THE IEEE S-100 STANDARD HAS ARRIVED!
See Pages 18-44

Also in this Issue
An Introduction to CP/M* BY JAKE EPSTEIN ......................... Page 6
Modifying the SDS VDB-8024 BY JON BONDY ...................... Page 11
Computerized Bulletin Board Systems BY RUSSELL GORR ........ Page 16
An 8080 Disassembler BY WILLIAM YARNALL .................... Page 48

*Registered trademark of Digital Research

and more
Complete Table of Contents on Page 3
Suddenly, S-100 microcomputer systems can easily handle 100 million bytes. Because Morrow Designs™ now offers the first 26 megabyte hard disk memory for S-100 systems—the DISCUS M26™ Hard Disk System.

It has 26 megabytes of usable memory (29 megabytes unformatted). And it's expandable to 104 megabytes.

The DISCUS M26™ system is delivered complete—a 26 megabyte hard disk drive, controller, cables and operating system—for just $4995. Up to three additional drives can be added, $4495 apiece.

The DISCUS M26™ system features the Shugart SA4008 Winchester-type sealed media hard disk drive, in a handsome metal cabinet with fan and power supply.

The single-board S-100 controller incorporates intelligence to supervise all data transfers, communicating with the CPU via three I/O ports (command, status, and data). The controller has the ability to generate interrupts at the completion of each command to increase system throughput. There is a 512 byte sector buffer on-board. And each sector can be individually write-protected for data base security.

The operating system furnished with DISCUS M26™ systems is the widely accepted CP/M™ 2.0.

See the biggest, most cost-efficient memory ever introduced for S-100 systems, now at your local computer shop. If unavailable locally, write Morrow Designs™ 5221 Central Avenue, Richmond, CA 94804. Or call (415) 524-2101, weekdays 10-5 Pacific Time.

*CP/M is a trademark of Digital Research.
IN THIS ISSUE

An Introduction To CP/M - Part 1.............6
by Jake Epstein
CP/M's Structure and Format

Modifying The SDS VDB-8024..................11
by Jon Bondy
Modifications to the VDB-8024 to pro-
vide UCSD PASCAL editor functions

Computerized Bulletin Board Systems........16
by Russell Gorr
An up-to-date guide of many of the
CBBS's across the country

Obscure And Dirty Programming Trick........17
by Bill Fuller
A clever way to exchange the A & B
registers on an 8080/280

The IEEE S-100 Proposed Bus Standard
Introduction..........................18
by Sol Libes
The Standard..........................19
courtesy IEEE

An 8080 Disassembler.......................48
by Bill Yarnall
A disassembler for all 8080 systems

S-100 MICROSYSTEMS is published six times a
year by Libes, Inc. Subscription information
will be found on page 57.

Copyright © 1980 by Libes, Inc.
all rights reserved, reproduction prohibited
without permission.
After ten years of being an amateur computerist I have finally succumbed and joined the ranks of commercial computerists; at least partly. I follow in the footsteps of many who have gone before me.

It had become apparent to me that there has been a significant void in the microcomputer magazine field. There are magazines catering exclusively to TRS-80, Apple, Pet, Heath, Kim, etc. system users and broad based publications such as BYTE, Creative Computing, Interface Age and Kilobaud. The result was that 8080 system users received only very minimal magazine coverage. There was a real need for a publication to cater to S-100 system users. After all, S-100 is far and away the most popular bus-oriented microcomputer system around. Its popularity is on the increase (last year saw the introduction of five new S-100 mainframes), particularly with the adoption of the IEEE S-100 Bus Standard and the availability of the new S-100 16-bit microprocessor boards.

The problem of no S-100 magazine was pointed out to me when two S-100 product manufacturers asked me for advice on where they should advertise their products to effectively reach the largest number of S-100 system users for the advertising dollars they had to spend. After all, magazine advertising is very expensive (e.g. BYTE charges about $2,700 per page) and when only a small percentage of the magazines readers are potential customers it means the advertiser is throwing away a lot of money advertising in the magazine. I really could not help these S-100 suppliers.

It was then that the idea of an S-100 magazine started to take shape in my mind. I spoke to a number of people, including magazine publishers, to see if they were interested in publishing the magazine and succeeded in drumming up zero interest. Basically, they felt that the circulation would not be large enough for it to be financially worthwhile. One told me he felt that the S-100 bus was a thing of the past. Another explained the magazine publishing business to me...that it would take at least three years to make it profitable and it would need a circulation of at least 50,000 subscribers.

I then started to think of publishing it myself. I am no businessman. Moreover, I am the kind of person who likes to take on a variety of projects and not tie myself to projects that take up all my time and run on for years. However, it has become apparent, after approaching many people, that the only way an S-100 magazine will come into existence is if I assume the responsibility.

Many friends, who also believe in the need for this magazine, have pitched in to help me get it off the ground. You will see some of their names on the table of contents. But I will need a lot more help to make it a success. I am therefore asking for you help...subscribe, get your friends to subscribe. Send in articles, letters of criticism, suggestions, etc. Let's make S-100 MICROSYSTEMS "the" forum for S-100 system users.
At Intersystems, “dump” is an instruction. Not a way of life.
(Or, when you’re ready for IEEE S-100, will your computer be ready for you?)

We’re about to be gadflies again.

While everyone’s been busy trying to convince you that large buses housed in strong metal boxes will guarantee versatility and ward off obsolescence, we’ve been busy with something better. Solving the real problem with the first line of computer products built from the ground up to conform to the new IEEE S-100 Bus Standard. Offering you extra versatility in 8-bit applications today. And a full 16 bits tomorrow.

We call our new line Series II™. And even if you don’t need the full 24-bit address for up to 16 megabytes (!) of memory right now, they’re something to think about. Because of all the performance, flexibility and economy they offer. Whether you’re looking at a new mainframe, expanding your present one or upgrading your system with an eye to the future. (Series II boards are compatible with most existing S-100 systems and all IEEE S-100 Standard cards as other manufacturers get around to building them.)

Consider some of the features: Reliable operation to 4MHz and beyond. Full compatibility with 8- and 16-bit CPUs, peripherals and other devices. Eight levels of prioritized interrupts. Up to 16 individually-addressable DMA devices, with IEEE Standard overlapped operation. User-selectable functions addressed by DIP-switch or jumpers, eliminating soldering. And that’s just for openers. The best part is that all this heady stuff is available now! In our advanced processor—a full IEEE Bus Master featuring Memory Map™ addressing to a full megabyte. Our fast, flexible 16K Static RAM and 64K Dynamic RAM boards. An incredibly versatile and economical 2-serial, 4-parallel Multiple I/O board. 8-bit A/D-D/A converter. Our Double-Density High-Speed Disk Controller. And what is undoubtedly the most flexible front panel in the business. Everything you need for a complete IEEE S-100 system. Available separately, or all together in our new DPS-1 Mainframe!

Whatever your needs, why dump your money into obsolete products labelled “IEEE timing compatible” or other words people use to make up for a lack of product. See the future now, at your Intersystems dealer or call/write for our new catalog. We’ll tell you all about Series II and the new IEEE S-100 Bus we helped pioneer. Because it doesn’t make sense to buy yesterday’s products when tomorrow’s are already here.

InterSystems™
Ithaca Intersystems Inc., 1650 Hanshaw Road/PO. Box 91, Ithaca, NY 14850 607-257-0190/TWX: 510 255 4346

Ithaca Intersystems Inc., 1650 Hanshaw Road/PO. Box 91, Ithaca, NY 14850
607-257-0190/TWX: 510 255 4346
AN INTRODUCTION TO CP/M—Part 1
CP/M's Structure and Format
by Jake Epstein
Box 571
Pittsfield, Ma. 01201

This is the first part of tutorial series on CP/M, the most widely used disk operating system for S-100 based microcomputer systems.

CP/M is an operating system written to run on 8080/8085/280 based microcomputers. It was written and is currently being supported by DIGITAL RESEARCH, PO BOX 579, PACIFIC GROVE, CALIFORNIA 93950. Their phone is 408-649-3896. Also there is a users group that provides a vehicle for the exchange of software, and currently, it has a quite large and varied list of languages and utility programs written to run under CP/M. There have been several versions of CP/M with the most current being ver. 2.0, and although at present I do not have this version, I will be getting it in the near future. The system that I have now is version 1.4 which DIGITAL RESEARCH continues to market and support. The prices that I have at this writing are $100 for ver. 1.4 and $150 for 2.0 which includes a FLOPPY DISKETTE and complete documentation.

In the past, CP/M has been limited to run mainly on 8 inch and 5 inch "floppies", for although there have been tape versions, the majority of use has been with disk-based systems. One of the abilities of ver. 2.0 is hard-disk compatibility which opens up the possibility of tremendous mass memory capability on a micro-computer. More on this in future articles. CP/M uses "SOFT-SECTOR" IBM 3740 format diskettes which store approximately 250,000 bytes of data in single density format on 8 inch disks and 70,000 bytes on the 5.25 inch variety, but double density is available on both systems and quad density is available for the 5.25 inch type. Thus, with a 4-drive dual-density system (DUAL REFERS TO THE ABILITY TO READ BOTH SINGLE AND DOUBLE DENSITY) the mass-memory size is 2 megabytes on 8 inch diskettes. Such large memory capability is simply incredible when one considers computer size and price tag.

What really makes this operating system so attractive for small system users, however, is that it is hardware independent in that once a user has it running, he/she can use CP/M software without having to interface it by writing drivers for his/her system configuration. This is made possible because a group of routines that comprise an area of the system called "BIOS" (BASIC INPUT OUTPUT SYSTEM) is implemented when the user first gets CP/M up and running, and then the operating system calls this area when it needs to do any I/O. Thus many combinations of terminal ports and disk controllers can be used. Currently, there are many disk controllers being manufactured, and several worthwhile articles could be written for this column by users of different boards. I have the TARBEll interface, and in a future column, I will discuss the board and my problems in getting it up and running. There are four other areas in the memory map of the system that I will mention here. BDOS (BASIC DISK
OPERATING SYSTEM) has routines that are used by the system to access its system disk drives and thus provides for file management. Next month I will discuss how files are constructed, but for now I will simply list the basic operations provided by BDOS:

- **SEARCH** FIND A PARTICULAR FILE BY NAME
- **OPEN** PREPARE A FILE FOR OPERATIONS ON IT
- **CLOSE** CLOSE A PREVIOUSLY OPENED FILE
- **RENAME** GIVE A FILE A NEW NAME
- **READ** BRING IN A RECORD (1 SECTOR/128 BYTES) OF A FILE
- **WRITE** STORE ON THE DISK ONE RECORD IN AN OPENED FILE
- **SELECT** CHANGE THE LOGGED ON DISK DRIVE

DIGITAL RESEARCH also uses the term FDOS (FLOPPY-DISK OPERATING SYSTEM) which is actually the combined areas of BIOS and BDOS. The CCP (CONSOLE COMMAND PROCESSOR) can be compared to a keyboard monitor, for the user converses with the operating system via this area by using the command syntax of CP/M. In other words, the operator can accomplish various tasks such as list a directory of files on a disk, or execute a program file on a disk simply by using the terminal that is logged on the system. Finally, there is an area of memory that is used by programs that run under CP/M.

This area, the TPA (TRANSIENT PROGRAM AREA), begins at memory location 0100H, and programs are written by the user to run so that they can access areas of BDOS, BIOS, and CCP to use routines already present in the system. Programs accomplish this through various "SYSTEM CALLS" and thus as I stressed above, a program that uses these calls can be run on any CP/M system. Finally, there is an area from 00H to 0FFH that provides for buffers and system memory registers (special storage areas) that are used by the operating system. Below is a memory map of a 16K CP/M system based on an INTEL MDS system BIOS as provided in DIGITAL RESEARCH'S documentation.

In the memory map, TBASE, CBASE and FBASE represent the memory address of the start of each area. TBASE is always equal to 0100 but CBASE and FBASE change with different sized systems. Systems can be made larger or smaller according to the amount of main memory or other

---

THE MM-103 DATA MODEM AND COMMUNICATIONS ADAPTER

S-100 bus compatible

**FCC APPROVED**
Both the modem and telephone system interface are FCC approved, accomplishing all the required protective functions with a miniaturized, proprietary protective coupler.

**WARRANTY**
One year limited warranty. Ten-day unconditional return privilege. Minimal cost, 24-hour exchange policy for units not in warranty.

**LOW PRICE—$359.95—** For Modem AND Coupler plus shipping & handling

**HIGH QUALITY**
-50 dBm sensitivity. Auto answer. Auto originate. Auto dialer with computer-controlled dial rate. 61 to 300 baud (anywhere over the long-distance telephone network), rate selection under computer control. Flexible, software-controlled, maskable interrupt system.

**ASSEMBLED & TESTED**
Not a kit! (FCC registration prohibits kits)

**Call for further information:**
VOICE: (703) 760-3727
MODEM: (703) 760-0930 (300 baud)
consideration, but actually the memory map stays essentially the same with only the TPA changing in size for different versions. One final note on the operating system map is that a program may overlay the CCP or FDOS. What this means is that once the program is loaded from disk, it may use areas of memory previously occupied by the CCP or FDOS as long as they are not needed by the program.

On power up, the operating system is either entirely or partially loaded into memory from a CP/M system diskette via a bootstrap loader sequence. Any diskette that has the operating system on it and is in drive 0 is called the "SYSTEM DISKETTE", and if you have a multiple drive system, only the 0 drive needs to have the operating system on it. Tracks 0 and 1 contain the operating system while a directory of all the files on a disk and their locations and the data files are found on the other tracks (2-76 on 8" disk or 2-35 on 5.25" disk). Sector 1 track 00, the very first sector on the disk, contains a loader program, which when placed in memory from 00-7FH will bring in the rest of CP/M when executed. In order to read sector 1, track 0, a bootstrap loader is used that can either be toggled in via a "front panel" or burned into a ROM and executed. In my system which uses a ROM bootstrap, the entire process takes less than 5 seconds. In some systems, different from mine, the BIOS or parts of BIOS such as disk I/O is placed in ROM also. The BIOS contains boot programs to bring CP/M back in after a program has run, thus after executing a file, if BIOS is not altered, one need only execute the area of BIOS used to bring in the system. Location 000H in memory is reserved for a jump vector to this boot routine, thus to bring in CP/M, all one has to do is to jump to location 00H. If BIOS is altered however, the entire system has to be rebooted from the starting point using a computer reset command to get control via either a ROM monitor, or a CP/M bootstrap that is initiated on reset. The main advantage of jumping to location 00 and not resetting the computer is that the former procedure will not alter the TPA or in other words, the memory located from 100H to CBASE (start of CCP). Once back at command level, the user may then save a memory image of the TPA that may be examined, modified, or executed at later time.

I have included the following literal flowchart to help summarize what happens when CP/M is brought in on a system after power up.

---continued on page 10---
**MICROFOCUS**

**MICROFOCUS**

- **BASIC-80**
  - Disk Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **FORTRAN**
  - ANSI standard
  - Supports up to 4000 lines
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **MACRO FOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

- **MICROFOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

- **XENON-38**
  - Disk Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **MICROFOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

**NOVA**

- **NOVA-80**
  - Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **NOVA-90**
  - Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **XENON-38**
  - Disk Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied

- **MICROFOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

- **MICROFOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

**SUNDESK**

- **SUNDESK-80**
  - Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **SUNDESK-90**
  - Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **XENON-38**
  - Disk Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied

- **MICROFOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

- **MICROFOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

**STUDY AND DESIGN**

- **STUDY AND DESIGN**
  - Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **STUDY AND DESIGN**
  - Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **XENON-38**
  - Disk Extended BASIC
  - ANSI compatible
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied

- **MICROFOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

- **MICROFOCUS**
  - Supports 32 macro commands
  - Can be used with all microcomputers
  - 150 programs supplied

**TEXTWRITER**

- **TEXTWRITER III**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **TEXTWRITER II**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **TEXTWRITER**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

**TEXTWRITER II**

- **TEXTWRITER II**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **TEXTWRITER**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

**TEXTWRITER**

- **TEXTWRITER**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **TEXTWRITER**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

**TEXTWRITER II**

- **TEXTWRITER II**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **TEXTWRITER**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer

- **TEXTWRITER**
  - Text formatter
  - Supports up to 9999 steps
  - Includes line numbers
  - Text editing, search, replace, backup, and recovery
  - 150 programs supplied
  - License for use on one computer
1. TURN ON POWER

2. PLACE SYSTEM DISKETTE IN DRIVE 0

3. HIT RESET INITIALIZES ROM BOOTSTRAP LOADER

4. HIT RUN SETS DISK DRIVE 0 TO TRACK 00, SECTOR 1

   ROM BOOTSTRAP LOADER LOADS IN Boot
   LOADER FROM SYSTEM DISKETTE TO LOCATION
   00-7FH

   COMPUTER JUMPS OUT OF ROM BOOTSTRAP TO
   THE TO THE BOOT LOADER AT LOCATION 00

   CCP, BDOS, AND BIOS ARE BROUGHT IN AND
   PLACED IN MEMORY AT THEIR RESPECTIVE
   LOCATIONS.

   THE COMPUTER JUMPS TO ROUTINE IN BIOS
   CALLED BOOT WHICH PRINTS A MESSAGE
   ASKS THE OPERATOR FOR THE NUMBER OF
   DISK DRIVES IN THE SYSTEM.

