a powerful structure: several independent counters, long time counting with concatenation of counters, triggering on edges or levels, intercounter master–slave relationships. These features allow elaborate sequencing with loop control on a given sequence of events, loop nesting, etc. They provide great flexibility at a moderate cost since control functions are programmable from the host computer. Hardware design is thus kept to a minimum.

We present here a 10-line–7 MHz timing generator built on a single board around two LSI timer chips interfaced to a 16-bit microcomputer. The LSI chips and the hardware architecture are described in Chap. I. Control of the timer from the system console is done through an easy-to-use software interface which is presented in Chap. II. Chapter III describes the customization of this timing generator to a specific application: pulsed NMR. Finally Chap. IV provides a comparison

 0034-6748/82/010001-08$01.30

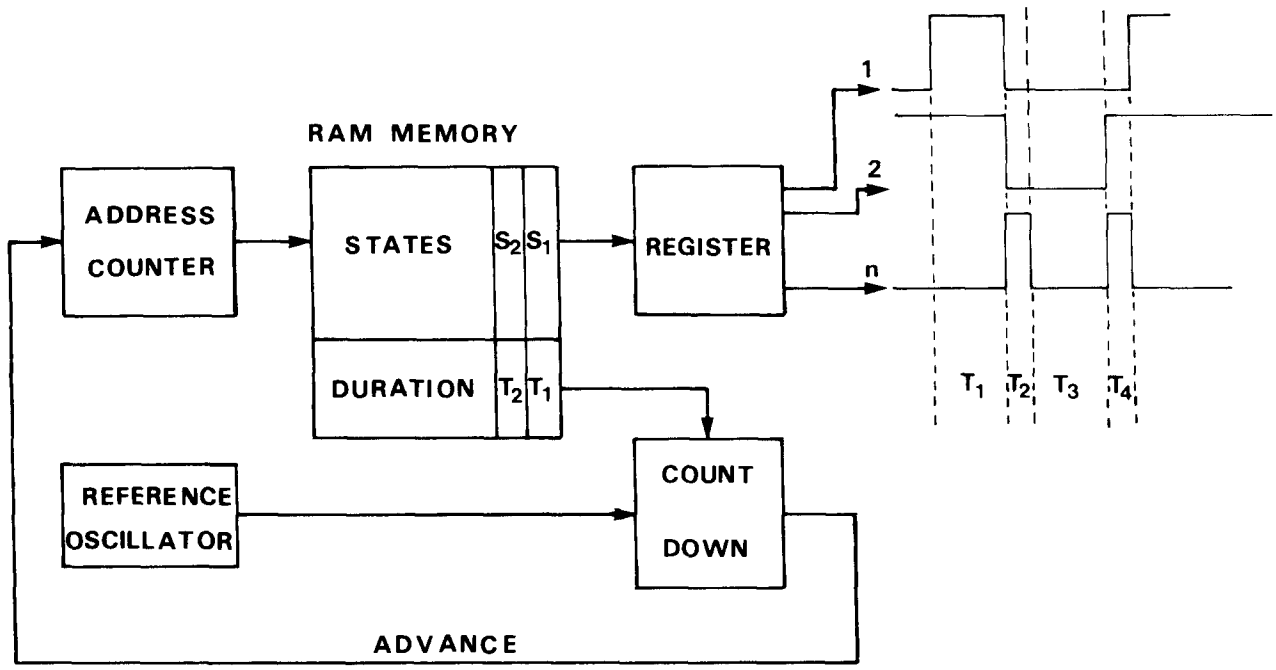

FIG. 1. General operation of a memory-based timing generator.

between such a system and other timing simulators or pulse generators.

I. HARDWARE IMPLEMENTATION: THE TIMER 10 BOARD

In order to optimize the design of the timing generator, we had to select an LSI timer chip. We chose the Advanced Micro Devices Am 9513 System Timing Controller (STC)[10] among several others (Intel 8253, Mostek Combo, Zilog 8036, etc.) for three basic reasons: it has a relatively high maximum counting frequency (7 MHz), five independent counters, and offers great versatility. It allows dynamic reconfiguration: establishing or changing a logic sequence is done by software without any hardware modification.

