Design of a microprocessor controlled data interface system
by Dennis Ivan Smith

A thesis submitted in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Electrical Engineering
Montana State University

Abstract:
The subject of this thesis is the design of a microprocessor controlled data interface system. The system presented may be used as a remote data acquisition system or a type of industrial controller. Several of these systems may be connected together to create a complex data collection system.

The system requirements and specifications are relatively few. After the specifications of the data bus are given, several interface circuits are described. This is followed by a description of the controller requirements. A microprocessor was chosen and a controller designed around it. The programming techniques required by this specific system are presented as well as the operation of the entire system.

STATEMENT OF PERMISSION TO COPY

Signature [illegible]

Date May 23, 1975

145

DESIGN OF A MICROPROCESSOR CONTROLLED DATA INTERFACE SYSTEM

by

DENNIS IVAN SMITH

A thesis submitted in partial fulfillment of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

[illegible]
Head, Major Department

[illegible]
Chairman, Examining Committee

[illegible]
Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

June, 1975

ACKNOWLEDGMENT

The design and development of the data interface system presented in this thesis has involved several individuals. The writer would especially like to thank Mr. Dick Weaver of Western Telecomputing Corporation of Bozeman, Montana, for providing suggestions for improving the original design, and for designing several of the interface circuits. Of course, the author is especially grateful to Dr. Donald K. Weaver for his assistance and support of the project.

D. I. S.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The subject of this thesis is the design of a microprocessor controlled data interface system. The system presented may be used as a remote data acquisition system or a type of industrial controller. Several of these systems may be connected together to create a complex data collection system.

The system requirements and specifications are relatively few. After the specifications of the data bus are given, several interface circuits are described. This is followed by a description of the controller requirements. A microprocessor was chosen and a controller designed around it. The programming techniques required by this specific system are presented as well as the operation of the entire system.

CHAPTER I

INTRODUCTION

A data interface system is a term ascribed to various systems capable of obtaining information or data from various devices, such as instruments, sensors, transducers, or computer type memory, and routing this data to other devices, such as another memory or a teleprinter. Exactly how the data are obtained and where they are routed is dependent on the characteristics of a given systems. In one system, the data may be stored in memory for later use while other systems may require immediate use of the data, in which case the data would be routed to the waiting device immediately. The complexity of the system is dependent on the number and particular types of devices interfaced to it as well as the complexity of the controller.

This data interface system has been designed to provide an economical data acquisition system or a control unit. One particular requirement met by including a microprocessor in the controller is that of including the ability to easily preprocess raw data from the interface circuits before sending them to an external device.

BACKGROUND

Many systems have been developed that could be classified as data interface systems. Computers could be classified as such since they interface memory to external peripherals. This interface may be through the central processor or through such schemes as direct memory access or separate I/O processors used in conjunction with multi-port memory. Data acquisition systems could also be classified as data interface systems since they interface transducers and sensors to meters, counters, recorders, etc.

The data interface system described in this thesis has evolved from previous data acquisition systems and present techniques of digitally bussing data through a system. The initial objective of this project was to update and replace the present data acquisition system at Spring Creek. Refer to Williams (6) for a report on that particular system. That system was designed to measure various weather and stream parameters along Spring Creek and to transmit the corresponding data back to Montana State University using telephone lines. The system was remotely controlled by an HP2115A minicomputer located in a laboratory at MSU. All conversions of the raw data to reasonable units took place in the computer. The design of this data acquisition system made it very difficult to allow the computer to do much more than initiate the transmission of the current data. The control program in the computer initiated the system periodically to collect the data and store it in memory. When not collecting data, the program would enable a user to call the computer and access the data stored in memory. The data that could have been collected between system initiations were lost. If the data could be collected continuously and preprocessed on site, the computer's program could be made more flexible as to when to access the system. The computer could then be allowed to access several systems. Many of these factors prompted the development of a new data acquisition system.

Several types of system architecture were investigated as possible structures for the new system. The architecture most applicable to the initial system requirements was a bus oriented structure. The Register Transfer Modules, designed and manufactured by the Digital Equipment Corporation (1), and the standard instrument interface system proposed by the International Electrotechnical Commission (3) were investigated. The resulting system is similar in many respects to each system. The Register Transfer Modules use a sixteen bit data bus in conjunction with a five bit control bus. Control is realized by a hardwired program using evoke modules or by a microprogrammed controller. The International Electrotechnical Commission's system utilizes an eight bit data bus, a five bit management bus, and a three bit handshake bus. The control functions are performed by one of the instruments connected to the bus. Commands are transmitted in the same manner that data are transmitted.