5. TYPE IN AT THE CONSOLE BIOS LOGS THE NUMBER OF DISKS, PLACE
   THE NUMBER OF DISK DRIVES IN SYSTEM
   A JUMP INSTRUCTION TO A BOOT ROUTINE
   IN BIOS AT LOCATION 000, AND THEN
   PLACES A JUMP INSTRUCTION TO FBASE
   (START OF BDOS) AT LOCATION 005H WHICH
   IS FOR SYSTEM CALLS BY PROGRAMS IN
   THE TPA.

6. PROMPT IS PRINTED SYSTEM IS READY FOR COMMAND PROCESSING
   ON CONSOLE TERMINAL

In the above, 36K is the CP/M memory size, V1.4 stands for version 1.4
of CP/M, and 2SIO VERSION is the type of
I/O board used in my system. I typed in
2 to the question "HOW MANY DISKS?"
because I currently have two disk
drives, and finally the prompt "A>"
signifies that drive "A" (drive 0) is
the currently logged on disk. Also,
memory locations 00-02 contain (C3 03
8A) which is the jump to the boot
routine in BIOS as mentioned above, and
locations 05-07 contain (C3 03 7D) which
is the jump to BDOS to service system
calls. All values in hexadecimal.

Hopefully, the above text will give
an idea of the structure and initial
workings of CP/M on power up and reset.
Next month I will discuss the basic
command syntax of CP/M, file structure,
and diskette mapping. In future
articles, I will discuss modifying CP/M
for different hardware configurations,
using system calls, running utility
programs, and implementing I/O STATUS
BYTE options.

NEXT MONTH

PART 2

CP/M's File Structure and Command Syntax

NEW PRODUCTS

CP/M VERSION 2.0 MP/M SOFTWARE RELEASED

DIGITAL RESEARCH, P.O. Box 579,
Pacific Grove, CA 93950 has announced
the release of CP/M Version 2.0. CP/M
2.0 allows for data file management for
up to 128 megabytes. The CP/M BDOS,
completely table drive, allows de-
finiteion of as many as 16 logical drives
of up to 8 megabytes each. It includes
new random access features, separation
of files by user tag number and read-
only protection of individual files.
Other enhancements are contained in the
modules and utilities. New manuals
include a "System Alteration Guide" and
an "Interface Guide" for customizing the
software. CP/M 2.0 is upward compatible
with previous releases of CP/M.

Also introduced is MP/M 1.0, which
is a multiprogramming monitor for 8080,
Z80 and 8085 processors. MP/M is also
upward compatible with CP/M and uses the
CP/M 2.0 file structures.

S-100 MICROSYSYEMS
MODIFYING THE
SDS VDB-8024
by
Jon Bondy
Box 148
Ardmore, Pa. 19003

This article describes how to alter the SDS-8024 Video Board to support all the sophisticated screen editing functions of UCSD PASCAL. Further, it contains a routine for adding PEEK and POKE like functions to UCSD PASCAL and a PASCAL program for burning in an EPROM.

S-100 Bus boards are available with an enormous range of functions, performance and prices. My experience with SD Sales (now SD Systems) computer products has been that they are good quality and reasonably priced. Their ExpandoRam boards are an exception, being of good quality and rock bottom prices.

When SDS introduced their 24 by 80 character VDB-8024 video board I was excited. since the limitations of my memory-mapped 16 by 64 character board were beginning to cause problems for me. I investigated their product, and discovered that it was a "smart" board, with an on-board Z-80, 2708 EPROM firmware to control the board's functions, a 2708 character EPROM, and one of the (at that time) new CRT controller chips, the SMC 5027. The kit was a bit steep at $320, but the fact that I could customize it made it attractive.

The board works as a small computer system in its own right, accepting data from the main computer and processing it in order to achieve the same functions as a smart terminal. Specifically, it must move the character data into the refresh RAM at the proper location, move the cursor around, and perform whatever "smart" functions, such as random cursor motion, are required. The CRT controller chip, the SMC 5027, shares the refresh RAM with the Z-80, as a DMA device, so that about one half of the memory accesses are made by the 5027 and the remainder by the Z-80.

Those of you who have written drivers for the memory mapped video boards will already know that scrolling all of the characters on the screen up one line is a time consuming business. Most of the CRT controller chips provide ways to perform scrolling rapidly, usually by changing the upper left character. The 5027 does this, and although it increases the speed of the VDB-8024 board, it also complicates the firmware which drives it. At any given time, the 5027 may "think" that the top of the screen is at any one of 24 locations in the refresh memory, and in order to function correctly, the firmware must keep track of what the 5027 is thinking. In order to fully understand the firmware listing given at the end of the article, you will probably have to read articles or data sheets on the 5027, since a complete discussion of the 5027's features is beyond the scope of this article. Note that discussions of the 5027 have appeared in BYTE and INTERFACE AGE magazines.

I have used computers for over twelve years, and have become a confirmed high level language freak, since the novelty of machine language wore off about seven years ago. I run the UCSD PASCAL operating system exclusively (except when I am forced to use CP/M to re-write my BIOS), so it was important to me that whatever "terminal" I used could support that software system. UCSD PASCAL includes a sophisticated screen editor, and thus requires some special CRT functions, such as erase-to-end-of-line, erase-to-end-of-screen, random cursor addressing, home and cursor motion controls. The VDB-8024 as delivered included a set of firmware which supported most of these features (and some additional features which I did not need), but did not support the erase-to-end-of functions which I required. I set out to find out how hard it would be to modify the board.

The first step was to get the kit working without any changes, and then to attempt the modifications. The kit went
together easily, and there were no problems getting the board to function as advertised. One flaky RAM chip was annoying, but since SDS would not send me a replacement without my first sending them the chip, I replaced it myself rather than go without the board for the weeks the trade might take. At the time that I purchased the kit (May of 1979), the documentation was not exactly correct for the board, but I assume that that has been improved.

SDS includes a listing of their firmware in their instruction book and so it was easy to start thinking about how to modify their software. My first observation was that, although their hardware was fine, their software practices and documentation were rather poor. For instance, they never gave the ports which controlled the SMC 5027 any logical names, using hex instead, as in the following:

```
OUT (SAH),A
LD A,065H
OUT (SlH)
LD A,OD7H
OUT (SOH)
```

Since I had written my own driver for my memory mapped video board previously, I had incorporated a somewhat unusual manner of performing random cursor addressing. Although there is no standard for this function, many terminals do it by accepting an ESCape character first (lBH), followed by the x- and y- locations for the cursor. One minor problem with this is that it would seem that only two characters should be required for this function, rather than three. The real problem, however, is that it does not allow for random x- and y-addressing independently. My non-standard method did, so I changed their firmware again.

My method uses (wastes?) the ASCII characters from 80H to 8FH by assigning the following values. If the character is between 80H (128) and 0DH (209), it controls the x-address, and the new x-address may be found by subtracting 128 from the character. This allows me to set up x-addresses from zero to 81. If the character is greater than ODH, it controls the y-address, and the new address may be found by subtracting 210 from the character. This allows me to set up y-addresses from zero to 45.

Reading through the SDS firmware was both interesting and confusing, since I had never seen such control oriented code before. The entire "computer system" was dedicated to the single-minded function of processing characters from the computer to the screen. After the power-on reset occurs, the firmware simply sets up the 5027, enables interrupts, and HALTS! It has nothing to do until the first character comes in from the main computer. When the character arrives the processor will be interrupted, so it made sense to write the program that way. Sequences which in a "normal" program would make little sense began to make sense as I realized that sections of unconnected code would be connected when interrupts began to happen.

Because it is interrupt driven, the real action starts at location 38H, where the interrupt service routine (actually the entire program) starts. It inspects the character to see if it is a control character or a displayable character. For control characters, it manipulates the 2-S0 registers and the 5027 registers to accomplish the required function. For displayable characters, it simply displays them and moves the cursor.

Since I do not use CP/M unless I have to, I wanted to write this new firmware under UCSD PASCAL. Each release of UCSD PASCAL after Version 1.4 has included an assembler for the machine on which the software runs; in my case it was a Z-80. It seemed as if I could in fact develop the assembler.
source and object using the UCSD system and its nice screen editor.

I then came to the question of how to program the EPROM from PASCAL. I had intended to program the new EPROM using a Cromenco Bytesaver EPROM programming board, to which I could get temporary access. The Bytesaver actually burns an EPROM by interpreting a "write" access to an EPROM memory location as a request that that EPROM location be programmed. It seemed that if I could write a PASCAL program which repeatedly wrote the required bit pattern into a particular location in memory, then the EPROM could be programmed in PASCAL. In BASIC, this would correspond to using the "peek" and "poke" facilities.

The UCSD Assembler has many features, among them the ability to generate relocatable code and the ability to assemble many modules in one pass, keeping track of them in a kind of directory at the start of each file. I

---continued on next page---

PROGRAM LISTING 1

PEEK & POKE FUNCTIONS

```
.FUNC MEMREAD;1
POP IX
;PARAM IS ADDRESS
POP IY
;RETURN ADDRESS
POP HL
;POPOP TWO WORDS OF ZEROS
LD E,(HL)
;READ ADDRESS
LD D,0
;READ BYTE
PUSH DE
;RETURN VALUE READ ON STACK
JP (IX)
;RETURN TO CALLER

.PROC MEMWRITE;2
POP IX
;PARAM ARE ADDRESS, DATA
POP IY
;RETURN ADDRESS
POP DE
;POPOP TWO WORDS OF ZEROS
POP HL
;WRITE ADDRESS
LD (HL),E
;WRITE BYTE
JP (IX)
;RETURN TO CALLER
```

---continued on next page---

SPECIALIZING IN
QUALITY MICROCOMPUTER HARDWARE

INDUSTRIAL • EDUCATIONAL • SMALL BUSINESS • PERSONAL

BUILDING BLOCKS FOR MICROCOMPUTER SYSTEMS, CONTROL & TEST EQUIPMENT

R2 I/O
2K ROM
2K RAM
3 Serial Ports
1 Parallel Port
WIRED: $295.00

ECT-100-F
RACKMOUNT CARD CAGES
KIT: $200.00
WIRED: $250.00

TT-10
TABLE TOP MAINFRAMES
KIT: $340.00
WIRED: $395.00

16 K RAM
FULLY STATIC MEMORY
KIT: $279.00
WIRED: $310.00

POWER SUPPLIES, CPU’s, MEMORY, OEM VARIATIONS

ELECTRONIC CONTROL TECHNOLOGY (201) 688-8080
had to use absolute assembly, to be certain that the bit patterns which were in the file were in fact those which I wanted to burn into the EPROM, and I had to find a way to ignore the directory block, which is 512 bytes long.

Since I had access to the Cromenco Bytesaver board, all I had to do in PASCAL was to read the firmware object file, throw away the first 512 bytes, and then transfer the remaining 1024 bytes repeatedly into the area of memory where the Bytesaver was assigned. That would have been easy, but UCSD PASCAL had no routines corresponding to BASIC's "peek" and "poke"; I had to write them also.

The assembly routines for the "peek" and "poke" functions are given in Program Listing 1 and correspond to a PASCAL FUNCTION ('memread') and a PASCAL PROCEDURE ('memwrite').

The PASCAL program to burn the EPROM is shown in Program Listing 2 and should be reasonably easy to read, since the variable names are clear. Comments in PASCAL are enclosed in curly brackets ('{" and "}'). All PASCAL keywords are written in upper case, and all user-defined values are in lower case.

PROGRAM LISTING 2
EPROM BURNER PROGRAM

PROGRAM burn;
TYPE
  byte = 0..255;
VAR
  code : PACKED ARRAY[0..1023] OF byte;
  i, j, addr, count : INTEGER;
  infile : FILE;
( DEFINE ASSEMBLY LANGUAGE MODULES AS EXTERNAL )
PROCEDURE memwrite(addr, data : INTEGER); EXTERNAL;
FUNCTION memread(addr : INTEGER) : INTEGER; EXTERNAL;
BEGIN
  RESET(infile,'JON:CR.T.COD.E'); (OPEN CODE FILE )
  WRITE('ENTER LOOP COUNT : '); 
  READ(count);
  ( READ THE SECOND 512 BYTE BLOCK OF THE FILE INTO 'CODE' ARRAY )
  i := BLOCKREAD(infile,code,2,i);
  addr := (15 * 4096) + (8 * 256) (BYTESAVER ADDRESS )
  ( BURN THE PROM 'COUNT ' TIMES )
  FOR i := 1 TO count DO BEGIN
    WRITELN('STARTING PASS ',i);
    FOR j := 0 TO 1023 DO
      memwrite(addr+j,code[j]);
    END;
    ( CHECK THE PROM )
    FOR i := 0 TO 1023 DO
      IF (memread(addr+i) <> code[i]) THEN
        WRITELN('ADDRESS ',i, IS ',memread(addr+i)', BUT SHOULD BE ',code[i]);
    END;
  END.

continued on page 52
The North Star Horizon computer can be found everywhere computers are used: business, engineering, home — even the classroom. Low cost, performance, reliability and software availability are the obvious reasons for Horizon's popularity. But, when a college bookstore orders our BASIC manuals, we know we have done the job from A to Z.

Don't take our word for it. Read what these instructors have to say about the North Star Horizon:

"We bought a Horizon not only for its reliability record, but also because the North Star diskette format is the industry standard for software exchange. The Horizon is the first computer we have bought that came on-line as soon as we plugged it in, and it has been running ever since!"
— Melvin Davidson, Western Washington University, Bellingham, Washington

"After I gave a ½ hour demonstration of the Horizon to our students, the sign-ups for next term's class in BASIC jumped from 18 to 72."
— Harold Nay, Pleasant Hill HS, Pleasant Hill, California

"With our Horizon we brought 130 kids from knowing nothing about computers to the point of writing their own Pascal programs. I also use it to keep track of over 900 student files, including a weekly updated report card and attendance figures."
— Armando Picciotto, Kennedy HS, Richmond, California

"The Horizon is the best computer I could find for my class. It has an almost unlimited amount of software to choose from. And the dual diskette drives mean that we don't have to waste valuable classroom time loading programs, as with computers using cassette drives."
— Gary Montante, Ygnacio Valley HS, Walnut Creek, Calif.

See the Horizon at your local North Star dealer.
A complete and up-to-date listing of CBBS facilities across the country. Also a summary of how to access them.

Over the past few years, the number of computer hobbyists who have modems has increased dramatically. In the past year alone, some computer clubs show an increase in modems of four to six times the number of the previous year. The increased number of modems on systems has also caused a proliferation of Computerized Bulletin Board Systems throughout the Personal Computing community.

The Computerized Bulletin Board System, or CBBS, acts as a clearinghouse for information of interest to Personal Computing users. Typical messages left on the system include messages of items for sale, people looking for help with hardware or software, people looking for others with the same type of equipment or applications, rumors and messages to others. Also, a few clubs have put their newsletters on the CBBS for distribution before the printed material is ready.

Most CBBS's are similar in operation, since most have gotten their software from the same source and thus they have only minor differences between them. Most run CP/M similar to the Chicago CBBS operated by Ward Cristensen and Randy Suess. Most are compatible with terminals running at 300 baud, RS-232C, while some can also handle other baud rates (the other most common being 110 baud). Usually the hard part is to get through to the system. Most systems are readily accessible after 11PM (the rates are cheaper then also). When you get the dial tone, engage your machine and hit a few carriage returns. From there most systems are self-prompting.

As mentioned earlier, the Chicago CBBS was created by Ward and Randy for members of the Chicago Area Computer/Hobbyist Exchange club to use. Access to the system is via 110 or 300 baud ASCII terminal at 312-528-7141. When you get the tone, press return several times for CBBS to detect your terminal speed.

Another club with a CBBS is the New England Computer Society. Their CBBS is located in Cambridge Massachusetts. They are using the standard CBBS CP/M software from Ward and Randy. The CBBS runs at 110 or 300 baud. Just hit a few returns once you are on for CBBS to detect your speed. A summary of the Cambridge CBBS commands are as follows:

-CBBS COMMAND SUMMARY-

Return = CBBS Command Summary
? = Command Summary
B = Print Bulletins
C = Case Switch
D = Half/Full Duplex Switch
E = Enter Message
Edit commands are:
A = Abort
C = Continue Input
E = Edit Line
H = Help
L = List Lines
R = Retype Line
S = Save Message
G = Goodbye (logoff)
H = Helpful Information
K = Kill (delete) message
N = Set (0-9) Null Fill Characters
Q = Quick Summary of Messages
R = Retrieve Messages
S = Summarize Messages
Search Parameters:
field = string
Where fields are:
P = From
T = To
The number of ABBs (Apple Bulletin Board System) and Forum-80 (TRS-80 Bulletin Board System) is growing in number rapidly as is the CP/M CBBS's. Some other CBBS's offer other features as well as being clearinghouses. A few of the Forum-80 systems have dedicated uses. The Family Historian Forum is an "electronic newsletter for the sharing of research problems, suggestions and messages for genealogists across the nation". The Engineer-80 is dedicated to dissemination of engineering related applications for microcomputers. The Memphis Forum-80 handles information on handicapped applications for microcomputers.

One group in the Washington DC area is AMRAD. Their CBBS is accessible via telephone (703-281-2125) or via an amateur radio repeater in McLean, Virginia. Radio access is via the WR4APC 2 meter repeater on 147.81/147.21 MHz. The computer recognizes the call WR4IWG when sent using 45 baud (60 wpm) Baudot RTTY with tones of 2125 Hz (mark) and 2295 Hz (space). When calling the CBBS, the baud rates are 110 or 300 baud. Other computer clubs are also in the planning stages of telephone/repeater set-ups. The South Florida Computer Club is planning a repeater link in the

continued on page 44

### OBSCURE AND DIRTY PROGRAMMING TRICK

by

Bill Fuller

Computer Hobbyist Group of North Texas

Ever gotten ready to call some subroutine and found that the argument you needed to send it in the A-register was in the B-register because of some previous calculation? And to make things worse you do not have another free register to use to make the swap. So you are forced to make the slow route through main memory to do this data shuffle?