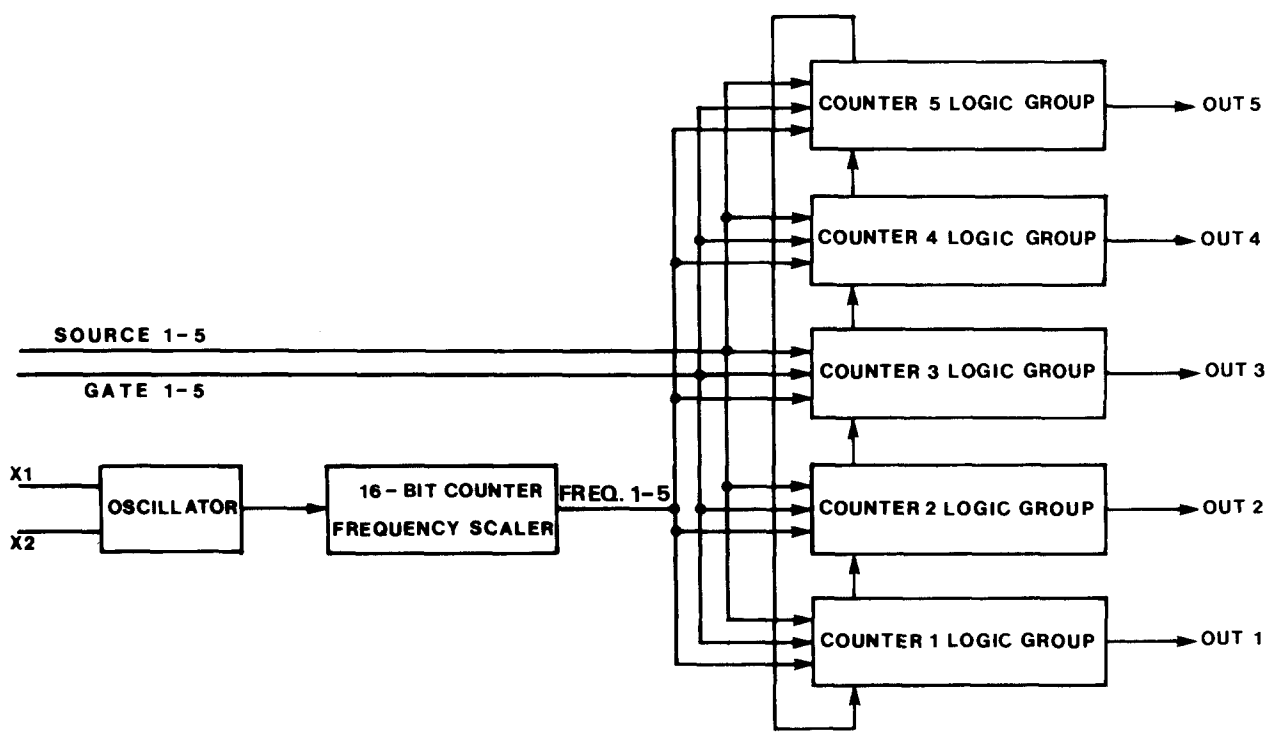

FIG. 2. Block diagram of the Am 9513 timer chip.

The Am 9513 STC chip includes five independent 16-bit counters (see Fig. 2). Each counter has three-state output providing software-controlled polarity and type of output: pulses or levels. A counter can be programmed to count up or down in either binary or BCD and to reload automatically from a load or hold register (see Fig. 3). Internal concatenation of the counters allows to form an effective counter length of up to 80 bits. A variety of counting and gating sources are offered for each counter, either internal or via external pins. An internal frequency scaler allows independent selection for each counter of one counting frequency among five: F to $F/65\,536$ or F to $F/10\,000$, where F is the master frequency. Moreover, five gating inputs may be used to control the operation of each individual counter, gating function options allowing level-sensitive gating or edge-initiated triggering. The data bus used to communicate with the CPU can be software configured in either 8- or 16-bit width depending on the host processor bus width.

Use of two chips (referenced A and B hereafter) on the TIMER 10 board (see Fig. 4) allows ten independent counters (1 to 5A and 1 to 5B) to be simultaneously used and hardware or software interconnected. This provides extensive capabilities for logical event sequencing, loop control of a given sequence, nesting of loops, etc.

