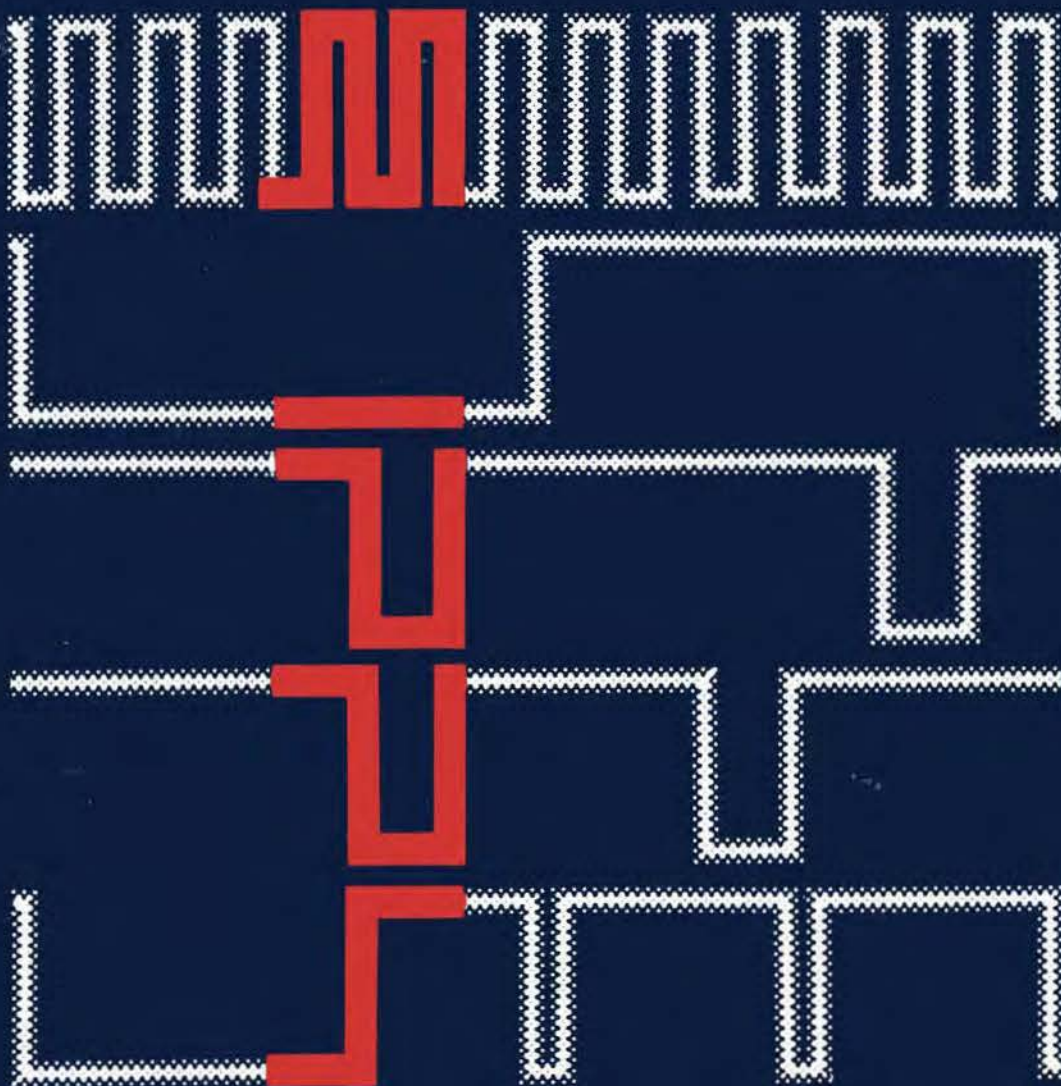


MICROCOMPUTER INTERFACING


Harold S. Stone



Microcomputer Interfacing

HAROLD S. STONE

UNIVERSITY OF MASSACHUSETTS, AMHERST

 **ADDISON-WESLEY PUBLISHING COMPANY**

READING, MASSACHUSETTS

MENLO PARK, CALIFORNIA

LONDON

AMSTERDAM

DON MILLS, ONTARIO

SYDNEY

This book is in the **ADDISON-WESLEY SERIES IN ELECTRICAL ENGINEERING**

SPONSORING EDITOR: *Tom Robbins*

PRODUCTION EDITOR: *Marilee Sorotskin*

TEXT DESIGNER: *Herb Caswell*

ILLUSTRATOR: *Jay's Publishers Service Inc.*

COVER DESIGN AND ILLUSTRATOR: *T. A. Philbrook*

ART COORDINATOR: *Joseph Vetere*

PRODUCTION MANAGER: *Sue Zorn*

PRODUCTION COORDINATOR: *Helen Wythe*

The text of this book was composed in Times Roman on a Mergenthaler 202 by Information Sciences Corporation and was printed by R. R. Donnelley and Sons.