A bus organization was developed by combining several features of both of the above systems. This organization, described in Smith (4), used an eight bit data bus, a six bit control bus, and four nonbussed lines to each channel. Western Telecomputing Corporation made several additions and changes to this initial bus organization to use in one of their new product lines called the WTC Data Interface System. This modified organization was used in this thesis because it was both convenient and functionally adequate to do so. This bus organization uses an eight bit data bus, an eleven bit control bus, and two non-bussed control lines to each channel. The initial design limits this system to sixteen channels. Each channel may have the capability of being both a source and a destination for data transfers. Like the Register Transfer Modules, there is only one channel acting as the source and one channel acting as the destination for each transfer. Control is realized by a centralized system controller which may range from a controller with a hardwired sequence of commands to a microprocessor based controller. With the microprocessor controller, data may be preprocessed before being transmitted to the external world.

ORGANIZATION OF CHAPTERS

This thesis will discuss in greater detail the various aspects and parts of the data interface system. Chapter II will cover the bus lines and the data transfer sequences. The description of several interface cirucits is contained in Chapter III. Controller development for the system will be covered in Chapter IV with a look at the overall system in Chapter V.

CHAPTER II

DESCRIPTION OF THE SYSTEM BUS

This chapter presents a description of the system bus lines and the sequence of signals necessary to perform the data transfers.

DESCRIPTION OF THE SYSTEM BUS LINES

Several lines are necessary to control the transfer of data between channels. Eleven bussed lines control the transfer operations. Two nonbussed lines from the controller to each of the sixteen channels select which channels are to participate in the data transfers. The data is transferred via a bidirectional eight bit data bus. All the lines are ground true to allow OR operations using open collector TTL.

Lines DBØ - DB7 (Data Bus) form the eight bit data bus. Any channel can be selected as the data source or the data destination. The selected source channel puts data on the bus, and the destination channel loads the data from the bus into one of its internal registers. Alternatively, the controller may also act as either the source or the destination.

After the controller has determined the source and destination channels, a single byte (eight bits) transfer is performed using the EDT, DAV, and DAC lines. The controller sets the EDT (Enable Data Transfer) line true to initiate the operation. The selected source then places eight bits of data on the data bus and sets DAV (Data

Available) true. When the destination detects that DAV has gone true, it transfers the data to its register and sets DAC (Data Accepted) true. The controller then sets EDT false, which in turn causes the source and destination channels to set DAV and DAC false respectively.

Generally, a source transmits a string of bytes to the destination and the handshake described above is performed for each byte. Normally the source determines how many bytes are to be transferred. The end of the byte string is usually indicated by setting the SKP (SKiP) line true. The SKP line may also be set true by the destination to indicate that it cannot accept further data. If the controller is programmable, it may use the SKP line for conditional branching.

Two of the bus lines, SCØ and SC1 (Source Channel), are used to address one of four possible separate source registers or to select one of four possible operating modes. The source may optionally ignore these lines or use them in any way that would improve the usefulness of the channel. Two other bus lines, DCØ and DC1 (Destination Channel), similarly control a destination. One of four destinations or one of four modes of operation can be selected.

Any channel can request service by setting SRQ (Service ReQuest) true. This line is OR tied to enable several channels to request service simultaneously. To service the request, the controller sets

ESC (Enable Status Check) true. This converts all the individual SSC and SDC lines (described below) into status lines to the controller. This allows the controller to identify which channels requested servicing and to take the appropriate action.

The remaining bussed line is the RST (ReSet) line. When this line is set true by the controller, all the channels are placed into a known state. This is done when the system is started or when recovering from a power failure.

The SSC (Select Source Channel) and SDC (Select Destination Channel) lines are not part of the bus structure, but are individual lines from the controller to each channel. These lines are used by the controller to select the data source and destination channels for the data transfers, and to bring status from the channels to the controller when ESC is true.

DESCRIPTION OF THE DATA TRANSFER OPERATION

Several operations occur simultaneously during each byte transfer. The following paragraphs describe the effect of the bus lines on the source and destination channels as well as the effects on the controller. A timing diagram for the bus lines involved in a byte transfer is given in Figure 1.