The Am 9513 does not support any interrupt capability. In order to permit synchronization of the CPU activity to timer-defined events, we have implemented two interrupt lines on the board. Connection of two counter outputs among the ten available to the interrupt lines allows the TIMER 10 board to notice any particular event to the CPU. The host processor can thus modify any parameter after a specified event has occurred.

The TIMER 10 board has been configured to work with DEC Q bus.[11] This bus standard is widely used (e.g., with LSI 11, 11/2, 11/23, Camac, etc.). Interfacing to Data General systems or to others 16-bit computers is straightforward. Use of 8-bit computers is also possible, although time-register loading is then twice as long. A block diagram of the TIMER 10 board is shown in Fig. 4.

The timing generator is viewed from the host computer as an ordinary memory-mapped peripheral. It uses only five memory locations shown below: A_0: Data to/from chip A; $A_0 + 2$: Data to/from chip B; $A_0 + 4$: Command to chip A; $A_0 + 6$: Command to chip B; and $A_0 + 8$: Control and Status Register (interrupt control), where A_0 is the device Base address which the user selects.

To minimize the interfacing task we have implemented the generator on a DEC DRV11-P interface board which directly plugs into the Q bus. The whole timing generator and its logic are wire-wrapped on the DRV 11-P board (see Fig. 5). The few SSI chips used are standard TTL LS circuits. The only power supply needed is $V_{cc} = +5$ V. The maximum clock frequency specified for the Am 9513 is 7 MHz, but correct operation is maintained up to 8.5 MHz, which gives a cycle time $\delta t = 120$ ns.

As can be seen in Fig. 5, the board includes not only the TIMER 10 chip set, but also some additional circuitry called QUADRA. This optional circuitry is specific to the NMR application considered in Chap. III.

It is to be remarked that some of the LSI timers considered earlier have been implemented on single microcomputer boards. These products are usually designed as multifunction boards. They include a CPU, serial data input/output, parallel data input/output, and timing generation. But these boards generally have limited timing capabilities because of their multiple purpose. They often offer only two or three output lines and a maximum clock rate of a few MHz. These features are not adapted to our specific use for NMR and this is why we designed a timing device. One can expect that in the near future general-purpose Timer

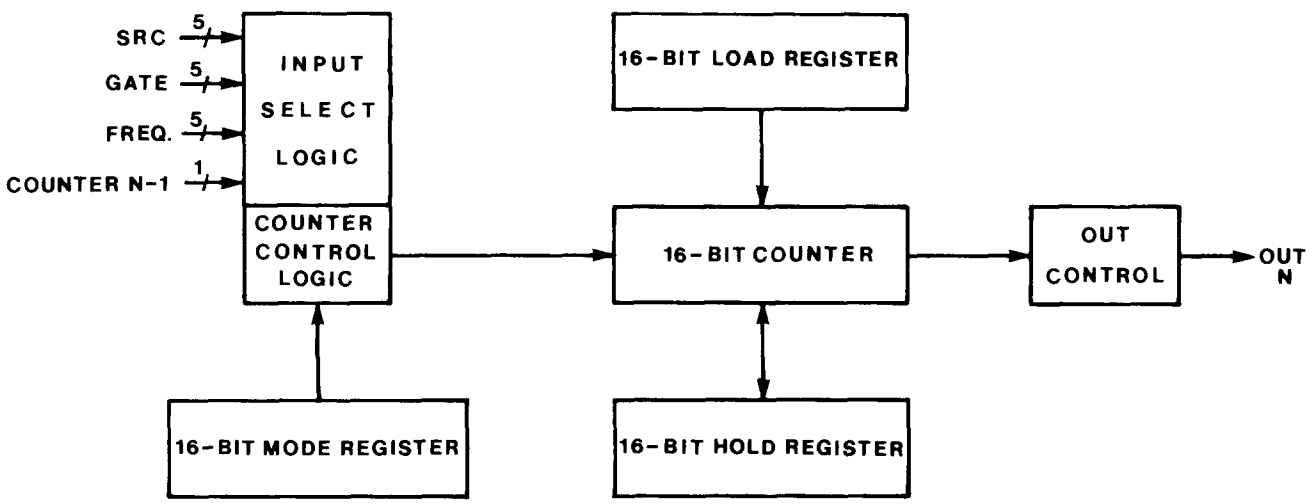

FIG. 3. Internal structure of a counter logic group.

