Developing TMS320C30-based VXI Instruments

Archimede *University of Salerno*

ABSTRACT

"This document was an entry in the TI DSP Challenge 2000, an annual contest organized by TI to encourage students from around the world to find innovative ways to use DSPs. For more information on the TI DSP Challenge 2000, see TI's World Wide Web site at www.ti.com/sc/dsp_challenge."

Write Last years have seen Digital Signal Processors (DSPs) find place in a higher and higher number of electronic devices, going from simple audio signal refresh modules to sophisticated digital video cameras or other hi-fi products. The phenomenon is even more widespread among measurement instruments: DSP-inside digital oscilloscopes or FFT analysers are today able to update the input signal spectrum more than 40 times in a second; Virtual Instruments based on DSP data acquisition boards show the same output rate of instruments thanks to the sharing of signal processing burden between the host PC and the on-board DSP.

This performance enhancement in digital signal processing is very useful in the realization of automatic stations with real-time tasks (monitoring, fault detection, process control, on-line testing, etc…), both if industrial instruments such as FFT analysers are adopted, and if DSP-based data acquisition systems are suitably programmed to the aim. The latters, even though requiring the measurement software be ad hoc developed, allow the DSP s' features to be fully used by system developers through optimized programming techniques. However, in a framework where several measurement devices are interfaced one with each other and with one or more control units, dynamic performance depends also on the interface system the station is based on. From this point of view, industrial instruments, which are usually equipped with standard interface circuits (IEEE-488 or 1155), offer easier and faster way of communication than PC hosted boards.

On the basis of these considerations a DSP-based measurement instrument was designed and realized in standard IEEE-1155 (best known as VXI). A TMS320C30TM DSP by Texas Instruments (TI™) mounted on a HP E1490C VXI Breadboard is the heart of a hardware platform which is able to implement several signal processing applications, due to the absence of on-board measurement firmware.

The instrument control and measurement software is down-loaded at the start up time via the TMS320C30TM by TI™ emulation port, thus allowing the hardware to be each time reconfigured for a different measurement task. All the measurement applications that do not require more than two 16 bit resolution analog input/output channels at 200 kHz of maximum sampling/generation frequency, can be implemented in the DSP software. The communication of the instrument on the VXIbus is granted by a standard register-based interface circuit that is controlled by the TMS320C30TM by TI™ via a custom circuit ad hoc designed for the purpose.

To make easier the development of new applications based on the hardware platform, a programming environment was realized in ANSI C language where the device command syntax is defined and all the C functions to be called to manage peripherals or I/O devices (such as input/output registers, DACs and ADCs via serial ports) are included in suitable libraries, thus letting the user deal only with the signal processing algorithms.

Keywords: DSP, VXI Instruments, register-based protocols, software development, environments.

Contents

Figures

Tables

The hardware platform architecture

The HP E1490C Breadboard Module is a C-size VXI device that provides A16/D16 register-based backplane interface circuitry and metal shields to enclose the printed circuit board. Any VXI mainframe can communicate with this module that has the following main features: decoding the switch-selected logical address in the VME address space; selecting one of the four VXI Configuration registers for 16 bit read/write operations addressed by the (master) VXI Controller; transferring the 16 bit word of the read addressed register on the VMEbus data lines via an internal data bus; transferring the 16 data lines word in the write addressed register via the internal data bus. ID and Device Type Register are read only registers mapped at 00h and 02h address in the A16 VME memory space of the module, while at address 04h are mapped both the Status (read) and Control (write) Registers.

Further five addresses (06h, 08h, Ah, Ch, Eh) are decoded by the interface even if they do not correspond to registers in the interface module. Their addressing is signalled by the module to the custom circuitry by tying low a corresponding output lines (REG0*,..REG4*). The LATCH* line is also tied low in case of write addressing to asses that the 16 bit word is available on the module Internal Bus, while line READ* must be tied low to enable user data placed onto the Internal Bus, in case of read addressing [1].