If the source is ready to transmit data, it will set its SSC line true when it detects that ESC is true. When both the channel's

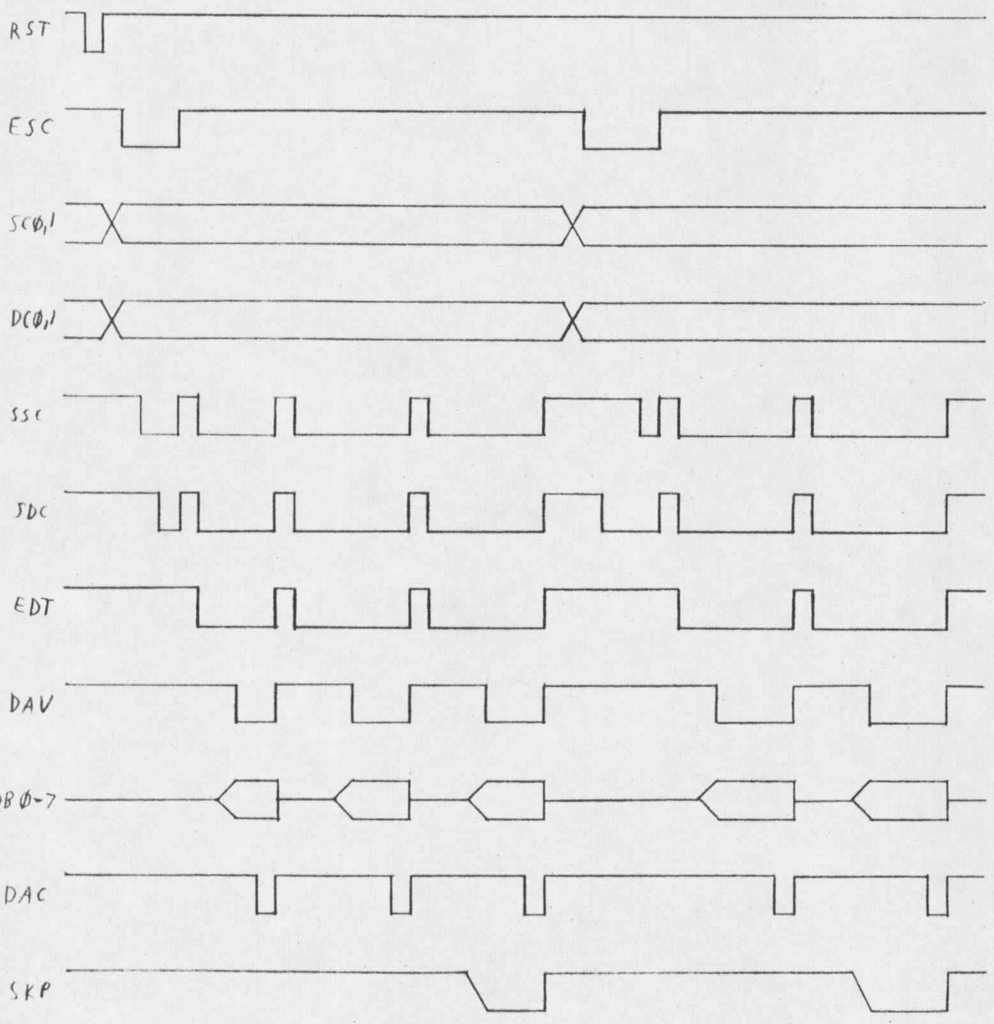

Figure 1. Bus Timing Diagram

SSC and the EDT lines are true, the source register selected by SCØ and SC1 will place its data on the bus. After an appropriate delay to allow the data to settle on the bus, DAV is set true to indicate that the source has placed its data on the bus. When EDT goes false signalling the end of the byte transfer, DAV is set false and the data is removed from the bus. On the transfer of the last byte, SKP is set true during the same interval that valid data is placed on the bus.

If the destination is ready to accept data, it will set its SDC line true when it detects that ESC is true. When both the channel's SDC and EDT lines are true, the destination register selected by DCØ and DC1 will be loaded with the data from the data bus when the destination channel detects that DAV has gone true. Concurrent with the transfer from the bus to a register, DAC will be set true to indicate that the destination has accepted the data from the bus. When EDT goes false at the end of a transfer, the destination sets DAC false. If the destination cannot accept any more data, it will set SKP true when it accepts its last possible byte. This feature must be included in several channels to prevent the destination from "hanging up" the system.

The controller checks the readiness of the channels by setting ESC true and checking the individual SSC and SDC lines. SCØ, SC1, DCØ, and DC1 may be used to check the status of any of the four registers of each source and destinations. To enable a byte transfer, the controller selects the desired channels by setting the appropriate SSC, SDC, SCØ, SC1, DCØ and DC1 lines. EDT is set true to initiate the transfer. When DAC is set true by the destination, the controller responds by setting EDT false.

CHAPTER III

INTERFACE CIRCUITS

This chapter describes several of the interface circuits that have been designed for the system. The first circuit, the counter-timer, will be described in greater detail than the subsequent circuits since the handshake and bus interface circuits are essentially the same for all the circuits and are of primary interest in this chapter.