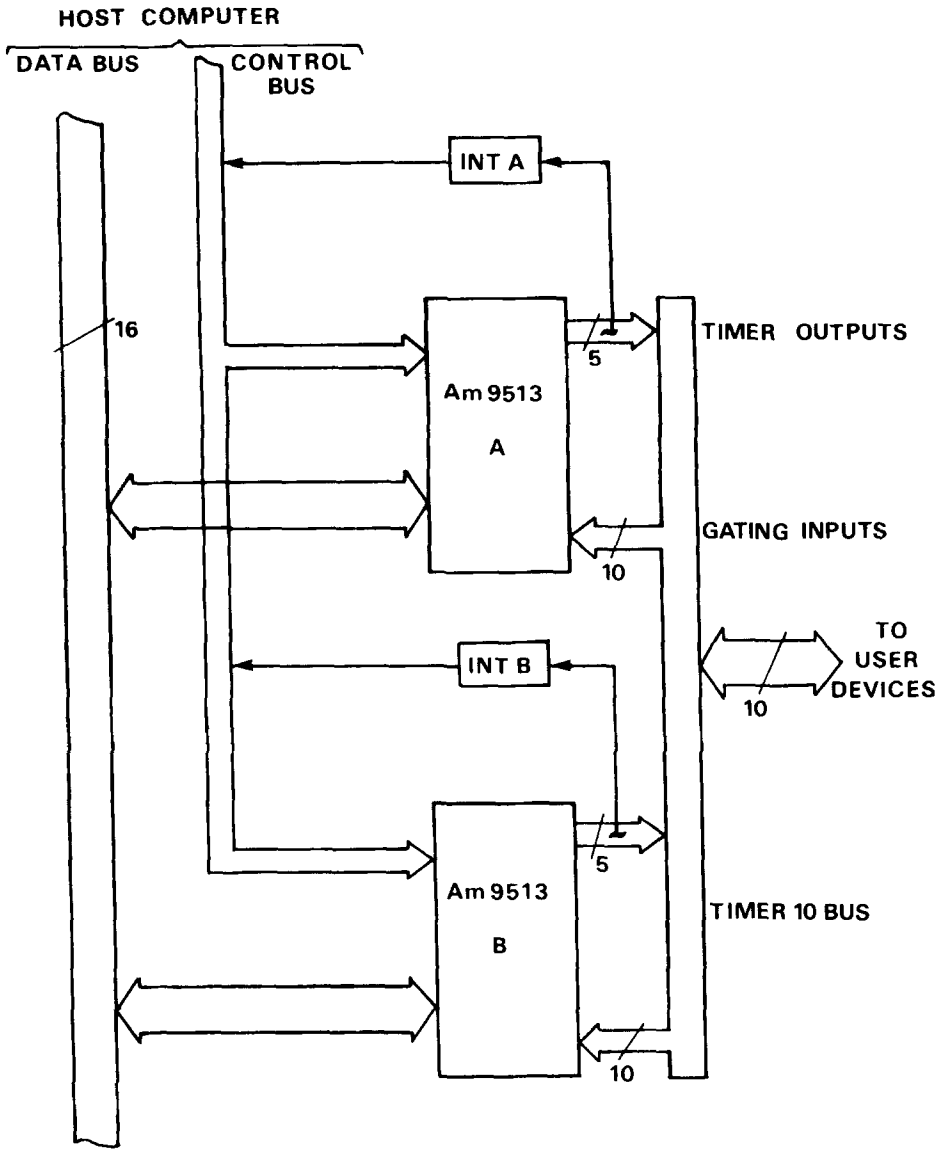

FIG. 4. Architecture of the TIMER 10 hardware.

boards will become commercially available for a number of standard microcomputer systems.

II. SOFTWARE CONTROL: THE TIMGEN INTERPRETER

In order to permit an easy command of the TIMER 10 board we developed an interpreter named TIMGEN. This software runs under the DEC RT11 operating system. It has been written in assembly language MACRO 11.

TIMGEN may be used either interactively from the system console via keyboard entries or it may be called from any FORTRAN program as a subroutine which accepts character strings as arguments.

The TIMGEN language is semisymbolic, allowing two levels of programming. Level 1 (for which one command transfers one 16-bit word to/from TIMER 10) allows the programmer to access all the internal registers and interrupt status register of the TIMER 10. Three command mnemonics are available at this level:

SDY m: send data m
SCY m: send command m
} to timer Y (Y = A or B)

STY m: send status m to the control and status register.