In the part of printed circuit board that is made available for the wire-wrap custom circuitry find location the followings: i) DSP Section, ii) Data Acquisition Module and iii) Custom Circuit (including the five device dependent non standard registers).

DSP Section.

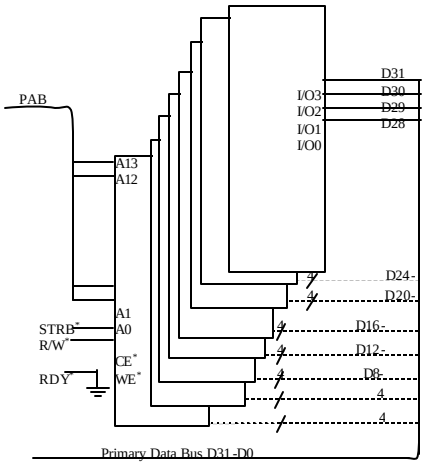
It includes: TMS320C30, Clock and Reset Generators, External Memory, and Emulation Port.

A TMS320C30-GEL PGA package (TI™) is mounted on a 225 pin ZIF socket (Yamaichi™). All ground and power pins are directly featured by corresponding lines of the VXI bus via the P1 and P2 connectors of the Breadboard Module.

A 30 MHz crystal oscillator provides the external clock to the TMS320C30™ through the CLKIN input pin. The reset signal generation exactly follows the 12.4.2 example in [2].

16K x 32 bit of external memory are interfaced to the Primary Bus. Zero wait-state interface to static RAM is granted by the use of n.8 Cypress Semiconductor's CY7C164-25VC 16K x 4 bit 25 ns CMOS static RAMs. The use of the first 14 lines of A23* locates the RAM at addresses 00000h through 3FFFh in external memory (STRB* active), and STRB* establishes the CE* controlled write cycle with RDY* tied low (see Figure 1).

Figure 1. TMS320C30 Interface to Cypress Semiconductor CMOS Static RAM



The TMS320C30™ Emulator is based on a controller card (XDS 510) that must be connected to the TMS320C30™ emulation serial port through an active buffer POD. In this case, the XDS card is installed in a ISA slot of the embedded VXIbus Controller, and the signals shown in Table 1 are supplied to a 12-Pin header positioned on the instrument front panel, allowing the use of the emulation connector of the POD.

Table1: Header signal descriptions

Signal	Description	TMS320C30 Pin Num.
EMU0	Emulation pin 0	F14
EMU1	Emulation pin 1	E15
EMU2	Emulation pin 2	F13
EMU3	Emulation pin 3	E14
GND	Ground	
H3		A1
PD		

Data Acquisition Module

It provides two analog input channels (± 2.75 V, 16 bit, 200 kHz maximum sampling frequency) and two analog output channels (± 3.00 V, 16 bit, 500 kHz maximum generation frequency). The DSP102 and DSP202 from Burr Brown are zero glue-logic interfaced to the TMS320C30 by TI™ as shown in 8.2.13.3 of [2]. The former is interfaced to the receive side, the latter to the transmit side of the full-duplex Serial Port0 of the TMS320C30 by TI™. Both are hard wired to run in cascade mode. The use of TCLK0 as CONV signal for both A/D and D/A means that a difference between sampling and generation software can be only implemented in the peripheral control software. In order to improve the analog input characteristics of the module, signals are input to the VINA and VINB pins of the DSP102 A/D via a NE5532 op-amp configured as double voltage follower.

Custom Circuit

It has been designed to allow the communication between the VXIbus Controller and the TMS320C30 by TI™ through five non standard 16 bit registers: Command, Hi-Dinput, Lo-Dinput, Hi-Doutput, Lo-Doutput. The Command Register is used by the VXIbus Controller to provide the instrument with 16 bit device commands whose coding, depending on the measurement application, is made on the basis of a structure defined in the software development environment. Dinput (High and Low) and Doutput (High and Low) Registers allow the VXIbus Controller to respectively write and read up to 32 bit data words to and from the TMS320C30.