<table>
<thead>
<tr>
<th>before operation</th>
<th>after operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>register-A</td>
<td>register-A</td>
</tr>
<tr>
<td>register-B</td>
<td>register-B</td>
</tr>
</tbody>
</table>

When using 4 bits (A4-A0) for registers.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A(\oplus)B</td>
<td>11100101</td>
</tr>
<tr>
<td>A(\oplus)B</td>
<td>11100101</td>
</tr>
<tr>
<td>A(\oplus)B</td>
<td>11100101</td>
</tr>
<tr>
<td>A(\oplus)B</td>
<td>11100101</td>
</tr>
</tbody>
</table>

where \(\oplus\) = the Exclusive-OR operation.
THE IMPORTANCE OF THE IEEE S-100 STANDARD

by Sol Libes

The IEEE S-100 Standard will soon be adopted. What is the significance of this document?

There are currently close to two dozen different manufacturers of S-100 mainframes and 50 manufacturers of over 400 S-100 plug-in boards. This makes the S-100 bus far and away the most popular computer bus system in current use. Further, it is increasing in popularity. Last year five new S-100 mainframes were introduced despite the fact that three S-100 manufacturers went out of business. Their demise however was due to poor management and not a lack of orders.

The fact is that a bus oriented system offers more power and flexibility than integrated systems such as the TRS-80, PET and APPLE. One example of this power is the fact that an S-100 system user can select from seven different 8-bit CPU boards (8080, 8085, 8088, 286, 6502, 6800 and 6809) and five different 16-bit CPU boards (9900, LSI-11, 8086, 28000 and Pascal Microengine).

However, the S-100 bus, created by MITS in 1975, was only very loosely defined by MITS and many S-100 suppliers changed pin assignments and employed different timing, drive and loading circuits. The result was that many S-100 boards claimed by their suppliers to be S-100 compatible would work in some S-100 systems but not in others. The new IEEE S-100 Standard should eliminate this problem. However, a word of caution is necessary here. The S-100 standard has been in circulation for about a year in proposal form. Some S-100 suppliers have already introduced S-100 products which the manufacturers claim to be "IEEE S-100 compatible" and are in fact compatible with an early version of the proposed standard. Some other suppliers have introduced boards compatible with the later version of the proposed standard which is reprinted here. In the case of the latter there is little likelihood of compatibility problems. However, in the former there could be problems. Once the standard is formally adopted (hopefully by May) this problem should disappear.

Secondly, we are moving into a new era of more powerful microcomputer systems. Things MITS did not even envision, when they created the bus. Things like expanding direct memory addressing beyond 64K bytes and multiprocessors on the bus. The new IEEE S-100 standard thus defines these new more powerful applications. It expands memory addressing to 16 megabytes and I/O addressing to 64K ports. It expands the vectored interrupt system to 11 inputs and provides for up to 16 masters on the bus.

This means that the new IEEE S-100 bus is far and away the most powerful microcomputer bus architecture available. The IEEE standard sets the foundation for even greater use and popularity of the bus, so that it will continue to maintain its dominance for many years to come.

I wish to thank Dr. Robert G. Stewart, Chairman of the IEEE-CS Computer Standards Committee, George Morrow of Thinker Toys and Tru Seaborn, editor of IEEE Computer magazine for their assistance in bringing this reprint to S-100 MICROSYSTEMS.

One last caution. The following is still a proposal. At this point it appears to be very close to what will actually be adopted. However, it is possible that there may be some minor changes (e.g. a pin assignment). As soon as the standard is adopted S-100 MICROSYSTEMS will print the revised standard or the changes.
This proposed standard eliminates many of the problems in the S-100 bus and upgrades it for 16-bit microprocessors. It is offered here for public comment before submission to the IEEE Standards Board.

Standard Specification for S-100 Bus Interface Devices

IEEE Task 696.11/D2
Kells A. Elmquist, InterSystems Inc.
Howard Fullmer, Parasitic Engineering Inc.
David B. Gustavson, Stanford Linear Accelerator Center
George Morrow, Thinker Toys

Copyright © 1979 by the Institute of Electrical and Electronics Engineers. Reprinted from Computer Vol. 12 No. 7.

Introductory comments by
Robert G. Stewart, Chairman,
IEEE-CS Computer Standards Committee

The following draft of a proposed standard for the S-100 bus is the culmination of over a year and a half of effort to eliminate many of the bus's problems and to upgrade it to be suitable for 16-bit microprocessors. The address bus has been extended to 24 bits, the data in and data out buses ganged to form a 16-bit wide data bus for 16-bit transactions, and two additional handshaking lines added to permit intermixing of 8- and 16-bit memory cards.

A binary encoded multiple master arbitration bus permits up to 16 masters on the bus. The necessary logic can be implemented in one chip. Additional ground lines, a power fail line, and an error line have been added. Three lines termed NDEF—for not to be defined—have been allotted to allow leeway to implementers for specialized use. Such use must be specified in all literature. Five lines are RFU—reserved for future use. Some lines formerly used for front panel purposes have been deleted, with the intention that such lines can best be handled by a jumper cable from the CPU card to the front panel. A DMA protocol is specified which provides overlap of the control lines at the beginning and end of the transition between permanent and temporary masters. This allows the address, data, and control buses to settle before information is transferred.

As a bit of personal testimony, I implemented the new DMA protocol on my own system, which includes a Digital Systems dual floppy disk interfaced to a MITS Altair 8800, using DMA for disk transfers. The soft error rate, presumably due to glitching on the positive true logic lines, dropped from a situation where a file would be seriously munged in a few hours to the present situation where I can work for days on end without an observable error.

We have observed a new typographic convention in publishing the proposed standard. The use of an overbar to denote electrically low active or negative true logic lines has been replaced by a postfix asterisk to avoid confusion with Boolean negation and permit typing on word processing systems. This is verbalized by the word "star," replacing the prior word "bar." The Boolean negation overbar can be optionally replaced by a prefix minus sign, with parentheses if needed.

The named authors of the standard were evenly divided as to whether the asterisk should be included in logic equations and state diagrams as well as in electrical signal names and timing diagrams. Two authors believe that the asterisk, when thought of as a designator rather than as the negation operator, adds clarity and consistency, and lessens the need to remember or look up the electrically active level when converting from logic to electrical representations. Such use makes logic state diagrams more directly useful for interpreting oscilloscope or logic analyzer waveforms.

The other two authors feel that the inclusion of the asterisk in the name of a logic state or variable is likely to carry with it the implication of logical negation, thus causing the logic statements to be interpreted incorrectly. Furthermore, they assert that many designers think mostly in terms of electrical levels, with high being true, which again causes logic statements to be interpreted incorrectly. They propose to resolve this hazard by removing the electrical information, i.e., the asterisk, from the variable name when it is us-
ed in a logic context as opposed to an electrical or timing context.

A compromise has been reached where the asterisk is not used in the context of logic equations, but is included elsewhere in the document. We solicit feedback from the readers on these two points of view.

The S-100 bus subcommittee has been ably chaired by George Morrow and Howard Fullmer. Both of them provided invaluable technical insights which have been incorporated throughout the draft standard. John Walker of Marinchip Systems suggested the method of using 16-bit memory and interface cards interchangeably with 8-bit cards. David Gustavson and Leo Paffrath of SLAC suggested the bus arbitration scheme which has also been implemented on the Department of Energy's Fastbus. Howard Fullmer suggested the DMA overlap protocol which lowers glitching noise. Kells Elmquist of InterSystems offered a critique of the draft published in May 1978 in Computer and provided many useful suggestions for improvement. He carefully investigated numerous timing and electrical alternatives and resolved many open questions relating to the standard. Kells wrote the final version of the draft for submission to and revision by the subcommittee.

The IEEE Computer Society is publishing this standard in draft form to allow you to comment upon it prior to submission to the IEEE Standards Board for adoption as an IEEE standard. For example, should the data bus be extended to 32 bits, and if so, how? Your comments should be sent to George Morrow by August 15, 1979, with copies to Gordon Force. Mr. Morrow's address is:

George Morrow
Thinker Toys
5221 Central Avenue
Richmond, California 94804

If you would like to participate in other standardization efforts of the Microprocessor Standards Committee, please contact its chairman:

Gordon Force
Logical Solutions
1128 Amur Creek Court
San Jose, California 95051

Finally, preparation of this proposed standard has benefited from the contributions of many individuals and companies. We indeed thank them all.

The proposed standard

1.0 General

1.1 Scope

This standard applies to interface systems for computer system components interconnected via a 100-line parallel backplane commonly known as the S-100 bus.

It applies to microprocessor computer systems, or portions of them, where

1) Data exchanged among the interconnected devices is digital (as distinct from analog).
2) The total number of interconnected devices is small (22 or fewer).
3) The total transmission path length among interconnected devices is electrically short (25" or less). That is, transmission line propagation delays are not important.
4) The maximum data rate of any signal on the bus is low (less than or equal to 6 MHz).

1.2 Object

This standard is intended:

1) To define a rational, general-purpose interface system for designers of new computer system components that will ensure their compatibility with present and future S-100 computer systems.
2) To provide the microprocessor computer system user with compatible device families which will communicate in an unambiguous way without modification, from which a modularly expandable computer system may be constructed.
3) To enable the interconnection of independently manufactured devices into a single system.
4) To specify terminology and definitions related to the system.
5) To define a system with the minimum number of restrictions on the performance characteristics of devices connected to the system.
6) To define a system that, of itself, is of relatively low cost, and allows the interconnection of low cost devices.
7) To define a system that is easy to use.

1.3 Definitions

The following definitions apply for the purpose of this standard. This section contains only general definitions. Detailed definitions are given in other sections as appropriate.

1.3.1 General system terms

Compatibility. The degree to which devices may be interconnected and used without modification, when designed as defined in Sections 2, 3, and 4 of this standard.

Interface. A shared boundary between parts of a computer system, through which information is conveyed.

Interface system. The device independent functional, electrical, and mechanical elements of an interface necessary to effect unambiguous communication among a set of devices. Driver and receiver circuits, signal line descriptions, timing and control conventions, message transfer protocols, and functional logic circuits are typical interface system elements.
System. A set of interconnected elements constituted to achieve a given objective by performing specified functions.

1.3.2 Signals and paths

Assert. To drive a signal line to the true state. The true state is either a high or low state, as specified for each signal.

Bidirectional bus. A bus used by any individual device, or set of devices, for the two-way transmission of messages, that is, both input and output.

Bit-parallel. A set of concurrent data bits present on a like number of signal lines used to carry information. Bit-parallel data bits may be acted upon concurrently as a group or independently as individual data bits.

Bus. A set of signal lines used by an interface system, to which a number of devices are connected, and over which messages are carried.

Byte. A set of bit-parallel signals corresponding to binary digits operated on as a unit. Connotes a group of eight bits where the most significant bit carries the subscript 7 and the least significant bit carries the subscript 0.

Byte-serial. A sequence of bit-parallel data bytes used to carry information over a common bus.

High state. The electrically more positive signal level used to assert a specific message content associated with one of two binary logic states.

Low state. The electrically less positive signal level used to assert a specific message content associated with one of two binary logic states.

Signal. The physical representation which conveys data from one point to another. For the purpose of this standard, this applies to digital electrical signals only.

Signal level. The magnitude of a signal when considered in relation to an arbitrary reference magnitude (voltage in the case of this standard).

Signal line. One of a set of signal conductors in an interface system used to transfer messages among interconnected devices.

Signal parameter. That parameter of an electrical quantity whose values or sequence of values convey information.

Unidirectional bus. A bus used by a device for one-way transmission of messages, that is, either input only or output only.

Word. A set of bit-parallel signals corresponding to binary digits and operated on as a unit. Usually connotes a group of 16 bits where the most significant bit carries the subscript 15 and the least significant bit carries the subscript 0.

1.4 State diagram notation

Each state that an interface function can assume is represented graphically by a circle. A mnemonic is used within the circle to identify the state.

All permissible transitions between states of an interface function are represented graphically by arrows between them. Each transition between states may be qualified by an expression whose value must be either true or false. If a state transition is not qualified by an expression it is assumed that transition from one state to another will occur after a minimum time period, as indicated in the timing specifications. An interface function must enter the state pointed to if and only if the driving expression becomes true, or in the case of a time dependent transition, as soon as the minimum specified time has passed.

An expression consists of two parts, a driving expression and a driven expression, separated by a slash (/). The driving expression is mandatory and specifies the conditions necessary for the state transition. The driven expression is optional and is used to indicate signal transitions as a result of the state transition. A signal transition is indicated by the signal name followed by an equal sign (=), followed by an indication of the state attained by the signal as a result of the transition. A driving expression consists of one or more messages used in conjunction with the operators AND (a·b), OR (a+b), and NOT (¬a). Precedence is defined by parentheses. An example expression is: (driving/driven)

A · (B + C) / D=F(ALSE), E=T(RUE)

If A AND (B OR C) is true, then D is forced false and E is forced true, and the state transition takes place.

1.5 Logical and electrical state relationships

This standard makes a distinction between the logical function of a signal and its electrical implementation. All equations in this standard are logic equations, not electrical equations (unless otherwise stated), and are written in terms of logic states. The use of the term “active” for the purpose of this standard is synonymous with the logic state true.

There are two types of electrical implementation of the logic states:

Active high signals. Active high signals are represented without a suffix after the signal name mnemonic (i.e. ABCD).

<table>
<thead>
<tr>
<th>LOGIC STATE</th>
<th>BINARY STATE</th>
<th>ELECTRICAL SIGNAL LEVEL</th>
<th>ELECTRICAL STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE (F)</td>
<td>0</td>
<td>≤ 0.8 V, CALLED THE LOW STATE</td>
<td>L</td>
</tr>
<tr>
<td>TRUE (T)</td>
<td>1</td>
<td>≥ 2.0 V, CALLED THE HIGH STATE</td>
<td>H</td>
</tr>
</tbody>
</table>
**Active low signals.** Active low signals are represented with an asterisk suffix after the mnemonic (i.e. ABCD*).

<table>
<thead>
<tr>
<th>LOGIC STATE</th>
<th>BINARY STATE</th>
<th>ELECTRICAL SIGNAL LEVEL</th>
<th>ELECTRICAL STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE (F)</td>
<td>0</td>
<td>≥ 2.0 V, CALLED H</td>
<td>H</td>
</tr>
<tr>
<td>TRUE (T)</td>
<td>1</td>
<td>≤ 0.8 V, CALLED L</td>
<td>L</td>
</tr>
</tbody>
</table>

In translating a logic equation into an electrical implementation, care must be taken to account for the active-high or active-low character of the electrical signal. For example, the logic equation

\[ MWRT = pWR \cdot -sOUT, \]  

when implemented electrically, becomes

\[ MWRT = (-pWR*) \cdot -sOUT, \]

since \( pWR^* \) is the electrical signal carrying the \( pWR \) information on the bus.

Note that this is equivalent to

\[ MWRT = -(pWR^* + sOUT), \]

by deMorgan's theorem; consequently, a single two-input NOR gate is sufficient to implement MWRT, if it meets the loading and drive requirements.

The edge or change of electrical value of an electrical signal on a timing diagram which causes a transition change of the variable as a logic variable from false to true is:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>active high</td>
<td>rising</td>
</tr>
<tr>
<td>active low</td>
<td>falling</td>
</tr>
</tbody>
</table>

Logic equations in state diagrams are written in terms of logic state, not electrical state.

The suffix asterisk "*" is not a negation operator. It is a designator (like a comment or footnote) attached to a name, telling what the relationship is between the truth state and the electrical state. That is, this variable is true when the line on the bus is low.

A prefix minus sign "-" represents the logical negation operator and is equivalent to the use of an overbar. Parentheses are used to enclose the negated variable when required for clarity.

1.6 Interface system overview

1.6.1 Interface system objective

The overall purpose of the interface system is to provide an effective communication link over which messages are carried in an unambiguous way among a group of interconnected devices.

Messages in an interface system belong to either of two broad categories:

1) Messages used to manage the interface system itself, called interface messages.
2) Messages used by the devices interconnected by the interface system, and carried by that system, but not part of the interface system itself (i.e. data). These are called device dependent messages.

The interface system herein described comprises the necessary functional and electrical specifications for interface messages to effect the objective of this standard, but it is beyond the scope of this standard to specify the nature or meaning (other than electrical signal level) of device dependent messages.

1.6.2 Fundamental communication capabilities

An effective communication link requires two basic functional elements to organize and manage the flow of information among devices:

1) A device acting as a bus master.
2) A device acting as a bus slave.

All data transfer communications between a bus master and a bus slave are carried out in terms of a generalized bus cycle generated by the bus master and responded to by the addressed bus slave.

In the context of the interface system described by this standard:

1) A device acting as a bus master has the capability to address all bus slaves, or some portion of them, by generating all interface messages necessary to effect a bus cycle, and has the capability to transfer device dependent messages to or from the addressed slave as a part of that bus cycle.
2) A device acting as a bus slave monitors all bus cycles, and has the capability, thus, to be addressed by the bus master and to transfer device dependent messages to or from the bus master.

Bus master and bus slave capabilities occur both individually and collectively in devices interconnected via the S-100 interface system.

1.6.3 Message paths and bus structure

The S-100 interface system consists of a set of signal lines used to carry all information, interface messages and device dependent messages among interconnected devices.