COUNTER-TIMER

This interface circuit contains both an event counter and an interval timer. The event counter advances one count whenever an input pulse is received from an external source, such as a digital anemometer. The interval timer measures the time interval between any two consecutive pulses. In the case of the anemometer, the counter will hold the wind flow, and the timer will hold the reciprocal of the wind speed.

The counter portion of the circuit, shown in Figure 2, is a four digit decade counter. Whenever the data interface system wants to access its contents, the count is latched into Tri-state latches, which each place their digits onto the bus in turn, the most significant digit first.

The timer, shown in Figure 3, is much the same as the counter,

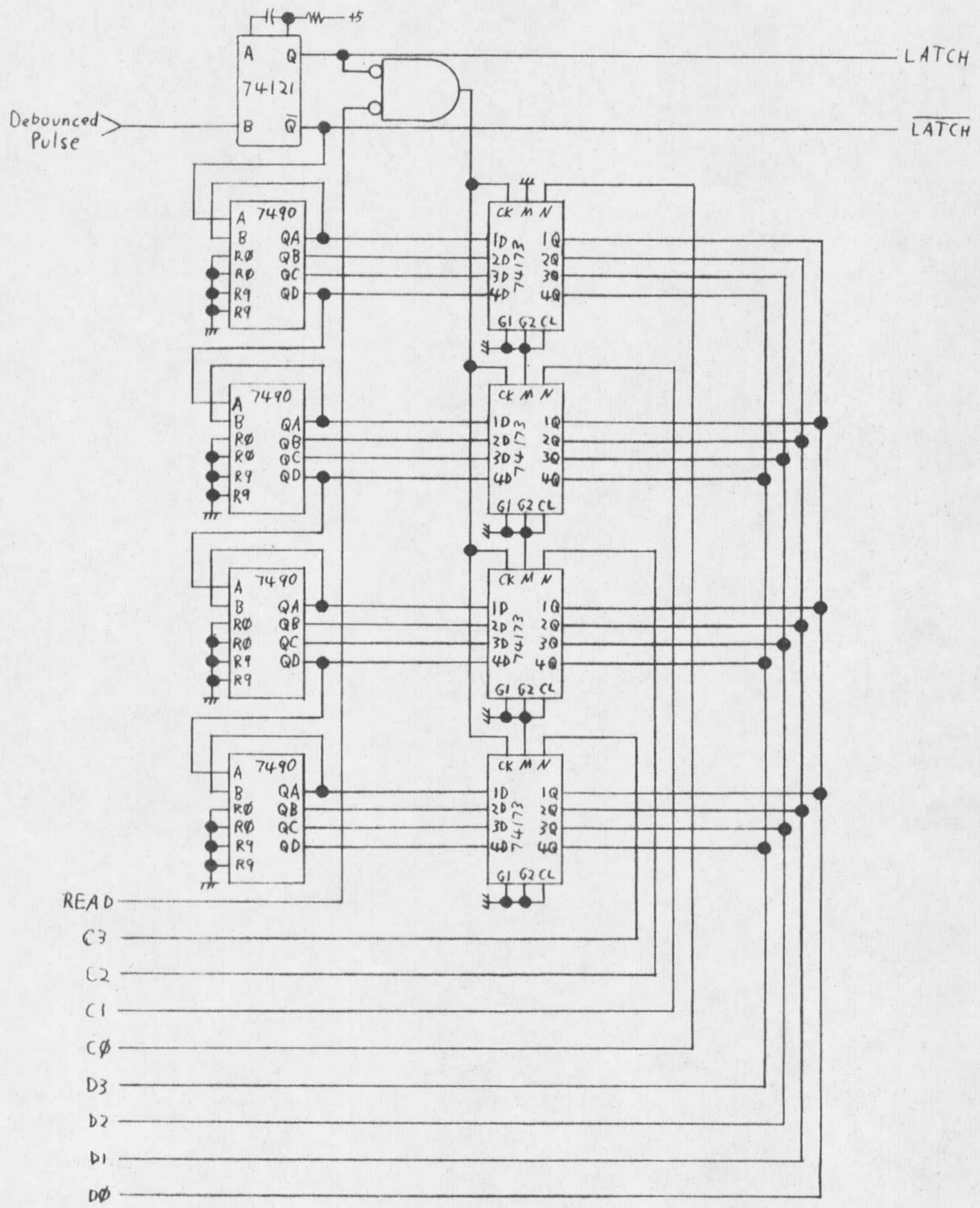

Figure 2. Counter-Timer Circuit: Counter