This basic level is reserved to programmers familiar with the architecture of the Am 9513 chip, and is especially useful for debugging purposes.

The second level (for which one command corresponds to several accesses) is designed to allow a nonspecialist user to program a sequence of events in a simpler semisymbolic language. Full knowledge of the internal structure of the Am 9513 chip is not required at this level. The commands operate directly on counter logic groups (see Fig. 3) and use the single mnemonic XnY m. X defines the counter register: M (Mode register), L (Load), H (Hold), C (hold Cycle). nY desig-

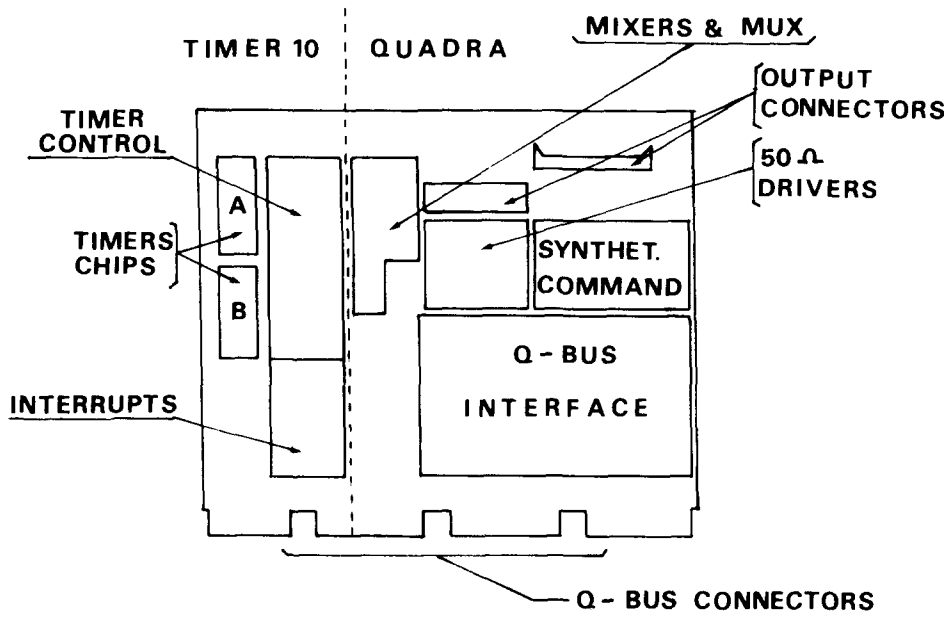

FIG. 5. Implementation of the TIMER 10 board.

nates the target counter among the ten available: 1 to 5A and 1 to 5B. *m* is the data to be sent. For example, to enter value 21 into the counter 3A load register the interpreter accepts the entry: L3A 21. Programming a sequence with level 2 commands is easy: it requires a maximum of 3 commands per active output line.

Moreover, TIMGEN is fully protected against any user error, either of syntactic or logical type. Another useful feature is the possibility of recording a particular sequence and generating it either as the start-up mode for TIMER 10, or as a variable default mode. Once the registers are loaded, a sequence can be started from the computer with an Arm command or via a trigger line on the TIMER 10 board when a particular event occurs. The design of both software and hardware allows one to read or change any timing parameter before or during the execution of a sequence.

The TIMER system (including TIMER 10 and TIMGEN) operates independently from the CPU once the proper registers have been loaded. It is therefore of particular interest to run TIMGEN in a Foreground/Background environment. (The amount of memory occupied by TIMGEN is less than 1.5 Kbytes.) The Foreground is dedicated to occasional modifications of timer parameters or response of the CPU to particular events. Lower priority tasks such as text edition, compilation, or execution of computational programs can proceed in the Background with complete independence from the timer activity. This division of tasks provides transparent operation of the TIMER system.

III. CUSTOMIZATION TO NMR AND PERFORMANCES

The command of a pulsed NMR spectrometer is accomplished by repeated emission of a sequence of pulses to the emitter and receiver parts of the spectrometer.