The bus structure is organized into eight sets of signal lines:

1) Data bus— 16 signal lines.
2) Address bus— 16 or 24 signal lines.
3) Status bus— 8 signal lines.
4) Control output bus— 5 signal lines.
5) Control input bus— 6 signal lines.
6) DMA control bus— 8 signal lines.
7) Vectored interrupt bus— 8 signal lines.
8) Utility bus— 20 signal lines.
2.0 Functional specification

2.1 Functional partition

Functional devices interconnected via the interface system are divided into two broad classifications, bus masters and bus slaves, according to their relationship to the generation and reception of interface messages.

Devices acting as bus masters are responsible for the initiation of all bus cycles, and for the generation of all signals necessary for the conduction of an unambiguous bus cycle. These signals are termed type M signals, and consist of the address, status, and control buses. Device dependent messages are transmitted and received on the data bus.

Bus masters are subdivided into two classifications, permanent masters and temporary masters. A permanent bus master (generally a CPU) is the highest priority master in the interface system. A temporary master may request the bus from the permanent master for an arbitrary number of bus cycles, and then returns control of the bus to the permanent master. The transfer of bus control from a permanent master to a temporary master and back to the permanent master is termed a DMA cycle.

The difference between a permanent bus master and a temporary bus master is that:

1) Only one permanent master may exist within the interface system, whereas up to 16 temporary masters may co-exist in a single system.
2) A temporary master is not subject to a DMA cycle, that is, there are no nested DMA operations.

Devices acting as bus slaves are bus cycle receptors. A bus slave monitors all bus cycles and, if addressed during a particular bus cycle, accepts or sends the requested device dependent message on the data lines. While bus masters must generate a specific set of signals in order to assure an unambiguous bus cycle, a bus slave need only examine and generate that subset of bus signals necessary to communicate with bus masters.

2.2 Signal lines

2.2.1 General

The bus is a collection of message paths defined relative to the current bus master. They are:

1) Address bus.
2) Status bus.
3) Data input/output bus.
4) Control output bus.
5) Control input bus.
6) DMA control bus.
7) Vectored interrupt bus.
8) Utility bus.

The nature and use of each bus is specified in the following sections.

2.2.2 Address bus

The address bus consists of 16 or 24 bit-parallel signal lines used to select a specific location in memory or a specific input/output device for communication during the current bus cycle.

All bus masters must assert at least 16 address bits, but may assert 24 address bits if extended address capability is desired. Validity of the address bus is defined in 2.7.3.

Table 1 summarizes address usage for various bus cycles.

<table>
<thead>
<tr>
<th>CYCLE TYPE</th>
<th>STANDARD ADDRESSING</th>
<th>EXTENDED ADDRESSING</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMORY READ</td>
<td>A0-A15</td>
<td>A0-A23</td>
</tr>
<tr>
<td>MEMORY WRITE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M1 (OP-CODE FETCH)</td>
<td>A0-A15</td>
<td>A0-A15</td>
</tr>
<tr>
<td>INPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPT ACKNOWLEDGE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>HALT ACKNOWLEDGE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
</tbody>
</table>

† see 2.2.2.3

2.2.2.1 Standard memory addressing

The standard memory address bus consists of 16 lines specifying 1 of 64K memory locations. These 16 lines are named A0 through A15, where A15 is the most significant bit.

2.2.2.2 Extended memory addressing

The extended memory address bus consists of 24 lines specifying 1 of 16 million memory locations. These 24 lines are named A0 through A23, where A23 is the most significant bit.

2.2.2.3 Standard input/output device addressing

The extended I/O device address bus consists of 8 lines, A0 through A7, specifying 1 of 256 I/O devices. A7 is the most significant bit.

NOTE: The I/O device address has traditionally been duplicated onto the high order address byte, A15-A8. While this is considered acceptable procedure, it is not recommended for new designs as it complicates expansion to extended I/O device addressing.

2.2.2.4 Extended input/output device addressing

The extended I/O device address bus consists of 16 lines, A0 through A15, specifying 1 of 64K devices. A15 is the most significant bit.

2.2.3 Status bus

The status bus consists of eight lines which identify the nature of the bus cycle in progress, and qualify the nature of the address on the address bus.
The mnemonics for status lines always begin with a lower-case s.

The 8 status lines are:

1) Memory read— sMEMR.
2) Op-code fetch— sMI.
3) Input— sINP.
4) Output— sOUT.
5) Write cycle— sWO*.
6) Interrupt acknowledge— sINTA.
7) Halt acknowledge— sHLTA.
8) Sixteen-bit data transfer request— sXTRQ*.

The S lines on the status bus must be generated by the current bus master.

Validity of the status bus is given in 2.7.3.

2.2.3.1 Status memory write

One relevant status signal is not directly available on the bus, but may be created by the combination of two others. Status Memory Write is defined as:

\[ s_{\text{Memory Write}} = (\neg s_{\text{OUT}}) \cdot s_{\text{WO}}, \text{ (logic equation)} \]

that is, status memory write is true when sOUT is false and sWO (write) is true.

2.2.3.2 Status usage chart

Table 2 gives the status word definition for all possible bus cycles. (W) refers to word (16-bit data path) operations; (B) refers to byte (8-bit data path) operations. H=high state. L=low state. X=don't care.

<table>
<thead>
<tr>
<th>CYCLE TYPE</th>
<th>sMEM</th>
<th>sMI</th>
<th>sWO*</th>
<th>sOUT</th>
<th>sINP</th>
<th>sINTA</th>
<th>sHLTA</th>
<th>sXTRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMORY READ</td>
<td>(B)</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>(W)</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>OP-CODE FETCH</td>
<td>(B)</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>(W)</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>MEMORY WRITE</td>
<td>(B)</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>(W)</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>(B)</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>(W)</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>INPUT</td>
<td>(B)</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>(W)</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>INTERRUPT</td>
<td>(B)</td>
<td>L</td>
<td>X</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>(W)</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>ACKNOWLEDGE</td>
<td>(W)</td>
<td>X</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>X</td>
<td>H</td>
</tr>
<tr>
<td>HALT ACKNOWLEDGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WHERE:

H = HIGH STATE
L = LOW STATE
X = DON'T CARE
W = 16-BIT OPERATION
B = 8-BIT OPERATION

2.2.4 Data bus

Data input and data output are always specified relative to the current bus master. Data transmitted by the current bus master to a bus slave is called data output. Data received by the current bus master from a bus slave is called data input.

The data bus consists of 16 lines grouped as two unidirectional 8-bit buses for byte operations and as a single bidirectional bus for 16-bit word operations.

2.2.4.1 Byte operations

Two unidirectional 8-bit buses are used for byte data transfers. Data output appears on the data output bus (DO0-DO7), where DO7 is the most significant bit. Data input appears on the data input bus (DI0-DI7), where DI7 is the most significant bit.

2.2.4.2 Word operations

For 16-bit data transfers the DI and the DO buses are ganged together, creating a single 16-bit bidirectional bus. Two signal lines control the ganging of the data buses, sixteen request (sXTRQ*) and sixteen acknowledge (sXTRN*). When both of these lines are true (in the low state), the data buses are ganged with DO0 corresponding to DATA 0 and DI7 corresponding to DATA 15, the most significant bit.

Complete specification of the 8/16-bit protocol is given in 2.6.

2.2.5 Control output bus

The 5 lines of the control output bus determine the timing and movement of data during any bus cycle. The mnemonics for the control output lines always begin with a lower-case p.

The five lines are:

1) pSYNC, which indicates the start of a new bus cycle.
2) pSTVAL*, which in conjunction with pSYNC indicates that stable address and status may be sampled from the bus in the current cycle.
3) pDBIN, a generalized read strobe that gates data from an addressed slave onto the data bus.
4) pWR*, a generalized write strobe that writes data from the data bus into an addressed slave.
5) pHLDA, the hold acknowledge signal that indicates the highest priority temporary master that the permanent master is relinquishing control of the bus.

The control output signals are subject to the functional and timing disciplines given in 2.7, 3.8, and 3.9.

2.2.6 Control input bus

The six lines of the control input bus allow bus slaves to synchronize the operations of bus masters with conditions internal to the bus slave (e.g., data not ready), and to request operations of the permanent master (e.g., interrupt or hold).

The six control input lines are:

1) RDY
2) XRDY

PRELIMINARY—SUBJECT TO REVISION
2.2.6.1 Ready lines

The ready lines are used by bus slaves to synchronize bus masters to the response speed of the slave. Thus cycles are suspended and wait states inserted until both ready lines are asserted.

The RDY line is the general ready line for bus slaves. It is specified as an open collector line.

The XRDY line is a special ready line commonly used by front panel devices to stop and single step bus masters. As it is not specified as an open collector line, it should not be used by other bus slaves, since a bus conflict may exist.

2.2.6.2 Interrupt lines

The two interrupt lines, INT* and NMI*, are used to request service from the permanent bus master.

The INT* line may be masked off by the bus master, usually via an internal software operation. If the master accepts the interrupt request on the INT* line, it may respond with an interrupt acknowledge bus cycle, accepting vectoring information from the data bus. The INT* line is often implemented as a "group interrupt" line in conjunction with the vectored interrupt bus. In this case, INT* indicates the presence of one or more vectored interrupt requests.

The NMI* line is a non-maskable interrupt request line, that is, it may not be masked off by the bus master. Accepting an interrupt on the NMI* line need not generate an interrupt acknowledge bus cycle.

An interrupt request on the INT* line is asserted as a level, that is, the line is asserted until interrupt service is received. An interrupt request on the NMI* line, on the other hand, is asserted as a negative going edge, since no interrupt acknowledge cycle need be generated.

Both these lines are specified as open collector lines.

2.2.6.3 Hold request

The hold request line, HOLD*, is used by temporary bus masters to request control of the bus from the permanent bus master. The HOLD* line may be masked by the permanent bus master to prevent temporary masters from gaining bus control.

The HOLD* line is specified as an open collector line, and may only be asserted at certain times. See 2.8.3.

2.2.6.4 Sixteen acknowledge

The sixteen acknowledge line, SIXTN*, is a response to the status signal sixteen request (sXTRQ*), and indicates that the requested 16-bit data transfer is possible.

The SIXTN* line is specified as an open collector line. Detailed specification of the use of this line is given in 2.6.

2.2.7 DMA control bus

The eight lines of the DMA control bus are used in conjunction with control bus signals HOLD* and pHLDA. They arbitrate among simultaneous requests for control of the bus by temporary masters and disable the signal drivers of the permanent bus master, thus effecting an orderly transfer of bus control.

All eight lines of the DMA control bus are specified as open collector lines.

The eight DMA control lines are:

1) DMA0*
2) DMA1*
3) DMA2*
4) DMA3*
5) ADSB*
6) DODSB*
7) SDSB*
8) CDSB*

Detailed specification of the use of these lines is given in 2.8.

2.2.7.1 DMA arbitration

The four lines that arbitrate among simultaneous requests for bus control by temporary masters are DMAO* through DMA3*. The encoded priority of requesters is asserted on these lines and, after settling, they contain the priority number of the highest priority requester.

Detailed specification of this process is given in 2.8.3

2.2.7.2 Bus transfer signals

Four signals are available on the bus to disable the line drivers of the permanent bus master. They are:

1) ADSB*, address disable.
2) DODSB*, data out disable.
3) SDSB*, status disable.
4) CDSB*, control output disable.

Use of these lines is tightly specified during the transfer of the bus from a permanent master to a temporary master, as given in 2.8.2, and any transfer involving the control output lines should follow a similar protocol.

The address, data, and status signals from the permanent master may be disabled and replaced using these signals as long as the contents of these buses is valid for the current bus cycle as though no replacement had occurred.

2.2.8 Vectored interrupt bus

The eight lines of the vectored interrupt bus are used in conjunction with the generalized vectored in-
The eight lines of the vectored interrupt bus are VIO* through VI7*, where VIO* is considered the highest priority interrupt.

The vectored interrupt lines should be implemented as levels, that is, they should be held active until service is received.

2.2.9 System utilities

2.2.9.1 System power

Power in S-100 systems is distributed to bus devices as unregulated voltages. A total of nine bus lines are used:

1) +8 volts, 2 lines.
2) +16 volts, 1 line.
3) -16 volts, 1 line.
4) GROUND, 5 lines.

Ground lines are distributed across the edge connector such that low impedance grounds are available on both sides of the edge connector, and on both sides of the circuit cards.

Power lines are subject to the specifications given in 3.2.

2.2.9.2 System clock

The system clock, Φ, is generated by the permanent master. The control timing for all bus cycles, whether they are cycles of the permanent master or cycles of temporary masters in control of the bus, must be derived from this clock.

This signal is never transferred during a bus exchange operation.

2.2.9.3 CLOCK

This clock is specified as a 2-MHz (0.5 percent tolerance) signal with no relationship to any other bus signal. It is to be used by counters, timers, baud-rate generators, etc.

2.2.9.4 System reset functions

System reset functions are divided into three lines:

1) RESET*, resets all bus masters.
2) SLAVE CLR*, resets all bus slaves.
3) POC*, power-on clear is active only on power-on, and asserts SLAVE CLR* and RESET*.

The POC* signal is specified as having a minimum active period of 10 msec.

RESET* and SLAVE CLR* are specified as open collector lines.

2.2.9.5 Memory write strobe

The memory write strobe, MWRT, must be generated somewhere in the system. It is usually generated by front panel type devices, but is optionally generated by permanent masters or mother boards in systems without front panels. Care must be taken that it is generated at only one point in a given system.

Memory write is defined as:

MWRT = pWR • -sOUT (logic equation)

2.2.9.6 Phantom slaves

A line, PHANTOM*, is provided for overlaying bus slaves at a common address location. When this line is activated phantom bus slaves are enabled and normal bus slaves are disabled.

This line is specified as an open collector line.

2.2.9.7 Error

The line ERROR* is a generalized error line that is asserted when an error of some sort (i.e., parity, write to protected memory) is occurring in the current bus cycle.

This line is specified as an open collector line.

2.2.9.8 Manufacturer specified lines

Three lines which can be specified by individual manufacturers are provided on the bus. These lines, termed NDEF (not to be defined), should only be implemented as options, and shall be provided with jumpers so that possible conflicts may be eliminated.

Any manufacturer MUST specify in detail any use of these lines. Signals on these lines are limited to 5 volt logic levels.

2.2.9.9 Power fail (PWRFAIL*)

The power fail line indicates impending power failure, and remains true until power is restored and POC* is true.

2.2.9.10 Reserved lines (RFU)

The five remaining lines are reserved for future use and may not be used for any purpose.

2.2.10 Pin list

Pin connections to the card edge connector shall conform to the list given in Table 3.

