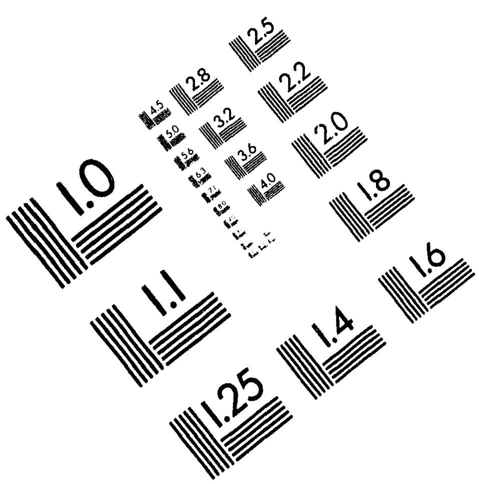
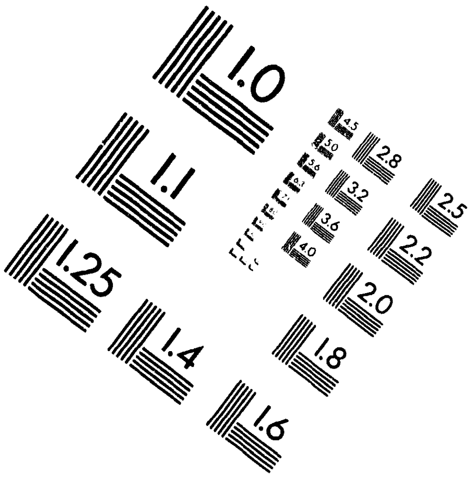


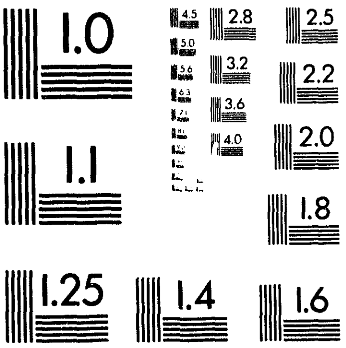
AIIM

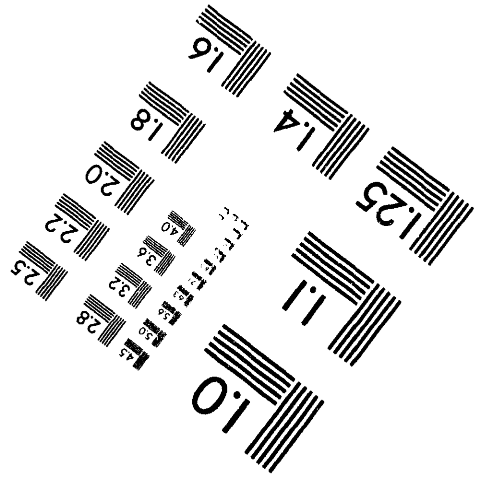
Association for Information and Image Management

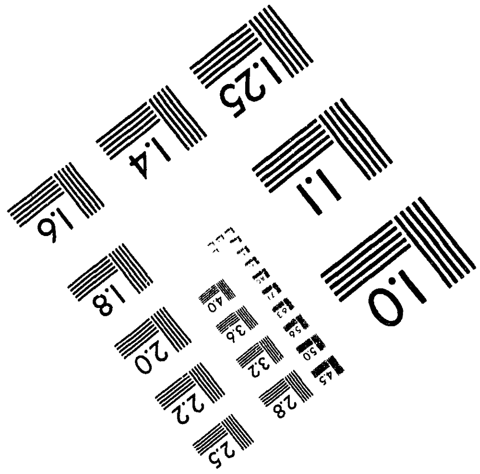
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

301/587-8202









MANUFACTURED TO AIIM STANDARDS

BY APPLIED IMAGE, INC.

1 of 1

ANL/ET/CP-83611
Conf-9408130--1

Real Time Simulator With Ti Floating Point Digital Signal Processor

Kaveh Razazian
James P. Bobis
Stephen L. Dieckman
Apostolos C. Raptis

Energy Technology Division
ARGONNE NATIONAL LABORATORY
Argonne, IL 60439

Invited Paper for Texas Instruments TMS320 Conference.
Houston, Texas, August 5-7, 1994

*This work was supported by the U. S. Department of Energy.