For this particular design, the emission requires 4 phase channels to be individually gated on or off and subsequently mixed and fed to a 50-Ω power amplifier. The receiver activity is controlled by an on–off signal. Two orthogonal lines are needed to perform phase-sensitive detection of the NMR signal. A last line delivers the sampling pulses required by the analog-to-digital converters.

The task of the TIMER system, in this application, is divided in two. Chip A of the TIMER 10 is devoted to the emission part of the spectrometer, while chip B is dedicated to reception. Specific configuration of trigger inputs and counter outputs for this application requires only a few hardware connections between the two LSI timer chips (see Fig. 6).

Generation of the 4 channels needed for rf emission is realized by a hardware add-on called QUADRA (see Fig. 5). QUADRA operates from a master clock of frequency 4F provided by a synthesizer. The frequency of the synthesizer is programmed in BCD from the computer through the same board. An on-board divider produces four rf signals in phase quadrature at frequency F. The section labeled Gates & Mixers in Fig. 6 combines the four rf channels with TIMER 10 outputs 1 to 4A. The output of the Mixer is sent to the rf power amplifier via a 50-Ω driver.

A typical pulse sequence used in NMR is represented by the timing diagram of Fig. 7. This sequence is programmed and loaded via the TIMGEN interpreter with the commands shown in Table I.

IV. COMPARISON WITH OTHER PROGRAMMABLE SEQUENCE GENERATORS

Timing generators can be classified on the basis of their general architecture. In one class, the timer is built with a number of different counters which may