2.3 The permanent master interface

2.3.1 General

The permanent master interface provides the capability to transfer device dependent messages to and from all bus slaves. It is responsible for the genera-
<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SIGNAL &amp; TYPE</th>
<th>ACTIVE LEVEL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+8 VOLTS (B)</td>
<td>H</td>
<td>Instantaneous minimum greater than 7 volts, instantaneous maximum less than 25 volts, average maximum less than 11 volts.</td>
</tr>
<tr>
<td>2</td>
<td>+16 VOLTS (B)</td>
<td>H</td>
<td>Instantaneous minimum greater than 14.5 volts, instantaneous maximum less than 35 volts, average maximum less than 21.5 volts.</td>
</tr>
<tr>
<td>3</td>
<td>XRDY (S)</td>
<td>L, O.C.</td>
<td>One of two ready inputs to the current bus master. The bus is ready when both these ready inputs are true. See pin 72.</td>
</tr>
<tr>
<td>4</td>
<td>VIO*(S)</td>
<td>L, O.C.</td>
<td>Vectored interrupt line 0.</td>
</tr>
<tr>
<td>5</td>
<td>VI1*(S)</td>
<td>L, O.C.</td>
<td>Vectored interrupt line 1.</td>
</tr>
<tr>
<td>6</td>
<td>VI2*(S)</td>
<td>L, O.C.</td>
<td>Vectored interrupt line 2.</td>
</tr>
<tr>
<td>7</td>
<td>VI3*(S)</td>
<td>L, O.C.</td>
<td>Vectored interrupt line 3.</td>
</tr>
<tr>
<td>8</td>
<td>VI4*(S)</td>
<td>L, O.C.</td>
<td>Vectored interrupt line 4.</td>
</tr>
<tr>
<td>9</td>
<td>VI5*(S)</td>
<td>L, O.C.</td>
<td>Vectored interrupt line 5.</td>
</tr>
<tr>
<td>10</td>
<td>VI6*(S)</td>
<td>L, O.C.</td>
<td>Vectored interrupt line 6.</td>
</tr>
<tr>
<td>11</td>
<td>VI7*(S)</td>
<td>L, O.C.</td>
<td>Vectored interrupt line 7.</td>
</tr>
<tr>
<td>12</td>
<td>NM1*(S)</td>
<td>L, O.C.</td>
<td>Non-maskable interrupt.</td>
</tr>
<tr>
<td>13</td>
<td>PWRFAIL*(B)</td>
<td>L, O.C.</td>
<td>Power fail bus signal. (See Section 2.10.1 regarding pseudo open-collector nature)</td>
</tr>
<tr>
<td>14</td>
<td>DMA3* (M)</td>
<td>L, O.C.</td>
<td>Temporary master priority bit 3.</td>
</tr>
<tr>
<td>15</td>
<td>A18 (M)</td>
<td>H</td>
<td>Extended address bit 18.</td>
</tr>
<tr>
<td>16</td>
<td>A16 (M)</td>
<td>H</td>
<td>Extended address bit 16.</td>
</tr>
<tr>
<td>17</td>
<td>A17 (M)</td>
<td>H</td>
<td>Extended address bit 17.</td>
</tr>
<tr>
<td>18</td>
<td>SDBS* (M)</td>
<td>L, O.C.</td>
<td>The control signal to disable the 8 status signals.</td>
</tr>
<tr>
<td>19</td>
<td>CODS* (M)</td>
<td>L, O.C.</td>
<td>The control signal to disable the 5 control output signals.</td>
</tr>
<tr>
<td>20</td>
<td>GND (B)</td>
<td>H</td>
<td>Common with pin 100.</td>
</tr>
<tr>
<td>21</td>
<td>NDEF</td>
<td>H</td>
<td>Not to be defined. Manufacturer must specify any use in detail.</td>
</tr>
<tr>
<td>22</td>
<td>ADSB* (M)</td>
<td>L, O.C.</td>
<td>The control signal to disable the 16 address signals.</td>
</tr>
<tr>
<td>23</td>
<td>D0DSB* (M)</td>
<td>L, O.C.</td>
<td>The control signal to disable the 8 data output signals.</td>
</tr>
<tr>
<td>24</td>
<td>Φ (B)</td>
<td>H</td>
<td>The master timing signal for the bus.</td>
</tr>
<tr>
<td>25</td>
<td>pSTVAL*(M)</td>
<td>L</td>
<td>Status valid strobe.</td>
</tr>
<tr>
<td>26</td>
<td>pHLDA (M)</td>
<td>H</td>
<td>A control signal used in conjunction with HOLD* to coordinate bus master transfer operations.</td>
</tr>
<tr>
<td>27</td>
<td>RFU</td>
<td>H</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>28</td>
<td>RFU</td>
<td>H</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>29</td>
<td>A5 (M)</td>
<td>H</td>
<td>Address bit 5.</td>
</tr>
<tr>
<td>30</td>
<td>A4 (M)</td>
<td>H</td>
<td>Address bit 4.</td>
</tr>
<tr>
<td>31</td>
<td>A3 (M)</td>
<td>H</td>
<td>Address bit 3.</td>
</tr>
<tr>
<td>32</td>
<td>A15 (M)</td>
<td>H</td>
<td>Address bit 15 (most significant for non-extended addressing.)</td>
</tr>
<tr>
<td>33</td>
<td>A12 (M)</td>
<td>H</td>
<td>Address bit 12.</td>
</tr>
<tr>
<td>34</td>
<td>A9 (M)</td>
<td>H</td>
<td>Address bit 9.</td>
</tr>
<tr>
<td>35</td>
<td>D01 (M)/DATA1 (M/S)</td>
<td>H</td>
<td>Data out bit 1, bidirectional data bit 1.</td>
</tr>
<tr>
<td>36</td>
<td>D00 (M)/DATA0 (M/S)</td>
<td>H</td>
<td>Data out bit 0, bidirectional data bit 0.</td>
</tr>
<tr>
<td>37</td>
<td>A10 (M)</td>
<td>H</td>
<td>Address bit 10.</td>
</tr>
<tr>
<td>38</td>
<td>D04 (M)/DATA4 (M/S)</td>
<td>H</td>
<td>Data out bit 4, bidirectional data bit 4.</td>
</tr>
<tr>
<td>39</td>
<td>D05 (M)/DATA5 (M/S)</td>
<td>H</td>
<td>Data out bit 5, bidirectional data bit 5.</td>
</tr>
<tr>
<td>40</td>
<td>D06 (M)/DATA6 (M/S)</td>
<td>H</td>
<td>Data out bit 6, bidirectional data bit 6.</td>
</tr>
<tr>
<td>41</td>
<td>D12 (S)/DATA10 (M/S)</td>
<td>H</td>
<td>Data in bit 2, bidirectional data bit 10.</td>
</tr>
<tr>
<td>42</td>
<td>D13 (S)/DATA11 (M/S)</td>
<td>H</td>
<td>Data in bit 3, bidirectional data bit 11.</td>
</tr>
<tr>
<td>43</td>
<td>D17 (S)/DATA15 (M/S)</td>
<td>H</td>
<td>Data in bit 7, bidirectional data bit 15.</td>
</tr>
<tr>
<td>44</td>
<td>sM1 (M)</td>
<td>H</td>
<td>The status signal which indicates that the current cycle is an op-code fetch.</td>
</tr>
<tr>
<td>45</td>
<td>sOUT (M)</td>
<td>H</td>
<td>The status signal identifying the data transfer bus cycle to an output device.</td>
</tr>
<tr>
<td>46</td>
<td>sINP (M)</td>
<td>H</td>
<td>The status signal identifying the data transfer bus cycle from an input device.</td>
</tr>
<tr>
<td>47</td>
<td>sMEMR (M)</td>
<td>H</td>
<td>The status signal identifying bus cycles which transfer data from memory to a bus master, which are not interrupt acknowledge instruction fetch cycle(s).</td>
</tr>
<tr>
<td>48</td>
<td>sHLT1 (M)</td>
<td>H</td>
<td>The status signal which acknowledges that a HLT instruction has been executed.</td>
</tr>
<tr>
<td>49</td>
<td>CLOCK(B)</td>
<td>H</td>
<td>The status signal which acknowledges that a HLT instruction has been executed.</td>
</tr>
<tr>
<td>50</td>
<td>GND (B)</td>
<td>H</td>
<td>Common with pin 100.</td>
</tr>
<tr>
<td>51</td>
<td>+6 VOLTS (B)</td>
<td>H</td>
<td>Common with pin 1.</td>
</tr>
<tr>
<td>52</td>
<td>−16 VOLTS (B)</td>
<td>H</td>
<td>Instantaneous maximum less than −14.5 volts, instantaneous minimum greater than −35 volts, average minimum greater than −21.5 volts.</td>
</tr>
<tr>
<td>53</td>
<td>GND (B)</td>
<td>H</td>
<td>Common with pin 100.</td>
</tr>
<tr>
<td>54</td>
<td>SLAVE CLR* (B)</td>
<td>L, O.C.</td>
<td>A reset signal to reset bus slaves. Must be active with POC* and may also be generated by external means.</td>
</tr>
<tr>
<td>55</td>
<td>DMA0* (M)</td>
<td>L, O.C.</td>
<td>Temporary master priority bit 0.</td>
</tr>
<tr>
<td>PIN NO.</td>
<td>SIGNAL &amp; TYPE</td>
<td>ACTIVE LEVEL</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>56</td>
<td>DMA1* (M)</td>
<td>L</td>
<td>Temporary master priority bit 1.</td>
</tr>
<tr>
<td>57</td>
<td>DMA2* (M)</td>
<td>L</td>
<td>Temporary master priority bit 2.</td>
</tr>
<tr>
<td>58</td>
<td>sXTR0* (M)</td>
<td>L</td>
<td>The status signal which requests 16-bit slaves to assert SIXTN*.</td>
</tr>
<tr>
<td>59</td>
<td>A19 (M)</td>
<td>H</td>
<td>Extended address bit 19.</td>
</tr>
<tr>
<td>60</td>
<td>SIXTN* (S)</td>
<td>L</td>
<td>The signal generated by 16-bit slaves in response to the 16-bit request</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>signal sXTR0*.</td>
</tr>
<tr>
<td>61</td>
<td>A20 (M)</td>
<td>H</td>
<td>Extended address bit 20.</td>
</tr>
<tr>
<td>62</td>
<td>A21 (M)</td>
<td>H</td>
<td>Extended address bit 21.</td>
</tr>
<tr>
<td>63</td>
<td>A22 (M)</td>
<td>H</td>
<td>Extended address bit 22.</td>
</tr>
<tr>
<td>64</td>
<td>A23 (M)</td>
<td>H</td>
<td>Extended address bit 23.</td>
</tr>
<tr>
<td>65</td>
<td>NDEF</td>
<td></td>
<td>Not to be defined signal.</td>
</tr>
<tr>
<td>66</td>
<td>NDEF</td>
<td></td>
<td>Not to be defined signal.</td>
</tr>
<tr>
<td>67</td>
<td>PHANTOM* (M/S)</td>
<td>L</td>
<td>A bus signal which disables normal slave devices and enables phantom</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>slaves—primarily used for bootstrapping systems without hardware</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>front panels.</td>
</tr>
<tr>
<td>68</td>
<td>MWRT (B)</td>
<td>H</td>
<td>pWR—pSOUT (logic equation). This signal must follow pWR* by not more</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>than 30 ns. (See note, Section 2.7.5.3)</td>
</tr>
<tr>
<td>69</td>
<td>RFU</td>
<td></td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>70</td>
<td>GND (B)</td>
<td></td>
<td>Common with pin 100.</td>
</tr>
<tr>
<td>71</td>
<td>RFU</td>
<td></td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>72</td>
<td>RDY (S)</td>
<td>H</td>
<td>0.C.</td>
</tr>
<tr>
<td>73</td>
<td>INT* (S)</td>
<td>L</td>
<td>The primary interrupt request bus signal.</td>
</tr>
<tr>
<td>74</td>
<td>HOLD* (M)</td>
<td>L</td>
<td>The control signal used in conjunction with pHLDA to coordinate bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>master transfer operations.</td>
</tr>
<tr>
<td>75</td>
<td>RESET* (B)</td>
<td>L</td>
<td>The reset signal to reset bus master devices. This signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>must be active with POC* and may also be generated by external means.</td>
</tr>
<tr>
<td>76</td>
<td>pSYNC (M)</td>
<td>H</td>
<td>The control signal identifying BS1.</td>
</tr>
<tr>
<td>77</td>
<td>pWR* (M)</td>
<td>L</td>
<td>The control signal signifying the presence of valid data on DO bus or data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bus.</td>
</tr>
<tr>
<td>78</td>
<td>pDBIN (M)</td>
<td>H</td>
<td>The control signal that requests data on the DI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bus or data bus from the currently addressed slave.</td>
</tr>
<tr>
<td>79</td>
<td>A0 (M)</td>
<td>H</td>
<td>Address bit 0 (least significant).</td>
</tr>
<tr>
<td>80</td>
<td>A1 (M)</td>
<td>H</td>
<td>Address bit 1.</td>
</tr>
<tr>
<td>81</td>
<td>A2 (M)</td>
<td>H</td>
<td>Address bit 2.</td>
</tr>
<tr>
<td>82</td>
<td>A6 (M)</td>
<td>H</td>
<td>Address bit 6.</td>
</tr>
<tr>
<td>83</td>
<td>A7 (M)</td>
<td>H</td>
<td>Address bit 7.</td>
</tr>
<tr>
<td>84</td>
<td>A8 (M)</td>
<td>H</td>
<td>Address bit 8.</td>
</tr>
<tr>
<td>85</td>
<td>A13 (M)</td>
<td>H</td>
<td>Address bit 13.</td>
</tr>
<tr>
<td>86</td>
<td>A14 (M)</td>
<td>H</td>
<td>Address bit 14.</td>
</tr>
<tr>
<td>87</td>
<td>A11 (M)</td>
<td>H</td>
<td>Address bit 11.</td>
</tr>
<tr>
<td>88</td>
<td>D02 (M)/DATA2 (M/S)</td>
<td>H</td>
<td>Data out bit 2, bidirectional data bit 2.</td>
</tr>
<tr>
<td>89</td>
<td>D03 (M)/DATA3 (M/S)</td>
<td>H</td>
<td>Data out bit 3, bidirectional data bit 3.</td>
</tr>
<tr>
<td>90</td>
<td>D07 (M)/DATA7 (M/S)</td>
<td>H</td>
<td>Data out bit 7, bidirectional data bit 7.</td>
</tr>
<tr>
<td>91</td>
<td>D14 (S)/DATA12 (M/S)</td>
<td>H</td>
<td>Data in bit 4 and bidirectional data bit 12.</td>
</tr>
<tr>
<td>92</td>
<td>D15 (S)/DATA13 (M/S)</td>
<td>H</td>
<td>Data in bit 5 and bidirectional data bit 13.</td>
</tr>
<tr>
<td>93</td>
<td>D16 (S)/DATA14 (M/S)</td>
<td>H</td>
<td>Data in bit 6 and bidirectional data bit 14.</td>
</tr>
<tr>
<td>94</td>
<td>D11 (S)/DATA9 (M/S)</td>
<td>H</td>
<td>Data in bit 1 and bidirectional data bit 9.</td>
</tr>
<tr>
<td>95</td>
<td>D10 (S)/DATA8 (M/S)</td>
<td>H</td>
<td>Data in bit 0 (least significant for 8-bit data) and bidirectional data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bit 8.</td>
</tr>
<tr>
<td>96</td>
<td>sINTA (M)</td>
<td>H</td>
<td>The status signal identifying the bus input cycle(s) that may</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>follow an accepted interrupt request presented on INT*.</td>
</tr>
<tr>
<td>97</td>
<td>swO* (M)</td>
<td>L</td>
<td>The status signal identifying a bus cycle which transfers data from a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bus master to a slave.</td>
</tr>
<tr>
<td>98</td>
<td>ERROR* (S)</td>
<td>L</td>
<td>0.C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The bus status signal signifying an error condition during present bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cycle.</td>
</tr>
<tr>
<td>99</td>
<td>POC* (B)</td>
<td>L</td>
<td>The power-on clear signal for all bus devices; when this signal goes low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>it must stay low for at least 10 msecs.</td>
</tr>
<tr>
<td>100</td>
<td>GND (B)</td>
<td></td>
<td>System ground.</td>
</tr>
</tbody>
</table>

The permanent master normally has control of the bus. It may relinquish bus control to a temporary bus master via a hold operation for an arbitrary number of cycles. Upon completion of the hold operation control of the bus is always returned to the permanent master.

### 2.3.2 Permanent master state diagram

The permanent master interface shall be implemented so as to conform to the state diagram given in Figure 1.
2.3.3 Permanent master state descriptions

2.3.3.1 Bus state 1

The initial bus state, BS₁, is the state in which the status and address buses are in transition to their values for the new bus cycle. pSYNC goes true in the middle of the BS₁ state, indicating the beginning of a new bus cycle.

2.3.3.2 Bus state 2

Bus state 2, BS₂, is the state during which the address and status lines become stable. When they are guaranteed stable the pSTVAL*, status valid strobe, is activated.

The ready lines and the sixteen acknowledge line are sampled during the BS₂ state.

2.3.3.3 Wait state

The wait state, BSₜ, is entered if the ready line sampled in BSₚ indicates that the addressed bus slave is not ready for data transfer. The ready line is sampled once every clock cycle until a ready condition is indicated. When the ready condition is indicated the BSₚ state is completed and the BS₃ state entered.

The BSₜ state is thus used to synchronize bus cycles generated by bus masters with the response speed of assorted bus slaves.

2.3.3.4 Bus state 3

Bus state 3, BS₃, is the bus state during which the data transfer actually takes place between the master and the addressed slave.

2.3.3.5 Idle bus states

After completion of the BS₃ state, the master may enter one or more idle bus states.

While in an idle bus state the generalized data strobes, pWR* and pDBIN, must not be active, and pSTVAL* must not be asserted in conjunction with pSYNC active.

2.3.3.6 Hold accept

Permanent masters must be configured to conditionally accept hold operations from temporary masters. This function may be gated off under hardware or software control, to allow indivisible test and set operations. If hold is enabled and active, the permanent master will enter the hold state HS following a BS₃ state, and pHLDA will be asserted.

The permanent master remains in the hold state until the hold request HOLD* becomes false.

Hold operations always take priority over interrupt operations.

2.3.3.7 Interrupt accept

If hold request is not active, if execution of the current instruction is complete, and if interrupts are enabled and an interrupt is being requested, then the permanent master accepts the interrupt request at the end of the BS₃ state. In the case of a vectored interrupt, the next bus cycle may be an interrupt acknowledge bus cycle. In the case of a non-maskable interrupt, the response is usually a transfer to a predetermined location.

2.3.4 Required signals for permanent masters

2.3.4.1 Output signals

The following signals are output signals from permanent masters to bus slaves:

1) AO-A23†.
2) All status signals.
3) All control output signals.
4) Data output signals (8 or 16 depending on processor type).
5) the system clock.

2.3.4.2 Input signals

The following signals are required input signals to permanent masters:

1) The control input signals, except NMI* and SIXTN*.
2) Data input signals (8 or 16 depending on processor type).
3) The four disable signals ADSB*, DODSB*, SDSB*, CDSB*.
4) RESET*.

† A16 through A23 are optional on permanent masters.
2.3.5 Dummy mastering

In cases where a number of processors co-exist in a single system as temporary masters, it may prove inefficient from a systems point of view to implement a permanent master.

In such a case it is permissible that the permanent master be implemented as a dummy, that is, as a device that conducts no bus cycles, but only supplies an arbitration interval so that the DMA control bus may settle.

The dummy master takes control of the bus between temporary masters, asserting the control output bus in the null state, and passes the bus to the next requester after an arbitration interval of one clock cycle.

Required output signals for dummy masters are the control output signals, and the system clock φ. Input signals are HOLD* and CDSB*.

2.4 The temporary master interface

2.4.1 General

The temporary master interface provides the capability to transfer device dependent messages to and from a selected set of bus slaves. The temporary master thus differs from the permanent master in that it need not generate all possible bus cycles.

The temporary master requests control of the bus from the permanent master. If the bus is granted, the temporary master is responsible for the generation and timing of all bus cycles until it returns control to the permanent master.

Since up to 16 temporary masters may co-exist in a single system, a protocol has been developed to arbitrate among simultaneous bus requests. Detailed specification of this protocol is given in 2.8.3.