Five couples of 8bit tri-state latches (74ALS373 by by TI™) realize the non-standard registers in the Custom Circuit. The registers are mapped in the A16 VME memory space of the VXI module at corresponding offsets (06h, 08h, 0Ah, 0Ch, 0Eh) through suitable connection with the REG#* lines. REG0* and REG1* drive the OD of Hi-Doutput and Lo-Doutput; REG2*, REG3*, and REG4* (all in NOR with LATCH*) drive the LE pin of Command, Hi-Dinput and Lo-Dinput respectively.

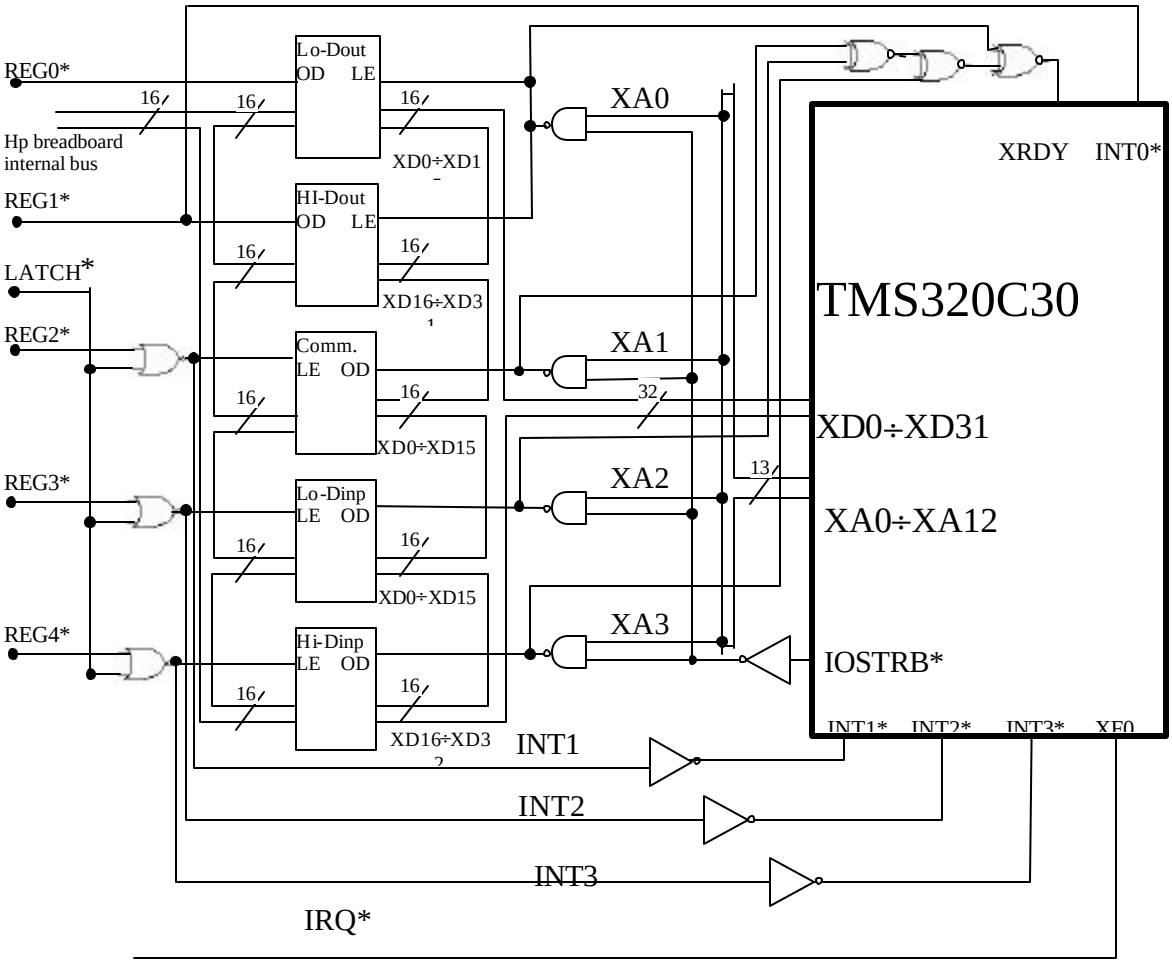
The 16 data output pins of the first two couples of latches and the 16 input pins of the last three couples of latches are in parallel connected to the 16 lines of the HP Breadboard Internal Bus. On the other side, Registers are interfaced to the TMS320C30 Expansion Bus using IOSTRB* cycles. Since TMS320C30 works on 32bit words, the four Doutput latches are seen from this other side as a unique 32 bit data output register. The Register's select is then driven by XA0÷XA3 which map the four registers at 804001h (Doutput), 804002h (Command), 804004h (Lo-Dinput), and 804008h (Hi-Dinput) in I/O address space of TMS320C30™ (Table 2).