Library of Congress Cataloging in Publication Data

Stone, Harold S., 1938–

Microcomputing interfacing.

Bibliography: p.

1. Interface circuits. 2. Microcomputers—Circuits.

I. Title. TK7868.158S76

621.3819'5835

81-17619

ISBN 0-201-07403-6

AACRZ

Reprinted with corrections, February 1983

Copyright © 1982 by Addison-Wesley Publishing Company, Inc.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Printed simultaneously in Canada.

ISBN 0-201-07403-6

BCDEFGHIJK-DO-89876543 ..

3 / BUS INTERCONNECTIONS

Now that we have covered both the high-level functional description of microcomputers and have looked as well at the lowest-level of implementation details, we will work our way quickly into practical interfacing techniques. This chapter treats the data paths that tie together the processor, memory, and I/O modules of a microcomputer. The strategy followed for these interconnections is similar across all microcomputers; they make use of a general structure that we call a *bus*. A bus is a collection of signal lines that carry module-to-module communications in a microcomputer. In almost all cases bus lines are unbroken, and modules simply tap onto a bus by connecting their respective inputs and outputs directly to corresponding bus signal lines. (The only exception to this rule is for signal lines used for priority resolution, as described later in this chapter.)

For high-performance applications, buses must be restricted in length, thus limiting their use to the short module-to-module connections within a computer chassis. Although these buses can be extended from one chassis to another, performance and reliability suffer as bus length increases. For the longer and lower-performance interconnections, most microcomputer systems rely on special buses, quite separate from their high-speed internal buses, or on other point-to-point connections in order to isolate the high-speed buses from the long physical buses, thereby reducing the degradation caused by excessive bus length. Exceptions to this practice occur in low-speed applications where the internal bus runs slow enough to be extended to a second chassis with little or no performance penalty. With just one type of bus, the system avoids an additional burden of integrating two distinct bus systems and protocols.

3.1 BUS FUNCTIONS

The signal lines that collectively form a bus break naturally into three groups as shown in Fig. 3.1. One group of signals carries the basic information to be communicated on the bus; the other two signal groups guarantee that the information is delivered during a bus transaction. From the earlier discussion of the functional behavior of a microprocessor, we know that the first group of signals carries such information as

1. memory address (or port ID),
2. data, and
3. command type (READ, WRITE, DATA, STATUS).

Since there are a vast number of different buses in use, there is a wide variation in just what information is carried on the first group of lines. Generally speaking, this group carries information that one module needs to convey to another in order to invoke a remote

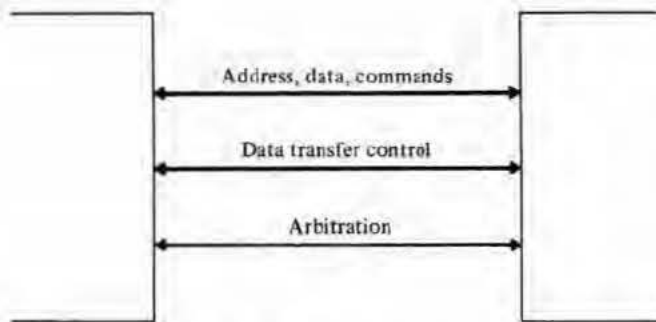


FIGURE 3.1 Bus signal and control lines.

function, response, or change of state in the remote module. In order to pass the information, the bus itself has to be controlled and operated correctly. The other two groups are dedicated to different aspects of the latter function.

The second group of signal lines controls the timing of the data transfer. This group is often called the *data handshake lines*, and contains the signals that dictate when each individual data transfer begins and ends. The handshake lines have a role analogous to traffic lights on a roadway. The handshakes start and stop transactions, and they exert the same functional control on all transactions regardless of transaction type.

The type of transaction comes into play on the third group of lines, the arbitration lines, which give critical transactions priority over less critical ones when deciding what transactions shall access the bus. This third group of lines arbitrates which module gains access to the bus. The necessity for arbitration is due to the inherent problem that occurs when two or more modules attempt to transmit information simultaneously. If module A spews forth a logic 0 while module B attempts to transmit a logic 1 at the same instant of time, we say that there is a *bus conflict*. The signal actually delivered depends on the logic family that drives the bus. A line driven by open-collector drivers moves to the 0 state during any conflict, so that in the given example, the logic 1 output by module B is lost. Then one or both modules lose data at the point of conflict, and what data are lost is unpredictable. Hence, conflicts almost certainly result in a communications failure on the bus. To ensure reliable communication, as a general rule only one module at a time can transmit on the bus, although potentially all other modules can accept the transmission and change state in response to it.

A bus conflict can be more disastrous than portrayed here. For example, what happens when tri-state drivers engage in a bus conflict? In this case, there is a possibility of damaging the bus drivers because the conflict creates a low impedance path from V_{CC} to ground through the output stages of the conflicting gates. The high current through this

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts



Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research



With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips



Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.