2.4.2 Temporary master state diagram

The temporary master interface shall be implemented so as to conform to the state diagram given in Figure 2.

2.4.3 Temporary master state descriptions

2.4.3.1 Arbitration (ARB)

If more than one temporary master is present in the system, bus requesters must arbitrate for the bus as given in 2.8.3.

During the arbitration sequence, bus requesters try to assert their priorities on the arbitration bus, and the contents of the arbitration bus are compared with each requester's priority.

If the contents of the arbitration bus is of higher priority than the locally attempted priority assertion, then a higher priority requester is present in the system, and the low priority requester removes its low order bits from the arbitration bus. Thus, after some settling time, the priority of the highest priority requester is present on the arbitration bus. This requester is granted the bus on the rising edge of hold acknowledge.

2.4.3.2 Bus transfer states (XS I and XS II)

Since the bus has positive polarity control signals, extreme care must be taken in bus transfer operations to avoid erroneous pulses on the control lines.

In general terms, this is accomplished by specifying that both the permanent master and the temporary master drive the control lines in specified logic states during the bus transfer.

Detailed specification of this operation is given in 2.8.2.
2.4.3.3 Bus cycle

The definition of bus cycle states is the same as that for the permanent master interface, given in 2.3.3.1 through 2.3.3.5.

An arbitrary number of bus cycles may be performed by the temporary master before returning control to the permanent master.

2.4.4 Required signals for temporary masters

2.4.4.1 Output signals

The following are required output signals for a temporary master interface:

1) Address lines A0-A23†.
2) All status signals.
3) All control output signals ‡‡.
4) Data output lines.
5) DMA arbitration lines DMA0*-DMA3*.
6) Hold request, HOLD*.

2.4.4.2 Input signals

The following are required input signals for a temporary master interface:

† Note: Temporary masters must generate A16-A23; they need only generate false or lows on these 8 lines, however.
‡‡ Note: Temporary masters should provide a jumper on the pSTVAL* line, as 8080 CPUs of old design do not transfer this line with the control output lines. In this case all bus masters use the same pSTVAL* signal.

---

1) The ready lines, RDY and XRDY.
2) Hold acknowledge, pHLDA.
3) Data input lines.
4) The system clock, φ.

2.5 The slave interface

A slave device responds to a bus cycle initiated by a bus master. Memory and input/output devices are examples of bus slaves.

A slave device may request service by a bus master by generating an interrupt request.

2.5.1 Slave interface state diagram

The slave interface shall conform, in general, to the state diagram given in Figure 3. Slave interfaces need not have both read and write capability.

2.5.2 Slave state definitions

2.5.2.1 Slave idle state

The slave idle state, S0, is a passive state with respect to the bus.

The slave monitors the stream of bus cycles to determine if it is selected for the current bus cycle.

The slave may be performing internal operations while in the idle state.

The assertion of SLAVE CLR* forces all slaves into the idle state.

2.5.2.2 Slave setup

A slave moves from the slave idle state to the setup state, SSE, when it has been addressed by the current bus cycle. This is an operation internal to the slave which sets up a data transfer with a bus master. If a

---

Figure 2. Temporary master state diagram.

Figure 3. Slave interface state diagram.
slave can tolerate spurious transitions from the idle state to the setup state, then the device select signal may be decoded statically from the address and status buses. If a device cannot tolerate spurious transitions, the device select line should be decoded in conjunction with the status valid strobe, pSTVAL*.

If synchronization is required by the slave before the data transfer may take place, the ready line is asserted false during this state until the device is ready for data transfer.

2.5.2.3 Slave read

Data from the addressed slave is gated onto the data bus during the slave read state, Sr. The generalized read strobe governs the transition to this state. When device select becomes false the slave returns to the idle state.

2.5.2.4 Slave write

Data from the current bus master is written into the slave during the active period of the generalized write strobe, pWR*.

When device select becomes false the slave returns to the idle state.

2.5.2.5 Interrupt request state

If a slave requires service by a bus master, an interrupt request may be generated by the slave. The interrupt should be held active until the slave is serviced, or until SLAVE CLR* is asserted.

2.5.3 Required signals for slave interfaces

Slave interfaces need only receive and generate that subset of bus signals necessary for communication with masters.

2.6 8/16-bit data transfer protocol

2.6.1 General

Implementation of the 8/16-bit data transfer protocol allows both 8-bit and 16-bit parallel data transfers over the bus, and hence allows both 8-bit masters and 16-bit masters and slaves to co-exist in a single system. For 16-bit transfers the two unidirectional 8-bit data buses are ganged to form a single 16-bit bidirectional data bus.

Two lines are assigned to control the ganging of the data bus:

1) sXTRQ*, status output from the master, which indicates a request for a 16-bit data transfer.
2) SIXTN*, an acknowledge input to the master, which indicates that a 16-bit data transfer is possible.

Use of the sixteen acknowledge line SIXTN* permits the use of current design 8-bit memory boards without modification. When SIXTN* is false, a 16-bit transfer may be accomplished by two sequential single-byte transfers.

2.6.2 8-bit data paths

The current bus master requests an 8-bit transfer by not asserting sXTRQ*.

Byte data output from the master to the addressed slave is asserted on the data output bus, DO0 through DO7.

Byte data input from the addressed slave to the current bus master is asserted on the data input bus, DI0 through DI7.

2.6.3 16-bit data paths

The current bus master requests a 16-bit transfer by asserting sXTRQ*.

If the addressed slave is capable of a 16-bit parallel data transfer, it asserts SIXTN*, as shown in the timing diagram (see page 52). Sixteen-bit data transfer is then conducted via the ganged data buses, where DO0 = DATA0, and DI7 = DATA15.

2.6.4 Memory organization

Memory devices capable of both 8-bit and 16-bit parallel data transfers are organized, as shown in Figure 4, as two banks of 8-bit memory, a high-byte bank and a low-byte bank. These data banks may be activated either together or separately, depending on the condition of the sixteen request status line, sXTRQ*.

2.6.4.1 Byte references

When sXTRQ* is not asserted, memory references are single-byte transfers.
The proper location in memory is selected by the address output on address lines A1 through A15 (A23 for extended addressing systems), while the A0 line selects the high byte or the low byte. A0 equals 0 selects the high byte of the 16-bit word, while A0 equals 1 selects the low byte of the word.

See Figure 5 for address usage.

In the 8-bit mode, data output from the master, on the DO bus, is connected to the data input lines of both memory banks; the low-byte data input lines are connected directly to the DO bus, and the high-byte data input lines are connected to the DO bus via a two-to-one multiplexer controlled by sXTRQ*.

Data output from the memory banks is routed to Tri-State† bus drivers A and B in Figure 4. One of these drivers is enabled when the read strobe is activated, depending on the condition of A0. The selected byte is thus available to the master on the DI bus.

2.6.4.2 Word references

When sXTRQ* is asserted by the master, and SIXTN* is asserted by the slave, memory references are double-byte transfers.

Address lines A1 through A15 (A23 in extended address systems) select the proper word from memory. The condition of the A0 bit does not enter into the decoding or addressing for word references.

See Figure 5 for address usage.

In the 16-bit mode, data output from the bus master is asserted on the 16 signal lines of the DO bus and the DI bus. The multiplexer on the data input lines now routes the high-byte data, on the DI bus, to the data input lines of the high-byte bank. Low-byte data, on the DO bus, is connected to the data input lines of the low-byte bank.

Data output from the memory banks is routed through buffers A and C to their respective data paths. Both A and C will be enabled by the read strobe.

2.6.5 Sixteen acknowledge (SIXTN*)

Implementation of the sixteen acknowledge line allows the use of 8-bit memory boards in a 16-bit system without modification, but with a reduction in maximum system bandwidth.

If a 16-bit master requests a 16-bit transfer, but the addressed slave is not capable of such a transfer, the sixteen acknowledge lines will not be asserted.

The master will respond in one of two ways, by generating an error trap or by conducting the transfer in byte-serial fashion.

2.6.5.1 Byte-serial response

If the sixteen acknowledge line is not activated after a specified period, circuitry may be included on bus masters to conduct the requested 16-bit transfer as two consecutive byte operations, thus assembling the requested 16-bit word while holding the master in a wait state.

For this process to occur, the sixteen acknowledge line must meet the timing specifications for the ready line inputs.

2.6.5.2 Error response

If circuitry does not exist on the master to conduct the requested 16-bit transfer as two consecutive byte operations, an error condition shall result immediately, with ERROR* asserted.

2.7 Fundamental bus cycle timing

2.7.1 General

This section deals with the fundamental timing concepts involved in the standard bus cycle. Detailed specification of the timing parameters discussed in this section is given in 3.8 and 3.9.

The standard bus cycle is a pseudo-synchronous cycle, that is, the timing of the control signals bears a specified relationship to the master system clock $\phi$.

All data transfers, including read or write cycles, 8- or 16-bit transfers, memory or input/output device transfers, and interrupt acknowledge are conducted on the bus as a standard bus cycle.†

Figure 6 shows the fundamental timing for a standard bus cycle, with a single wait state inserted by the addressed slave.

2.7.2 Address and status buses

The beginning of a new bus cycle is indicated by the rising edge of the pSYNC signal, which closely follows the rising edge of the system clock, $\Phi$.

The address and status buses are changing to their values for the new cycle during the beginning of the

†See possible exception, section 2.7.5.3.
pSYNC interval. Shortly after they can be guaranteed stable on the bus, the status valid strobe, pSTVAL*, is asserted. pSTVAL*, decoded in conjunction with pSYNC, indicates to all bus slaves that stable address and status may be sampled from the bus.

The position of the status valid strobe within the pSYNC interval is independent of the system clock, $\Phi$. This affords the designer of bus masters considerable flexibility in interfacing different processors to the bus. The status valid strobe should be positioned within the pSYNC interval such that the delay between guaranteed status on the bus and the activation of the status valid strobe is as close to the minimum specification as possible, thus maximizing memory and device access time.

In order to prevent false cycle starts in bus slaves, only one negative edge of the status valid strobe may occur while pSYNC is asserted.

Address and status information is thus stable on the bus from the negative transition of the status valid strobe during pSYNC, and is held stable until a specified period after the trailing edge of the data strobe (pDBIN in the read case, and pWR* in the write case). This hold time ensures that false decoding of the address and status information will not occur at the end of the bus cycle.

2.7.3 Ready and sixteen acknowledge lines

The sixteen acknowledge line, since it may be used to place the bus master in a wait state while a requested 16-bit transfer is conducted in byte-serial fashion, is subject to the same timing constraints as the ready lines.

The ready lines are first sampled by the bus master on the rising edge of the system clock during the BS 2 state, and if active, the master enters a wait state, sampling the ready line once every clock cycle on the rising edge of the system clock until the slave is ready for data transfer.

A minimum setup time before the rising edge of the system clock, and a minimum hold time after sampling must be met for the proper operation of the ready lines.

The time between the active edge of the status valid strobe and the sampling of the ready line may be very short. Hence, it is recommended practice not to make assertion of the ready line dependent on pSTVAL*.

Data output, address, and status are held stable during wait states.

2.7.4 Read cycles

2.7.4.1 General

There are four types of read cycles: op-code fetch (M1), memory read, input, and interrupt acknowledge. These cycles are all similar with respect to timing, but make different use of the status bits and the address bus. See Tables 1 and 2.

2.7.4.2 The read strobe

The generalized read strobe pDBIN is used to gate data from an addressed slave onto the data bus during a read operation. The read strobe is asserted true by the bus master after a minimum specified time from the assertion of the status valid strobe.

It is held true during any inserted wait states, and returns to the false state, returning the data bus to the high impedance state, shortly before the address and status buses are allowed to change.

2.7.5 Write cycles

2.7.5.1 General

There are two possible types of write cycles on the bus, a memory write cycle and an output cycle.

These two cycles are similar with respect to timing, but make different use of the status bits and address bus. A special write strobe, MWRT, is generated for memory cycles.

2.7.5.2 The write strobe

The generalized write strobe, pWR*, is used to write data from the data bus into the addressed bus slave. The write strobe may be asserted by the master after the completion of the pSYNC interval.

Data output on the data bus must be guaranteed valid for a specified period both before and after the activation of the write strobe. Hence, either the leading or the trailing edge of the write strobe may be used to strobe data into the addressed slave.

Address and status information must be held valid for a specified period of time from the trailing edge of the write strobe.

Figure 6. Bus cycle fundamental timing relationships.
2.7.5.3 Memory write strobe

While the generalized write strobe is activated for all write cycles, the memory write strobe is activated for memory write cycles only. The memory write strobe is usually generated by front-panel devices, if they exist in the system, as a function of bus memory write or a front-panel deposit. If front-panel devices do not exist the memory write strobe must be generated somewhere in the system, but at only one point. This circuit should be designed such that it generates the memory write strobe for all bus masters. Jumpers shall be provided to allow extra circuits to be disabled.

The memory write strobe, MWRT, is defined as:

\[ MWRT = pWR \cdot \neg sOUT, \text{ (logic equation)} \]

that is, memory write is true when \( pWR \) is true and \( sOUT \) is false.

The memory write strobe must follow the \( pWR^* \) strobe by not more than a specified period.†

2.8 Special bus operations

2.8.1 General

This section describes two special bus operations related to DMA operations, that is, the transfer of bus control from the permanent bus master to a temporary bus master for an arbitrary number of bus cycles, and the return of control to the permanent bus master.

These two operations are:

1) The bus transfer protocol.
2) The arbitration protocol among simultaneous bus requesters.

2.8.2 Bus transfer protocol

2.8.2.1 General

When a temporary bus master has been granted the bus by the permanent bus master, control must be transferred to the temporary master in such a way that spurious signals are not generated on the control output lines, causing false bus cycles. Since some of the control output signals are of positive polarity, extreme care must be taken in this operation. In general, the specified bus transfer protocol accomplishes this by having the permanent master and the temporary master drive the control output lines simultaneously at specified levels during the bus transfer.

2.8.2.2 Bus transfer state diagram

The bus transfer operation shall be implemented so as to conform to the bus transfer state diagram given in Figure 7.

2.8.2.3 Bus transfer state definitions

2.8.2.3.1 Idle

The idle state signifies that the temporary master is either involved in internal operations, and does not require the bus, or that it is waiting for the bus to become free so that it may assert its bus request.

2.8.2.3.2 Arbitration

If a temporary master desires the bus, and HOLD is false and pHLDA is false, the temporary master enters the arbitration sequence, where it contests with other bus requesters for control of the bus. Detailed specification of this process is given in 2.8.3.

2.8.2.3.3 Bus grant

Priority assertions on the arbitration bus settle in the interval between the assertion of a hold request and a hold acknowledge. At the rising edge of the hold acknowledge signal the bus is granted to the highest priority requester, enabling the bus transfer operation for that requester.

If the bus is not granted to a requester, that requester returns to the idle state.

The bus grant state is termed MINE.

†Note: Historically the MWRT strobe has been generated by front-panel devices to accomplish the deposit function. In such a case, the MWRT strobe is asserted while the front-panel holds the CPU in a wait state during a memory read cycle. Note that the status will indicate a read state and the pDBIN strobe will be active. While this is acceptable procedure for 8-bit systems, and 8-bit memories should respond to MWRT as well as pWR*, it is not permissible in 16-bit systems as a conflict will exist on the bidirectional data bus. A front-panel could be implemented either as a temporary master, or integrated into the CPU.

Figure 7. Bus transfer state diagram.
2.8.2.3.4 Transfer state one, XS I

The bus transfer sequence begins with transfer state one, XS I. The bus transfer control circuit asserts the following signals together:

1) ADSB*
2) SDSB*
3) DODSB*

Disabling the address, status, and data output drivers of the permanent bus master and enabling the control output drivers of the temporary master. Both the permanent master and the temporary master are now driving the control output lines. These lines are required to have the following levels during this time.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Logic state</th>
<th>Electrical level</th>
</tr>
</thead>
<tbody>
<tr>
<td>pSYNC</td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>pSTVAL*</td>
<td>F</td>
<td>H†</td>
</tr>
<tr>
<td>pDBIN</td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>pWR*</td>
<td>F</td>
<td>H</td>
</tr>
<tr>
<td>pHLD A</td>
<td>T</td>
<td>H</td>
</tr>
</tbody>
</table>

The transfer state is terminated by the assertion (by the bus transfer control circuit) of the CDSB* line, disabling the control drivers of the permanent master and enabling the address, status, and data output drivers of the temporary master. The temporary master now has control of the bus and begins its first bus cycle.

2.8.2.3.5 Bus cycles

Any number of standard bus cycles are then conducted by the temporary bus master. Bus control is never transferred between cycles. When the temporary master is done, the process proceeds to XS II, transfer state two.

2.8.2.3.6 Transfer state two

Transfer state two, XS II, is the mirror image of the sequence in XS I. The state begins with the release of the CDSB* signal, enabling the control output drivers of the permanent master and disabling the address, status, and data output drivers of the temporary master.

Both the temporary master and the permanent master drive the control output lines for the remainder of XS II at the levels prescribed for XS I.

The state is ended by the release of other disable signals and HOLD*, enabling the address, status, and data output drivers on the permanent master, and disabling the control output drivers of the temporary master. The permanent master now has complete control of the bus and the temporary master returns to the idle state.

2.8.2.4 Bus transfer timing relationships

2.8.2.4.1 General

The fundamental timing relationships for a bus transfer and a single DMA bus cycle are given in Figure 8.

Relationship to the bus transfer states is shown in boxes at the bottom of the figure.

Detailed specification of these times is given in 3.10 and Table 5.