Table 2. Registers mapped onto the Expansion Bus lines

XA3	XA2	XA1	XA0	Register
0	0	0	1	Doutput(32bit)
0	0	1	0	Command
0	1	0	0	Lo-Dinp. (16bit)
1	0	0	0	Hi-Dinp. (16bit)

In particular, XA0 (in AND with IOSTRB) drives the LE pin of Doutput latches, while XA1÷XA3 (all in AND with IOSTRB) drive the OD pin of Command, Low-Dinput and Hi-Dinput respectively. As for data lines, input pins of Doutput and output pins of Dinput and Command are connected to XD0÷XD31. At end of each word transfer XRDY is tied low by a suitable cascade of XORs (see Figure 2).

Figure 2. The custom circuit



The Custom Circuit is then completed by the connection of the HP Breadboard module to the TMS320C30™ interrupt pins (Table 3), in order to allow an interrupt-based interface protocol to be defined between VXIbus Controller and the TMS320C30™.

Table 3. Line connections between Breadboard and TMS320C30™.

Breadboard	**TMS320C30™**
REG1*	INT0*
REG2* NOR LATCH*	INT1*
REG3* NOR LATCH*	INT2*
REG4* NOR LATCH*	INT3*
IRQ*	XF0

Controller – Instrument Communication Protocol

This architecture of the custom circuit has been designed and realized having in mind a precise scheme of communication between the VXIbus Controller and the TMS320C30™-based instrument. Since the access to standard registers is fully self-managed by the Breadboard interface, the communication scheme is mainly based on read and write accesses to device dependent registers.

The following five accesses are allowed to the Controller: i) write to Command Register, ii) write to Lo-Dinput Register, iii) write to Hi-Dinput Register; iv) read from Lo-Doutput Register; v) read from Hi-Doutput Register.

In i) case device commands (set-up commands or measured data queries) must be transmitted to the instrument. The Controller takes the control of the VME DTB and write a 16bit word to the slave address (0Ah). The slave module of the HP breadboard decodes the address and puts the word on DB0-DB15 lines of the Internal Bus. Then it ties low for one clock cycle LATCH* and REG2* thus driving the latching of the 16bit word in the Command Register. After the word latching, LE goes down and INT1* generates an interrupt on the TMS320C30TM, whose service routine reads at the address (804002h) corresponding to the Expansion bus line XA1 in the memory map of the TMS320C30TM.

The same protocol is followed in ii) and iii) cases but INT2* and INT3* interrupts are generated and XA2 and XA3 are strobed. 16 or 32 bit data words (waveform samples, measurement parameters) are transmitted to the instruments. In the former case only Lo-Dinput is addressed, else both Dinput registers must be addressed in sequence.

Viceversa the TMS320C30™ needs to write a word in the Doutput Register (iv) and v)) if the answer to a query is ready (i.e. measured data) or an error message must be sent to the Controller.