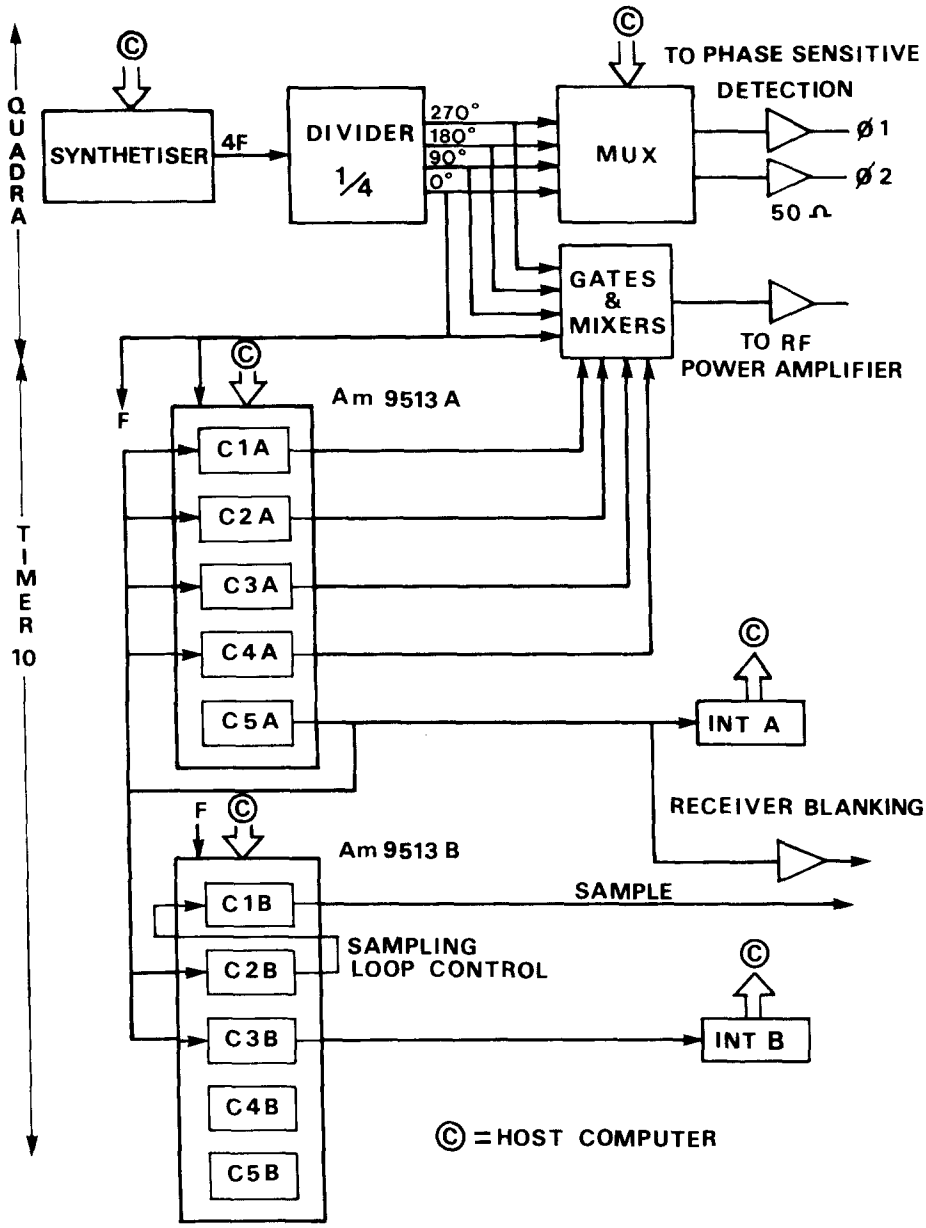

FIG. 6. Functional diagram of TIMER 10 + QUADRA board.

operate independently or may be interconnected. The TIMER system belongs to this class. Some earlier designs also use the same ideas,[12] although they have drastic limitations arising from technological reasons (non-LSI circuitry) or from cumbersome structure. In a second class of generators, which we call "memory timers," the devices are built around a memory of events (e.g., Bruker Z17 C [1], Nicolet 293B [2], Interface Technology RS 680 [3] and similar designs[4,5,13,14] (see Fig. 1).

A comparison between the two kinds of design is summarized in Table II.

Programmable memory timers are generally based upon a single counter implemented with discrete components (most often in ECL technology). Excellent time resolution ($\delta t \sim 10$ ns) can thus be obtained. This high speed is far from being reached by current MOS/LSI chips. However, with LSI timers the minimum delay between events is equal to the time resolution ($\delta t \sim 150$ ns), whereas in generators using memory this delay is necessarily larger than the memory cycle time. For example, one design [4] uses memories with a 250-ns cycle time, and because of various internal delays, this results in a minimum time between events of 800 ns. High-speed memories and faster logic should bring a significant reduction of the minimum duration.

The TIMER system offers ten programmable counters vs only one for the other generators. This large number of independent counters is the key feature of the design. The ten counters allow very powerful loop facilities including nested loops, a feature which is usually not available with memory devices. Another interesting feature is the ability of choosing the source frequency independently for each counter. Thus fast events can be implemented on lines with maximum resolution (counting source is the master clock at fre-

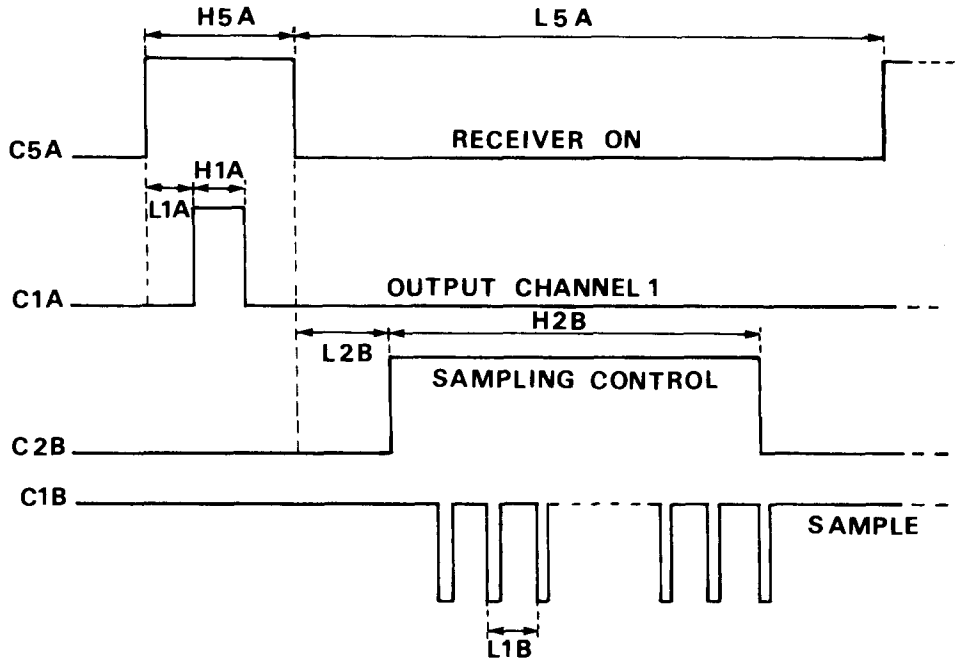

FIG. 7. Timing diagram of a simplified typical NMR sequence.

quency F); events requiring long time counting can use source frequencies scaled down by several orders of magnitude (e.g., $F/65\,536$). Full synchronism can be maintained between all output lines.