2.8.2.4.2 Tset

A minimum time between the rising edge of the hold acknowledge signal and the assertion of the disable signals in XS I allows time for completion of the preceding bus cycle.

2.8.2.4.3 Tnu

The time that both the temporary master and the permanent master must drive the control output signals has a specified minimum to assure a smooth bus transfer.

Assertion of the XFER II signal, or CDSB*, is specified relative to the rising edge of the system clock, Φ, so that the assertion of this signal may be used by the temporary master as a cycle start signal.

2.8.2.4.4 Tdh

The “done” signal is a signal internal to the temporary master. This signal should not be asserted until the hold time for data output, status, and address signals in the standard bus cycle has been met.

2.8.2.4.5 Trel

Completion of the reverse bus transfer XFER II shall precede the release of the pHLD A signal by a minimum specified time.

*See note in section 2.4.4.1.
2.8.3 Bus arbitration protocol

In a system which allows more than one master to use the system bus, for example a CPU permanent master and several temporary masters such as DMA controllers or multiple CPUs, some means must be provided to determine which device will be allowed to control the bus at any given time.

The bus arbitration system uses four bus lines for arbitrating among 16 temporary masters. These lines are driven by open collector drivers, and are pulled high by pullup resistors. Each temporary master has a unique priority number which it asserts on the arbitration bus at an appropriate time. A higher binary number indicates a higher priority.

The temporary masters compare the priority appearing on the active-low open-collector bus with the priority they are asserting, starting with the most significant bit. If disagreement is detected by any temporary master at any given bit position, then another temporary master must be asserting that priority bit and thus must have a higher priority. In that case all less significant bits are removed by the detecting temporary master. All more significant bits agree, and thus need not be removed, and the bit which disagreed must have been a 0 and thus was not asserted. Leaving the agreeing bits asserted reduces system noise caused by the redistribution of driving currents in the bus, and speeds settling of the correct priority on the arbitration bus. This process is a continuous asynchronous parallel process, not a sequential bit-by-bit process as it may seem from the above description. Incorrect comparisons will occur and be removed as the bus lines settle for as long as four bus delays (not related to the choice of four bus lines) plus logic delays.

The four lines which comprise the arbitration bus are DMA0* through DMA3*, where DMA3* is the most significant bit. These lines, in conjunction with HOLD* and pHLDA, control the bus arbitration process.

2.8.3.1 Bus arbitration implementation

An implementation of the bus arbitration protocol is shown in Figures 9a and 9b.

Any implementation shall obey the rules summarized in section 2.8.4.

2.8.3.2 Bus arbitration state definitions

2.8.3.2.1 IWANT

The IWANT state is an internal state for a temporary master which has determined that a bus access is necessary and thus wishes to arbitrate for bus control.

Temporary masters may not assert their priorities nor remove them at arbitrary times, or the arbitration bus may be in transition when the result is needed. A temporary master may assert its priority and the HOLD* bus request only if (1) pHLDA is not asserted (the permanent master has the bus), and (2)
HOLD* is not already asserted. This guarantees an ample time to settle the arbitration bus before the granting of the bus on the rising edge of pHLDA.

This scheme usually results in the first requester winning the bus. Only if simultaneous bus requests occur will the arbitration have any effect. This, however, is not improbable, since multiple unsuccessful requesters will become synchronized by waiting for the falling edge of pHLDA.

2.8.3.2.2 Priority compare states

The priority comparison states, C3 through C0, are the states where each requester compares the priority it is attempting to assert on the arbitration bus with the priority actually on the arbitration bus. Though C3 through C0 are shown and described as sequential, they are actually parallel processes. While disagreement occurs at any bit position, less significant bits are removed from the arbitration bus. If no disagreement persists after the settling time, the requester has the highest priority and will be granted the bus on the rising edge of pHLDA, proceeding to the state “MINE”, where the bus transfer begins. All requesters continue to assert their priorities on the arbitration bus until the falling edge of pHLDA. Thus the priority number of the current bus master is available on the DMA bus while pHLDA is true. If the permanent master has the bus, pHLDA will be false.

A temporary master that wins the bus continues to assert its priority and HOLD* until its bus cycles are complete. A temporary master that loses the bus continues to assert its priority bits not turned off by the arbitration process, but must remove its assertion of the HOLD* line, so that the winner may indicate that it is finished by releasing HOLD*. A losing requester in this state is said to be in the “WAIT” state.

2.8.3.3 Bus arbitration timing relationships

Figure 10 shows two possible cases of the bus arbitration procedure. The first of these is a case where the requester has no competition; it requests the bus and the bus is granted. The second case shows the requester waiting for the bus to be free, arbitrating for the bus and losing, and arbitrating for the bus and winning.

2.8.3.3.1 No competition

When the temporary master determines that it requires the bus, it raises the internal signal IWANT. In this case, the rising edge of IWANT finds the pHLDA signal unasserted, meaning the permanent master has the bus, and the HOLD* signal unasserted, meaning no other devices are requesting the bus. The temporary master may then assert the HOLD* signal and assert its priority on the arbitration bus. The ISME signal is the result of the arbitration process, and is asserted if none of the bit-wise comparisons on the arbitration bus fail. This arbitration result is clocked by the rising edge of the pHLDA signal, creating the bus grant signal MINE.

When the temporary master is finished with the bus, the IWANT signal is released, releasing the HOLD* signal and resetting the bus grant signal, MINE. The permanent master releases the pHLDA signal, and all assertions are removed from the arbitration bus.

2.8.3.3.2 Wait-lose-win

In this example the requester raises its IWANT signal, but finds the bus already busy and must wait to assert its bus request and priority until the falling edge of pHLDA.

The requester arbitrates for the bus during try 1, but another requester has a higher priority and the arbitration result ISME is low at the rising edge of pHLDA, indicating a loss in the arbitration process. The losing requester removes its assertion of the HOLD* signal, but continues to assert the non-conflicting high-order bits of its losing priority until the falling edge of pHLDA. At the falling edge of pHLDA, the process repeats, but this time results in a win for the requester.

2.8.4 Summary of arbitration protocol

Figures 9A and 9B represent an example, not a required implementation. Any implementation which obeys the rules may be used. The rules which must be obeyed by a temporary master are:

1) HOLD* may be asserted only when it is not already asserted and pHLDA is low.
2) HOLD* must be removed when pHLDA rises if another controller has asserted higher priority.
3) HOLD* must be removed when the controller no longer needs the bus.
4) Priority must be asserted whenever HOLD* is asserted, and must remain asserted until the next falling edge of pHLDA.

Figure 10. Bus arbitration timing diagrams.
5) The priority level must be user-selectable by switches and asserted by open-collector drivers on bus lines DMA3*-DMA0*.
6) The most significant bit of the priority level (appearing on DMA3*) must be compared with the priority asserted. If the line is asserted low but not by this temporary master, all less significant priority bit assertions must be removed. Similarly, bits DMA2*, DMA1*, and DMA0* must be examined and possible less significant conflicting bits removed.
7) If no lines are asserted low except those asserted by this temporary master after sufficient settling time, this temporary master has highest priority and may take the bus when pHLDA rises.
8) Logic implementations must be such that setting of the arbitration circuitry and bus will be completed between the assertion of HOLD* and the rise of pHLDA.

2.9 Interrupt protocol

The purpose of an interrupt system is to allow peripheral devices to suspend the operation of a bus master in an orderly way and to request that the master service the requesting peripheral. When service is complete, the bus master returns to the operation from which it was interrupted.

The interrupt protocol is comprised of an 8-level vectored interrupt system and a non-maskable interrupt. A complying master need only implement INT*.

2.9.1 Vectored interrupts

2.9.1.1 Vectored interrupt requests

Eight levels of vectored interrupt requests are issued on the vectored interrupt lines, VIO* thru VI7*, where VIO* is the most significant interrupt priority level. Vectored interrupt requests, however, may be rotated, masked individually, or "fenced out" by the interrupt control slave, and hence the priority levels are not fixed. Requests on the VI lines should be asserted as levels, that is, they should be held active until service is received. A slave which asserts a VI line need take no further action to generate an interrupt. It is assumed that if interrupt acknowledge cycles occur, an interrupt controller somewhere in the system will respond appropriately.

The generalized interrupt request line, INT*, is implemented as a communication line between the interrupt controller and an interruptable master. Any slave or interrupt controller, using the INT* line, must respond appropriately to any interrupt acknowledge cycles. The interrupt controller is not required to use INT*. A vectored interrupt may occur without INT* ever being asserted.

2.9.1.2 Interrupt acknowledge

The interrupt acknowledge cycle is a standard bus read cycle. The interrupt acknowledge cycle requests vectoring information from the interrupt controller to be asserted on the data bus during pDBIN.

Since no address information is asserted during an interrupt acknowledge cycle, only one interrupt controller may exist on the bus. If multiple interrupt controllers exist, they must either be "daisy chained" to avoid possible bus conflicts, or polled by the bus master.

2.9.2 Non-maskable interrupt (NMI*)

The non-maskable interrupt is an optional control input to bus masters. This interrupt is not maskable by a software instruction, and takes priority over other interrupt requests. The NMI* line may be used in the implementation of the special condition lines, ERROR* and PWRFAIL*.

NMI* is an open collector line. The bus master shall respond to negative going transitions on the NMI* line.

2.10 Special condition lines

Two special condition lines, PWRFAIL* and ERROR*, are available on the bus. Their use is optional.

2.10.1 Power-fail pending (PWRFAIL*)

This line indicates an impending system power failure. It is specified that this line shall be activated at least 50 msecs before the local voltage regulators drift out of specification.

The line stays low until the power-on clear signal is activated. This implies that either a normally closed relay or a battery powered circuit drive the power fail line. The circuit driving this line must meet the electrical specifications for an open collector line.

2.10.2 ERROR*

This is a generalized error line that indicates that the current bus operation is producing an error of some sort (i.e., memory parity error, write to protected memory, inability to accommodate 8-bit slaves, etc.)

The ERROR* line should be implemented as a trap. All relevant information about the error-causing cycle—address, data, status, device number (for temporary masters)—should be latched on the falling edge of ERROR*.

ERROR* is implemented as an open collector line.

3.0 Electrical specifications

3.1 Application

This section defines the electrical specifications for interface devices to be used in S-100 bus systems. Proper operation of these devices also depends on two other factors:

1) Short physical distance between devices.
2) Relatively low electrical noise.
The electrical specifications for the bus driver and receiver circuits do not imply a particular technology, unless otherwise noted.

All specifications apply over the temperature range $T_a = 0^\circ\text{C}$ to $70^\circ\text{C}$.

### 3.2 Power distribution

Power in S-100 systems is distributed as unregulated DC power at three voltages, $+8$ volts, $+16$ volts, and $-16$ volts. Because these voltages are on adjacent lines it is relatively easy to short these lines on card removal. Therefore, bleeder resistors or other constant loads sufficient to discharge all three supplies rapidly are recommended.

#### 3.2.1 +8 volt specification

Instantaneous minimum must be greater than $+7$ volts, instantaneous maximum less than 26 volts, and average maximum less than 11 volts.

#### 3.2.2 +16 volt specification

Instantaneous minimum must be greater than $14.5$ volts, instantaneous maximum less than 35 volts, and average maximum less than $21.5$ volts.

#### 3.2.3 -16 volt specification

Instantaneous maximum must be less than $-14.5$ volts, instantaneous minimum greater than $-35$ volts, and average minimum greater than $-21.5$ volts.

### 3.3 General signal discipline

Other than the power lines noted above, all signals on the bus are limited to positive signal levels between 0 volts and $+5$ volts, and may not have loaded rise or fall times less than 5 nsecs.

### 3.4 Driver requirements

#### 3.4.1 Driver types

Three types of bus drivers are defined:

1) An active driver, either in the high state or in the low state or in transition, which has the capability to accept current in the low state and to provide current in the high state.

2) An open collector driver, which will not accept or provide current in the high state. A $1000\Omega \pm 5\%$ pullup resistor to $+5$ volt or equivalent must be provided somewhere in the system for open collector lines. It is recommended that these pullup resistors be provided on the bus. However, implementation on the permanent master is also acceptable.

3) A Tri-State driver, which has the capability to be in the high-impedance state as well as in the high and low states.

#### 3.4.2 Driver specifications

Specifications for bus drivers shall be as follows:

- **Low state ($V_{OL}$):** Output voltage less than or equal to $+0.5$ volts at $24$ mA sink current.

- **High state ($V_{OH}$):** Output voltage (for active and Tri-State drivers) greater than or equal to $+2.4$ volts at $2$ mA.

The leakage current for Tri-State drivers in the high-impedance state is specified as not greater than $\pm 25\mu\text{A}$.

The internal capacitive load of a driver shall not exceed 15 pF at $25^\circ\text{C}$ whether in the active or the high-impedance state.

The rise and fall times of bus drivers should be minimized, subject to 3.3. In no case should the rise or fall times exceed 50 nsec at rated capacitive load.

### 3.5 Receiver specifications

The specifications for receivers on the bus shall be as follows:

- **Low state:** A voltage less than or equal to $+0.8$ volts shall be recognized as a low state.

- **High state:** A voltage greater than or equal to $+2.0$ volts shall be recognized as a high state.

Bus receivers shall source no more than $0.5$ mA at 0.5 volts and sink no more than $50\mu\text{A}$ at 2.4 volts.

Bus receivers shall have diode clamp circuits to prevent excessive negative voltage excursions. Additional noise immunity is afforded by the use of Schmitt-type receiver circuits. Recommended hysteresis for such receivers should be greater than or equal to 0.4 volts.

### 3.6 Bidirectional signals

Some interface signals, such as the data bus, are combined Tri-State drivers and receivers. For each function these devices must meet the same specifications as separate drivers and receivers.

The total internal capacitive load for a line transceiver shall not exceed 20 pF at $25^\circ\text{C}$.

### 3.7 Card-level bus loading

At the card level, the following specifications apply:

1) The total capacitive load on any bus input shall not exceed 25 pF.

2) A card may not source more than $0.5$ mA at 0.5 volts nor sink more than $80\mu\text{A}$ at 2.4 volts on any signal line except for DMA0*, DMA1*, DMA2*, DMA3*, PHANTOM*, and PWRFAIL*. On these lines a card may not source more than $0.4$ mA at 0.5 volts.
3.7.1 Bus termination

All bus lines except the power and ground lines may be terminated to reduce bus noise using a circuit equivalent to

\[ \text{BUS LINE} + \frac{V}{R} \]

where \( V = 2.6 \text{ volts } \pm 0.2 \text{ volts} \) and \( R \) is no less than \( 180 \Omega (\pm 5\%) \).

Open collector lines may have a combination pullup and termination scheme using a circuit equivalent to

\[ \text{OPEN COLLECTOR BUS LINE} + \frac{V}{R_1} \]

where \( V = 2.6 \text{ volts } \pm 0.2 \text{ volts} \), \( R_1 = 1.5K \Omega \pm 5\% \), and \( R_2 \) should be no less than \( 180 \Omega (\pm 5\%) \).

3.8 Read cycle timing specification

Figure 11a depicts the read cycle timing waveforms with the pertinent timing parameters shown. Table 4 specifies these parameters.

3.9 Write cycle timing specification

Figure 11b depicts the write cycle timing waveforms with the pertinent timing parameters shown. Table 4 specifies these parameters.

3.10 Ready and sixteen request timing specification

Figure 12 depicts RDY, XRDY, and SIXTN* timing waveforms during read and write cycles, with pertinent timing parameters shown. Table 4 specifies these parameters.

3.11 Bus transfer timing specification

Figure 8 depicts bus transfer timing waveforms with the pertinent timing parameters shown. Table 5 specifies these parameters.

4.0 Mechanical specifications

4.1 Application

This section defines the mechanical specifications for standard interface systems.

4.2 Connector type

The card edge connector is a 100-pin (dual 50) connector with contacts spaced on 0.125" centers. It is nominally designed for printed circuit boards 0.062" thick.

The connector is subject to the specifications in 4.2.1 and 4.2.2.

4.2.1 Electrical considerations

1) Voltage rating: 200 volts DC, minimum pin to pin.
2) Current rating: 2.5A per contact.
3) Contact resistance: 50 m\( \Omega \) maximum at rated current after 100 insertions.
4) Insulation resistance: 1000 M\( \Omega \) minimum.

4.2.2 Connector spacing

Connectors should be spaced 0.75 inches - 0.01 inches center to center.

Figure 11a. Read cycle timing diagram.

Preliminary—Subject to Revision
Figure 11b. Write cycle timing diagram.