Real Time Simulator With Ti Floating Point Digital Signal Processor

by Kaveh Razazian, James P. Bobis Stephen L. Dieckman, and Apostolos C. Raptis

Abstract

This paper describes the design and operation of a Real Time Simulator using Texas Instruments TMS320C30 digital signal processor. This system operates with two banks of memory which provide the input data to digital signal processor chip. This feature enables the TMS320C30 to be utilized in variety of applications for which external connections to acquire input data is not needed. In addition, some practical applications of this Real Time Simulator are discussed.

Introduction

Simulation is often the first step in implementation of a digital signal processing technique. However, one often experiences incompatibility between simulated result and practical work . A Real Time Simulator (RTS) prototype was designed for both simulation and real time application. In this board, a block of memory is assigned as virtual input and output to a TMS320C30 either by Expansion Bus Interface or the TMS320C30 serial port itself . Due to flexibility of the design, the board may easily be modified and interfaced to other commercial DSP systems. The serial port 1 of TMS320C30 Evaluation module (EVM) board is used communicate to external memory for loading data. In addition, the board is interfaced with MATLAB (Numerical Simulation Package) so that it can receive or transmit data in MATLAB format. During execution, the C30 receives data from memory (virtual input) and the output can either be observed on an oscilloscope through EVM analog interface board at proper serial port location or data can be moved from C30 on-chip array to a PC via the host communication register and displayed by MATLAB graphical interface.

The following topics are discussed in this report,

- An example describing the practical implementation of the RTS

System Configuration Options Overview

The Real Time Simulator (RTS) can be driven by various commercial DSP system. In this case, the RTS is interfaced to the EVM board. The EVM development system allows execution and debugging of application programs. This board consists of three major sections.

I- A microprocessor (TMS320C30), is interfaced directly to 16 K word , 35-ns SRAM.

II- An A/D-D/A convert with 14 bit resolution which may be interfaced directly to serial port 0 for data acquisition .

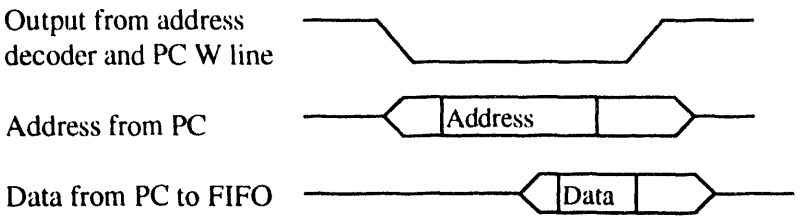
III- A PC host interface which provides 200 Kbytes/s data transfer rate.

Figure 1 illustrates a typical configuration of the EVM and its connection to the RTS. In this configuration EVM board accesses input data from first-in-first-out (FIFO) memory which is located on the RTS.

Figure

Design consideration for RTS

Two parallel-in serial-out FIFO on RTS are linked to the data line of the PC bus which supplies input data to the TMS320C30. The circuit and timing diagram of figure 2 shows the basic interface through which data is written in to the FIFO. Upon power up, the FIFO must be reset with a master reset (RS) cycle. This causes the FIFO to enter the empty condition. The falling edge Write pin ($\overline{W}$) initiates a write cycle. Data appears at the input (D0-D7) before and after the rising edge of $\overline{W}$ is stored sequentially into the FIFO. In this design the write cycle is established on the falling edge of the PC write control ($\overline{W}$) line and output signal of address decoder (74SN138).



Sending data from PC to the FIFO is accomplished by loading the desired data into the PC port location (in this case port location B810h) . All I/O transmissions are handled by software. Data provided by Mothball can be saved in an ASCII file format with proper normalization. A simple I/O C-code routine will load the data into FIFO.

After loading the data into the FIFO, it can be accessed through EVM serial port. This interface is shown in Figure 3 . The EVM board has one unused bi-directional serial port which can be configured to transfer 8 , 16 , and 32 bit data simultaneously . The serial port is controlled by five on- chip registers as follow; Global control register, Transmitter Port Control, Receiver Port Control, Receiver / Transmitter timer control, Receiver / Transmitter timer period, data Transmitter port, and data Receiver port.

Serial port operating configuration can be classified in two categories: fixed data-rate timing and variable data rate timing. In this work, we only concentrate on variable data-rate timing. In this mode Frame Synchronize Receiver (FSR) pulses typically last for the entire transfer interval. The FSR status changes prior to all the data bits being shifted out and the Data Receiver (DR) pin is placed in a high impedance state. Frame Synchronize Transmitter (FSX) input is exactly complementary to FSR.