When the 32 bit word is written to the 804001h address the 16 LSB are loaded in the Lo-Doutput Register while the 16 MSB are loaded in the Hi-Doutput Register. Once data are latched and the XRDY is tied low the TMS320C30™ using the XF0 line ties low the IRQ* line of the Breadboard module. The interrupt generated on the VXIbus is served by the Controller by read addressing at first Lo-Doutput Register (06h) and then Hi-Doutput Register (08h). The latter reading causes the generation of INT0* interrupt to the TMS320C30™ tanks to the connection between REG1* and the INT0* pin. This interrupt announces that data transfer is well-ended and that VXIbus Controller is ready for the next transfer.

Software Environment

The realized VXI instrument needs two software types: VXI controller software and instrument software. VXI controller software comprehends communication software and user interface software, while instrument software includes communication software and measurement application dependent software.

VXI controller communication software and instrument communication software implement communication protocol between VXI controller module and realized instrument. To accomplish this task they have to agree the IEEE 1155 communication protocol. The measurement application dependent software is downloaded at the start-up in the instrument RAM via an emulation port.

The software environment realized in this work uses a model based on libraries. This choice allows continuously upgrading, without changing the whole software structure.

Indeed it has been build up two libraries: one for the VXI Controller and another for the TMS320C30™ - based instrument. The main scope of this software environment is to make easy the development of measurement applications for the VXI realized instrument. In this way the developer can use a lot of function to manage the low instrument hardware structure easily. Moreover many useful communication and memory data-structures have been defined, to provide peripherals software drivers to the unskilled developer. The idea of this software environment is the following: the whole instrument can be divided in two different platforms (Instrument and controller) with different layers, different data-structures and functionality. Each layer must control some instrument definite functions. Each layer possesses two software interfaces for the connection with the previous and successive level.

The lowest layer (LAYER0) is the VXI bus, and its rules are defined in the IEEE1155 standard [3]. It grants the physical link between the instrument and the controller. In this layer are coded all the electrical and timing specifics.

The second layer (LAYER1) is the bus interface and it’s available on each platform (Instrument and controller). It is a software substrate, which implements the basic communication functions.

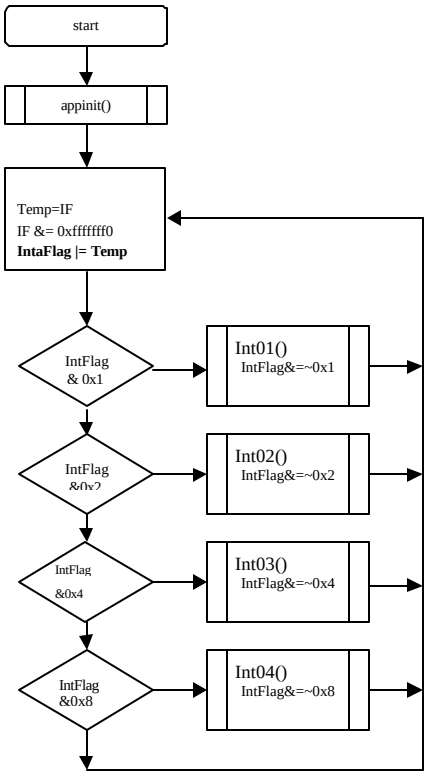
The third layer (LAYER2) implements the functions to allot on the controller the local resources as buffers, SAD, registries, and so on.

The fourth layer is the measurement application software. The samples acquired by the instrument are processed in this level.

As described in the hardware section four non-standard registries either for the VXI instrument addressing space, or for the TMS320C30™ addressing space, have been mapped in the Expansion Bus area. Therefore the developer can use a word array called ExtBus, and four constants called Latch1, Latch3, Latch4 e Latch5 which make him able to read and write data from the four latches using an ExtBus[Latchx] addressing, where Latchx is one of the four defined constants.