All these characteristics of the TIMER system are used with a simple software module: TIMGEN. Programming an elaborate sequence is done simply because of a global approach of the time sequence. The n output lines can be programmed in parallel, using the concepts of loops, triggering, gating, etc. in opposition to the sequential step-by-step programming of memory timers.

V. DISCUSSION

One recent improvement in the TIMER system has been introduced in the software control: a third level of command allows a more global definition of sequences than level 2 and a comprehensive set of powerful instructions is being developed. This permits further simplification of sequence programming for inexperienced users, either from a keyboard, or from a FORTRAN program.

Hardware expansion is easy to implement. When a 10-MHz MOS speed is acceptable the TIMER system constitutes a solution 10–20 times cheaper than memory timers. Thus if a specific application requires more than ten outputs, additional TIMER 10 boards and TIMGEN modules may be paralleled. If necessary, the different boards may be freely interconnected. Moreover the TIMGEN software may be extended to support m timer chips with $m > 2$: the counter vector dimension can easily be changed from 10 to $5 \times m$.

TABLE I. TIMGEN commands used to generate the NMR pulse sequence of Fig. 7.

$F = 4.000$	Master clock (MHz) $F \leq 8.5$
Program emission	
M5A 6142	Load mode register of counter 5A Count repeatedly F/16
L5A 1000	Duration of low state on line 5A
H5A 20	Duration of high state on line 5A
M1A 146142	Load mode register 1A, gated by 5A
L1A 3	
H1A 7	Duration of rf pulse
SCA 161	Load and arm C1A and C5A
Program reception, lines 1B and 2B	
M1B 106045	1B = sample line
L1B 2	
M2B 166142	2B = loop control
L2B 4	Delay before sampling
H2B 2000	512 sampling pulses
SCB 143	Load and arm C1B and C2B

TABLE II. Comparison between the counter-based timer system and memory-based timers.

Performances	Timer system	Memory-based timers
Number of independent counters	10	1
Time resolution	$\delta t \sim 150$ ns	$\delta t > 10$ ns
Minimum delay between events	δt	~500 ns
Number of loops	≤9 with nesting	0, 1, or 2
Counting source frequencies	At least 10 (scaling: $F/1$ to $F/65536$)	1 with prescaling
Sequence programming	Load registers	Load memory
Easy to extend	Yes: modular system	?
Compactness	Single board LSI chips	Multiboard system memory + control

The continuous advance of LSI technology implies that in the particular domain of timing simulation and word generation, LSI chips will gradually take over discrete logic applications because of the powerful combination of flexible architecture and enhanced programmability.

[1] Z17 C BRUKER Pulse Generator, Bruker Physik AG, Karlsruhe, West Germany.
[2] Model 293 B programmable pulser, Nicolet Instrument Corporation, Madison, Wisconsin.
[3] Model RS 680 Word Generator/Timing Simulator, Interface Technology, San Dimas, California.
[4] F. Caron and R. F. Herzog, J. Magn. Reson. **31,** 357 (1978).
[5] T. A. Case and H. T. Stokes, J. Magn. Reson. **35,** 439 (1979).
[6] 8253 Programmable Interval Timer, Intel Corporation.
[7] MC 6840 Programmable Timer Module, Motorola, Inc.
[8] MK 3886 Combo, Mostek Corporation.
[9] Z 8036 Counter/Timer and Parallel I/O Unit, Zilog, Inc.
[10] Am 9513 System Timing Controller, Advanced Micro Devices, Sunnyvale, California.
[11] Digital Equipment Corporation, Maynard, Massachusetts.
[12] A. Gianotti, R. Luther, and D. Ziessow, J. Phys. E **13,** 956 (1980).
[13] T. Cosgrove, J. S. Littler, and K. Stewart, J. Magn. Reson. **38,** 207 (1980).
[14] S. V. Kartalopoulos, The Bell System Technical Journal, Nov. 1980, 1599–1607.