The serial data in FIFO is clocked out on the rising edge of Clock Receiver (CLKR) . The serial word is shifted out in the order of least to most significant Bit. The serial word width must be controlled by connecting FSR to the NR on the FIFO. For a eight-bit word configuration the (Serial Output Expansion) SOX line is tied high and

High-to-Low transition of the FSR provide the first data transfer. This continue until the FSR line connected to the NR goes high completing the serial word.

A typical command sequence between the EVM and FIFO involve the following steps. First the serial port should be reseted by setting the serial port global control register. This resets the serial port logic and configure the serial port operation modes, include data transfer lengths, and enable the serial port interrupts. The serial port transmit and receive control registers must also be initialized for general propose serial port output operation.

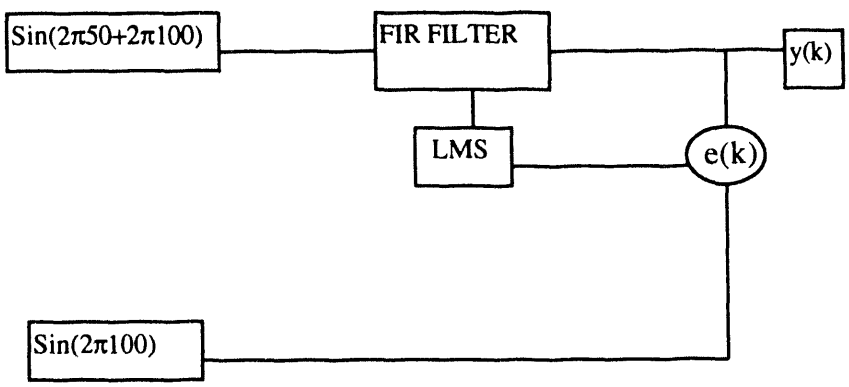
The serial port has access to a dual inputs. Input can be selected under control of the XF1 pin. Setting XF1 to zero (enable FIFO#1) is accomplished by setting the C30 IOF register to 2. FIFO#2 can be enabled by setting the IOF register to 6 (XF1 = high).

There are also four flags provided to monitor the FIFO operation (EF, HF, FF, and AEF). Empty flag (EF) LOW-to-HIGH transition occurs after the first LOW-to-HIGH transition of W for an empty FIFO. HF goes LOW after the falling edge of W following the FIFO actually being half full. FF will remain low while less than one half of total memory is available for writing. FF goes low after the falling edge of W, during the cycle in which the last available location is filled. AEF sets low if 7/8 of total memory is filled. Following table shows amount of words data can be use to control the FIFO capacity and also use these flags to activate retransmit flag.

The high state of empty flag indicates that the FIFO are ready to output the data. An empty condition exists when there has been an equal number of write cycles and read cycles. An empty flag being active prohibits any future read operations. On the other hand, full flag (FF) goes low during the cycle in which the last available location is filled the retransmit (RT) function resets the read address pointer allowing the data that was previously read to be read again. At empty stage, a logical combination of full and empty flags assert a low pulse on RT pin which initiates the internal read pointer to address zero and leaves the write pointer unaffected. The retrasmit capability is intended for use of continues data transfer to C30. A more detailed explanation on TMS320C30 serial port and FIFO operation can be found in [1].

A Practical Application

To demonstrate the operation of system describe in previous section, we implement a simple LMS adaptive filtering problem. Block diagram of configuration is shown in figure 4.



The input of the adaptive filter consists of two real sinusoids of amplitude 1 volt and frequencies of 50 Hz and 100 Hz . The error e(k) is the difference between the filter output (10-Tap FIR filter) and reference input $\sin(2\pi 100)$. We intent to reduce the error of filter by adjusting its impulse response to provide a gain of 1 at frequency of 100 Hz and zero gain at frequency of 50 Hz. This is done by loading $\sin(2\pi 50 + 2\pi 100)$ into the FIFO #1 and $\sin(2\pi 100)$ into FIFO#2. Then TMS320C30 starts executing the LMS adaptive instructions. During execution, input data to C30 is supplied through FIFO#1 and #2. Output data can be observed by an oscilloscope from EVM analog interface board at serial port 0 (AIB) or data can move from C30 on-chip array to the PC via the host communication register and is displayed by MATLAB graphical interface. In this application acquiring input data from A/D was simply replace by a set of FIFO on RTS.