Four interrupt routines named int01, int02, int03 and int04 are related with the external events signaled by INT0, INT1, INT2 and INT3 TMS320C30™ pins. The structure of the measurement programs is reported Figure 3. This diagram describes software operation starting after a reset command. In this diagram the *appinit()* subroutine represents the initialization routine for measurement application.

Figure 3. The main structure of the measurement software



Texas Instrument TMS320C30™ was programmed to bufferize interrupts to satisfy them when it's free from other elaboration processes. This choice was implemented because TMS320C30™ communicates with its peripherals using one of its integrated serial ports via an interrupt mechanism, but it is not able to manage nesting interrupts. Therefore this process in considered critical and developed by the use of a particular semaphore. For this aim a special macro was created. It is able to modify the interrupt registry to make the TMS320C30™ sensible to serial-port0 interrupts only and mask other events. Masked events are reported in the IF register, which is continuously polled by the main software. This procedure grant to give high priority to serial ports interrupts, but don't forget the others.

In the following the characteristic and the functionality of the various layers will be described in detail.

The basic software level: LAYER1

In this layer a software section to allow protocol synchronization is present. When an event happens, a jumper programmable interrupt is generated. In this way the controller becomes master and transfer data from one of the instrument's latch. This action realized varying the XF0 line onto the TMS320C30™, is mapped using the INT signal on the instrument board.

The interchange level: LAYER 2

This layer is used to map the instrument resources on the controller, by using the previous layer. In particular the managed resources are SAD with all its functional parameters, instrument Input Buffer, instrument Output Buffer.

To allow normal operation on these resources some commands have been defined.

The command structure is reported in Table 4.

Table. 4 The command structure

15 8	7 4	3 2	1	0
APPCHAR	CMD	Xx	16/32 OP	OPER

The Command registry Most Significance 8 bits (APPCHAR) have been destined to select the application command. The developer can implement these commands using the primitives realized in the previous layer. Bits 7 – 4 are destined to binary coding instrument default commands. Some of these commands are reported in Table 5.

Table 5 Device Commands

Command	Function	Bit 7 - 4
GET_STATUS	Get 16 bit LATCH2 status	0000
APP_CMD	Identifiy application command	0001
SET_INSIZE	Size input buffer	0010
SET_OUTSIZE	Size output buffer	0011
STOP_OUTBUF	Stop transmission from D/A	1100
STOP_INBUF	Stop receive from A/D	1101
SET_FS	set sampling freqeuncy	1110
USER1	User-defined Command	1111

Bit 0 indicates weather the command is able to accept additional parameter or not. If yes bit 1 select 16 bits or 32 bits additional parameters length.

The instrument status register is configured as reported in table 6.

Table 6. Instrument status register

15 5	4	3	2	1	0
Xx	DATA READY	WAIT PARAM	WAIT DATA	WAIT OP32	WAIT OP16

Bits 15 – 5 are protocol free and can be used for non-standard application.

Bit 4 is high if the instrument need to transfer data to controller, and it's reset after the last read. Bit 3 is set when a parametric command is recognized. Bit 2 is set when the instrument must put data onto its input buffer. If the instrument is waiting for a 32-bit parameter bit 1 is set; else bit 0 is set for 16-bit parameter.

The Application level: LAYER 3

In this level measurements application software must be introduced. Here the developer realizes TMS320CDSP software and VXI controller's software. VXI controller software uses the primitive functions realized in the VXI protocol.

Layer2 gives some command to the controller to direct instruments resources, but it take on the possibility of a direct interface with layer3, implemented onto the instrument by using application commands.

References

[1] HP E1490C C-Size VXIbus Register-Based Breadboard Module – User's Manual, U.S.A. E0396.

[2] TMS320C3x User's Guide - revision E, Texas Instruments ed., U.S.A. June 1991.

[3] IEEE std 1155-1992 "IEEE standard for VME Extension for Instrumentation:VXIbus", IEEE, New York, USA. 1993