Conclusion

The motivation of this paper was to introduce a new feature that could enhance the performance of TMS320C30. This technique enables a digital signal processor chip to perform in vast variety of applications regardless of input data. Results from the simulation has shown noise in output signal. If 16 bit wide FIFO is applied it can be expected the system will have a good signal to noise ratio.

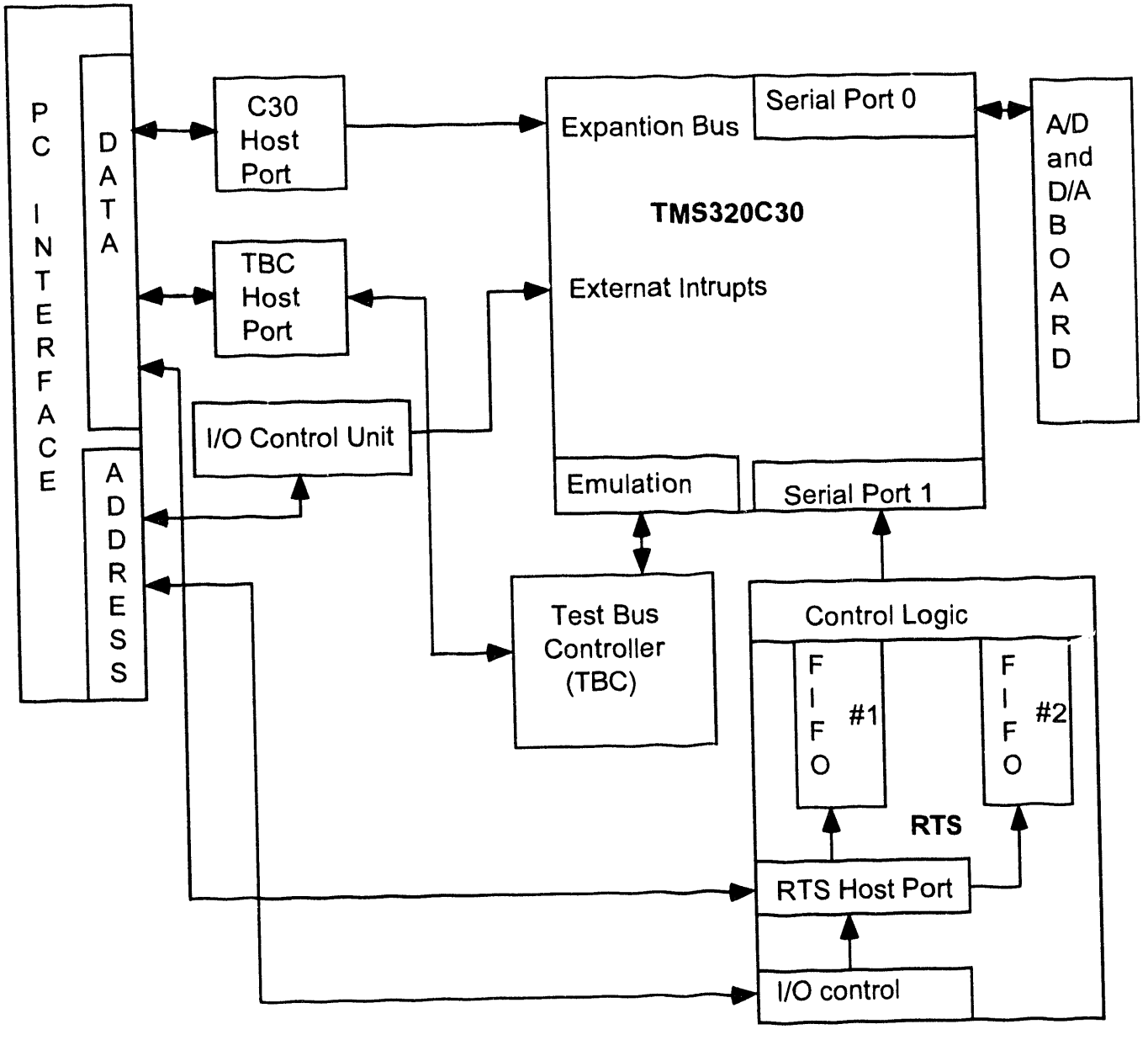
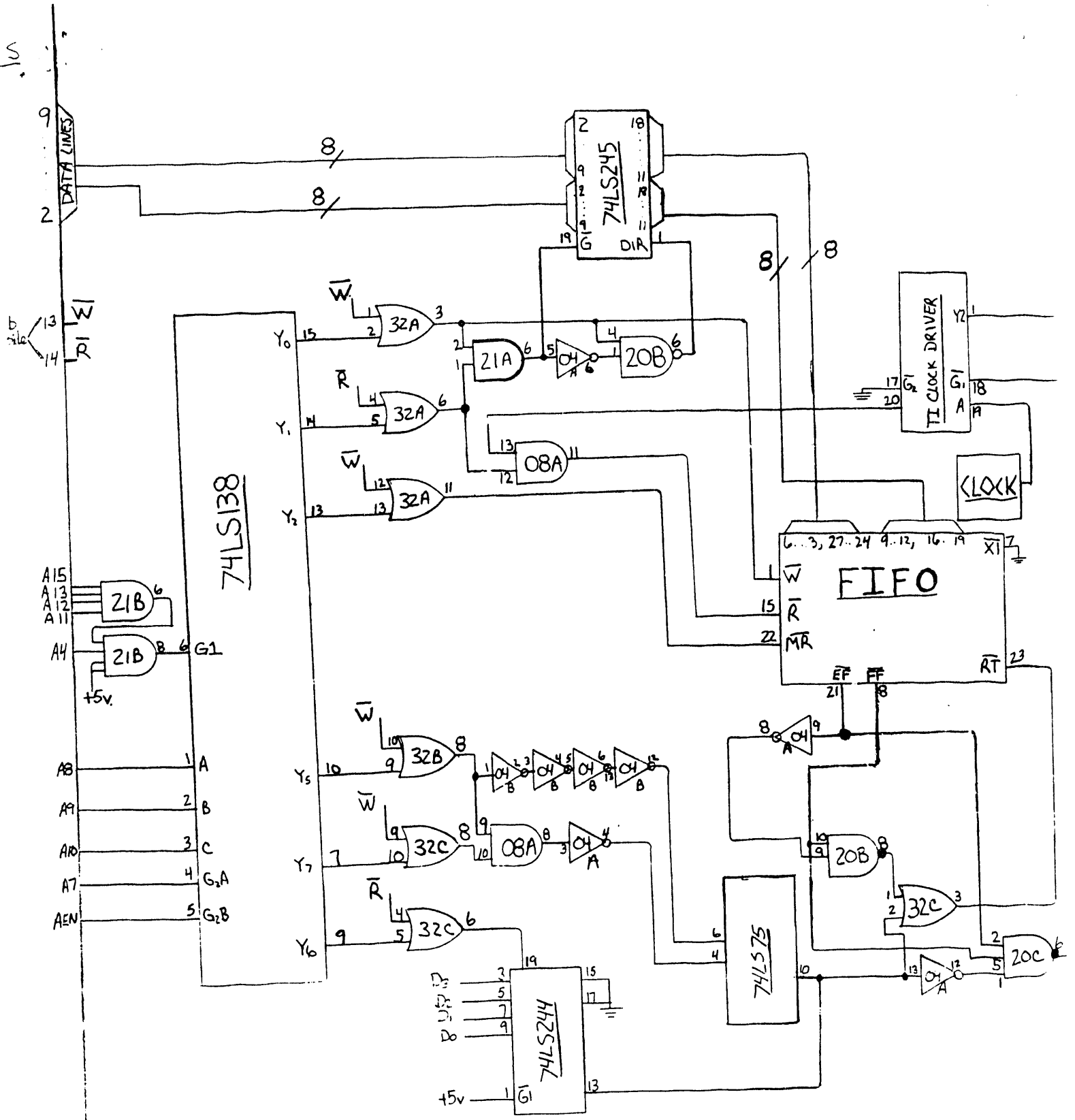


Figure 1. Overview configuration of EVM and RTS



Y_0 = W → BB10
Y_1 = R → B910
Y_2 = RESET → BA10
Y_3 = LOAD TO COUNTER → BB10
Y_4 = ENABLE LOOP → BC10
Y_5 = ENABLE CONTINUOUS → BD10
Y_6 = READ INTERUPT → BE10
Y_7 = DISABLE CONTINUOUS → BF10

DATE
FILMED
9/29/94

END