Table 4. Read/write cycle timing parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Description</th>
<th>Min. (ns)</th>
<th>Max. (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td>φ PERIOD</td>
<td>1066</td>
<td>2000</td>
</tr>
<tr>
<td>t&lt;sub&gt;CYH&lt;/sub&gt;</td>
<td>φ PULSE WIDTH HIGH</td>
<td>0.4t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td>0.4t&lt;sub&gt;CY&lt;/sub&gt;</td>
</tr>
<tr>
<td>t&lt;sub&gt;CYL&lt;/sub&gt;</td>
<td>φ PULSE WIDTH LOW</td>
<td>0.4t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td>0.4t&lt;sub&gt;CY&lt;/sub&gt;</td>
</tr>
<tr>
<td>t&lt;sub&gt;SY&lt;/sub&gt;</td>
<td>DELAY φ HIGH TO pSYNC HIGH; DELAY φ LOW TO pSYNC LOW</td>
<td>10</td>
<td>0.4t&lt;sub&gt;CY&lt;/sub&gt;</td>
</tr>
<tr>
<td>t&lt;sub&gt;SY&lt;/sub&gt;</td>
<td>pSYNC PULSE WIDTH HIGH</td>
<td>0.7t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;SST&lt;/sub&gt;</td>
<td>pSYNC PULSE WIDTH LOW PRIOR TO φ LOW DURING pSYNC</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;ST&lt;/sub&gt;</td>
<td>pSYNC PULSE WIDTH HIGH</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;ST&lt;/sub&gt;</td>
<td>pSYNC PULSE WIDTH LOW</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;SST&lt;/sub&gt;</td>
<td>pSYNC PULSE WIDTH HIGH</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;SST&lt;/sub&gt;</td>
<td>pSYNC PULSE WIDTH LOW</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;DB&lt;/sub&gt;</td>
<td>pDBIN PULSE WIDTH HIGH</td>
<td>0.9t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;DB&lt;/sub&gt;</td>
<td>DELAY pDBIN LOW TO pDBIN HIGH</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;DBS&lt;/sub&gt;</td>
<td>DELAY pDBIN LOW TO pSYNC HIGH</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;DBAS&lt;/sub&gt;</td>
<td>HOLD TIME FOR ADDRESSES AND STATUS AFTER pDBIN LOW</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;DBZ&lt;/sub&gt;</td>
<td>DELAY pDBIN LOW TO SLAVE DI DRIVERS HI-Z</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;DBZ&lt;/sub&gt;</td>
<td>DELAY pDBIN LOW TO SLAVE DI DRIVERS ACTIVE</td>
<td>25 + 0.1t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;ACC&lt;/sub&gt;</td>
<td>DELAY pDBIN LOW TO DATA VALID</td>
<td>SPECIFIED BY MANUFACTURER, WORST CASE MAXIMUM FOR ALL SLAVES AND WORST CASE MINIMUM FOR ALL MASTERS.</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;SDB&lt;/sub&gt;</td>
<td>DATA VALID SETUP TIME TO pDBIN LOW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;WR&lt;/sub&gt;</td>
<td>pWR* PULSE WIDTH LOW</td>
<td>0.9t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;SSTWR&lt;/sub&gt;</td>
<td>DELAY pSTVAL* LOW TO pWR* LOW</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;WKSY&lt;/sub&gt;</td>
<td>DELAY pWR* HIGH TO pSYNC HIGH</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;WR&lt;/sub&gt;</td>
<td>SETUP TIME TO pWR* LOW</td>
<td>0.1t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;WASD&lt;/sub&gt;</td>
<td>HOLD TIME ADDRESSES, STATUS, AND DO FROM pWR* HIGH</td>
<td>0.2t&lt;sub&gt;CY&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;WRMR&lt;/sub&gt;</td>
<td>DELAY pWR* LOW TO MWRT HIGH, DELAY pWR* HIGH TO MWRT LOW</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;RDYt&lt;/sub&gt;</td>
<td>SETUP TIME RDY, XRDY, SIXTN* TO φ RISING</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;RDYt&lt;/sub&gt;</td>
<td>HOLD TIME RDY, XRDY, SIXTN* AFTER φ RISING</td>
<td>70</td>
<td></td>
</tr>
</tbody>
</table>
Figure 12. Timing of RDY, XRDY, and SIXTN* during read and write cycles.

Table 5.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{SET} )</td>
<td>( \text{DELAY pHLDA to ADSB}^<em>, \text{SDSB}^</em>, \text{DOSB}^* )</td>
<td>30</td>
</tr>
<tr>
<td>( t_{OV} )</td>
<td>( \text{TIME BOTH TEMPORARY AND PERMANENT MASTER DRIVE THE CONTROL OUTPUT LINES} )</td>
<td>0.5( t_{CY} )</td>
</tr>
<tr>
<td>( t_{DH} )</td>
<td>( \text{HOLD TIME ADDRESS, STATUS, AND DATA OUT DURING DMA CYCLE} )</td>
<td>0.2( t_{CY} )</td>
</tr>
<tr>
<td>( t_{REL} )</td>
<td>( \text{SETUP TIME, END OF BUS TRANSFER TO pHLDA RISING EDGE} )</td>
<td>20</td>
</tr>
</tbody>
</table>

4.3 Board size specification

Circuit boards shall conform to the board size specifications given in Figure 13. The edge connector pin shown in the figure is pin 50. Pin 100 opposes pin 50 on the back side of the board.

Total board depth shall not exceed 0.65".

Nominal board thickness is 0.062".

Figure 13. S-100 board mechanical parameters.
<table>
<thead>
<tr>
<th>State</th>
<th>City</th>
<th>Name</th>
<th>Address</th>
<th>Phone</th>
<th>Time Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>Birmingham</td>
<td>Birmingham Microcomputer Group</td>
<td></td>
<td>-942-6621</td>
<td>8AM-5PM @ 300 baud</td>
</tr>
<tr>
<td>CA</td>
<td>Canoga Park</td>
<td>San Fernando Valley ABBS</td>
<td></td>
<td>213-340-0135</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>Fresno</td>
<td>California CBBS</td>
<td></td>
<td>209-638-6392</td>
<td>CP/M</td>
</tr>
<tr>
<td>CA</td>
<td>Hawthorne</td>
<td>Hawthorne ABBS</td>
<td></td>
<td>213-675-8803</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>Huntington Beach</td>
<td>Kors-Meyer Electronics ABBS</td>
<td></td>
<td>714-964-4346</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>Lawndale</td>
<td>Computer Components Assoc of Orange County</td>
<td></td>
<td>213-370-3160</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>Long Beach</td>
<td></td>
<td></td>
<td>213-648-4748</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>Los Altos</td>
<td>Homebrew Computer Club</td>
<td></td>
<td>415-948-1474</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>Los Angeles</td>
<td>San Fernando CBBS</td>
<td></td>
<td>213-843-5390</td>
<td>CP/M</td>
</tr>
<tr>
<td>CA</td>
<td>Marina Del Ray</td>
<td></td>
<td></td>
<td>213-821-7369</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>San Diego</td>
<td>Apple Corps of San Diego (Computer Merchant)</td>
<td></td>
<td>714-592-9557</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>San Diego</td>
<td>Bill's ABBS (Peoples Message System)</td>
<td></td>
<td>714-449-5689</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>San Diego</td>
<td>Stan Skoglund (Infobit) CBBS</td>
<td></td>
<td>714-565-0761</td>
<td>CP/M</td>
</tr>
<tr>
<td>CA</td>
<td>San Diego</td>
<td>San Diego Computer Society CBBS</td>
<td></td>
<td>714-571-5550</td>
<td>CP/M</td>
</tr>
<tr>
<td>CA</td>
<td>San Francisco</td>
<td></td>
<td></td>
<td>213-668-4246</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>Santa Clar</td>
<td></td>
<td></td>
<td>408-246-2805</td>
<td>CP/M</td>
</tr>
<tr>
<td>CA</td>
<td>Signal Hills</td>
<td>Peripherals Unlimited Inc. ABBS</td>
<td></td>
<td>213-424-3506</td>
<td>ABBS</td>
</tr>
<tr>
<td>CA</td>
<td>Westmester</td>
<td>Computer Components Assoc of Orange County</td>
<td></td>
<td>714-898-1984</td>
<td>ABBS</td>
</tr>
<tr>
<td>DC</td>
<td>Washington</td>
<td>Washington DC CBBS</td>
<td></td>
<td>703-281-2125</td>
<td>?</td>
</tr>
<tr>
<td>DC</td>
<td>Westminster</td>
<td>Program Store Forum 80</td>
<td></td>
<td>202-337-4694</td>
<td>F080</td>
</tr>
<tr>
<td>FL</td>
<td>Destin</td>
<td>Ft. Walton Beach ABBS</td>
<td></td>
<td>904-243-1257</td>
<td>ABBS</td>
</tr>
<tr>
<td>FL</td>
<td>Miami</td>
<td>South Florida Computer Group CBBS</td>
<td></td>
<td>305-821-7401</td>
<td>6:30-8:30PM</td>
</tr>
<tr>
<td>FL</td>
<td>Tampa</td>
<td>Tampa Forum 80</td>
<td></td>
<td>813-223-7688</td>
<td>F080</td>
</tr>
<tr>
<td>GA</td>
<td>Atlanta</td>
<td>Northstar CBBS</td>
<td></td>
<td>404-939-1520</td>
<td>CP/M</td>
</tr>
<tr>
<td>GA</td>
<td>Atlanta</td>
<td>Atlanta Computer Society CBBS</td>
<td></td>
<td>404-394-4220</td>
<td>CP/M</td>
</tr>
<tr>
<td>GA</td>
<td>Atlanta</td>
<td>Atlanta CBBS</td>
<td></td>
<td>404-458-4866</td>
<td>CP/M evenings</td>
</tr>
<tr>
<td>GA</td>
<td>Atlanta</td>
<td>Microstuff Inc CBBS</td>
<td></td>
<td>404-325-0526</td>
<td>CP/M</td>
</tr>
<tr>
<td>GA</td>
<td>Augusta</td>
<td>Augusta Forum 80</td>
<td></td>
<td>803-270-5392</td>
<td>F080</td>
</tr>
<tr>
<td>IL</td>
<td>Chicago</td>
<td>Personal Computers of Chicago</td>
<td></td>
<td>312-337-6331</td>
<td>ABBS</td>
</tr>
<tr>
<td>IL</td>
<td>Chicago</td>
<td>Ward &amp; Randy's Chicago CBBS</td>
<td></td>
<td>312-528-7141</td>
<td>CP/M</td>
</tr>
<tr>
<td>IL</td>
<td>Chicago</td>
<td>Forum 80 of Chicago</td>
<td></td>
<td>312-925-0259</td>
<td>F080</td>
</tr>
<tr>
<td>IL</td>
<td>Joliet</td>
<td>Wayne Jupiter CBBS</td>
<td></td>
<td>815-727-7069</td>
<td>110 baud only</td>
</tr>
<tr>
<td>KS</td>
<td>Olathe</td>
<td>Engineer 80 - Forum 80</td>
<td></td>
<td>913-764-1520</td>
<td>F080</td>
</tr>
<tr>
<td>KS</td>
<td>Wichita</td>
<td>Forum 80</td>
<td></td>
<td>316-746-2078</td>
<td>F080</td>
</tr>
<tr>
<td>MA</td>
<td>Boston</td>
<td>New England Computer Society CBBS</td>
<td></td>
<td>617-864-3819</td>
<td>CP/M 24 hours</td>
</tr>
<tr>
<td>MA</td>
<td>Maynard</td>
<td>NECs/DEC CBBS</td>
<td></td>
<td>617-897-0346</td>
<td>DEC 6PM-7AM &amp; weekends</td>
</tr>
<tr>
<td>MA</td>
<td>Dunstable</td>
<td>Forum 80</td>
<td></td>
<td>617-649-7097</td>
<td>F080</td>
</tr>
<tr>
<td>MD</td>
<td>Parkville</td>
<td>Muse Co.</td>
<td></td>
<td>301-661-8962/3</td>
<td></td>
</tr>
<tr>
<td>MD</td>
<td>Silver Springs</td>
<td>WBETS</td>
<td></td>
<td>301-959-3042</td>
<td></td>
</tr>
<tr>
<td>MI</td>
<td>SEMCO</td>
<td>SEMCO CBBS - Macomb county</td>
<td></td>
<td>313-286-8820</td>
<td>code: HEL-R901, SEMCO</td>
</tr>
<tr>
<td>MI</td>
<td>Westland</td>
<td>SEMCO CBBS - Westland</td>
<td></td>
<td>313-326-6020</td>
<td></td>
</tr>
<tr>
<td>MI</td>
<td>Grosse Pointe</td>
<td>SEMCO CBBS - 300 baud only</td>
<td></td>
<td>313-343-2375</td>
<td></td>
</tr>
<tr>
<td>MI</td>
<td>Grosse Pointe</td>
<td>SEMCO CBBS - 110 baud only</td>
<td></td>
<td>313-343-2370</td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>City</td>
<td>CBBS Type</td>
<td>Code</td>
<td>Notes</td>
<td>Time</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------------------</td>
<td>--------</td>
<td>---------------------</td>
<td>-------</td>
</tr>
<tr>
<td>MO</td>
<td>Kansas City</td>
<td>Kansas City CBBS</td>
<td>EMS</td>
<td>816-737-1031</td>
<td></td>
</tr>
<tr>
<td>MO</td>
<td>Kansas City</td>
<td>Kansas City Forum 80</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO</td>
<td>St. Louis</td>
<td>St. Louis Forum 80</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NE</td>
<td>Las Vegas</td>
<td>Las Vegas Forum 80</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NJ</td>
<td>Bound Brook</td>
<td>SJ Electronic Mail Center</td>
<td></td>
<td>201-457-0893 F080</td>
<td>6800</td>
</tr>
<tr>
<td>NJ</td>
<td>Piscataway</td>
<td>ACG-NJ Apple User Group ABBS</td>
<td>F080</td>
<td>code:HEL-I999,MAIL</td>
<td></td>
</tr>
<tr>
<td>NJ</td>
<td>Marlton</td>
<td>Johnathan's Apple Store ABBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NJ</td>
<td>Princeton</td>
<td>Princeton Forum 80</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NY</td>
<td>New York</td>
<td>Long Island ABBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NJ</td>
<td>Bound Brook</td>
<td>SJ Electronic Mail Center</td>
<td></td>
<td>201-457-0893 F080</td>
<td>6800</td>
</tr>
<tr>
<td>NJ</td>
<td>Bound Brook</td>
<td>SJ Electronic Mail Center</td>
<td></td>
<td>201-457-0893 F080</td>
<td>6800</td>
</tr>
<tr>
<td>TX</td>
<td>Dallas</td>
<td>Dallas CBBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>Dallas</td>
<td>Dallas Forum 80</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>Dallas</td>
<td>KA Electronics ABBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>Dallas</td>
<td>Computer Hobbyists Group of N Texas CBBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>Fort Worth</td>
<td>Fort Worth Forum 80</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>Fort Worth</td>
<td>CHG-NT CBBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>Houston</td>
<td>Houston ABBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>San Antonio</td>
<td>Appleseeded ABBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>Wichita Falls</td>
<td>Forum 80</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VA</td>
<td>Falls Church</td>
<td>North Star CBBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VA</td>
<td>McLean</td>
<td>Paul's ABBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VA</td>
<td>Fairfax</td>
<td>Family Historians Forum 80</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VA</td>
<td>Vienna</td>
<td>AMRAD CBBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WA</td>
<td>Puget Sound</td>
<td>Washington ABBS</td>
<td>F080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TN</td>
<td>Memphis</td>
<td>Memphis Forum 80</td>
<td>F080</td>
<td>Handicapped Appl;</td>
<td></td>
</tr>
</tbody>
</table>

Fort Lauderdale area with ASCII on 229 MHz and 400 MHz.

The list of CBBS's across the country is growing. Some are dying out after a short existence, but for every one that folds, there are at least three new ones to take it's place. The following is a list of some of the CBBS's and no doubt there are many more that I am not aware of. We will reprint and updated directory of CBBS systems periodically. So if you or your club has a CBBS system that you would like to have listed send me the information care of S-100 MICROSYSTEMS.

CBBS CODES USED:

- CP/M - indicates a CBBS using a CP/M system
- ABBS - indicates a CBBS using an Apple system
- F080 - indicates a CBBS using a TRS-80 system
- 6800 - indicates a CBBS using a 6800 system

Notes:

additional notes are given in the extreme right hand column. Some CBBS's require a password which is listed there.
HI THERE!

It has come to our attention that many of our customers are not aware of the wide variety of products we have available. Following is a list of all our current major products and prices.

<table>
<thead>
<tr>
<th>Part No.</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM-32K-ASM</td>
<td>Tarbell 32k Static Memory for S-100 bus</td>
<td>725.00</td>
</tr>
<tr>
<td>MEM-16K-ASM</td>
<td>Tarbell 16k Static Memory for S-100 bus (Memories only come assembled &amp; checked out)</td>
<td>440.00</td>
</tr>
<tr>
<td>CI-ASM</td>
<td>Cassette Interface Assembled &amp; Checked out</td>
<td>175.00</td>
</tr>
<tr>
<td>CI-KIT</td>
<td>Cassette Interface Kit</td>
<td>120.00</td>
</tr>
<tr>
<td>FDI-ASM</td>
<td>Floppy Disk Interface Assembled &amp; Checked out</td>
<td>265.00</td>
</tr>
<tr>
<td>FDI-KIT</td>
<td>Floppy Disk Interface Kit</td>
<td>190.00</td>
</tr>
<tr>
<td>DD-ASM</td>
<td>Double Density Floppy Disk Interface A&amp;T</td>
<td>425.00</td>
</tr>
<tr>
<td>DD-KIT</td>
<td>Double Density Floppy Disk Interface Kit</td>
<td>325.00</td>
</tr>
<tr>
<td>VDS-II</td>
<td>Vertical Disk Subsystem</td>
<td>1888.00</td>
</tr>
<tr>
<td>VDS-IID</td>
<td>Double Density Disk Subsystem</td>
<td>1999.00</td>
</tr>
<tr>
<td>PS270</td>
<td>PerSci Model 270 Dual Floppy Disk Drive</td>
<td>1295.00</td>
</tr>
<tr>
<td>STM120</td>
<td>Siemens Model 120-8 Single Floppy Disk Drive</td>
<td>495.00</td>
</tr>
<tr>
<td>SHU800</td>
<td>Shugart 800/801 Floppy Disk Drive</td>
<td>525.00</td>
</tr>
<tr>
<td>CP272</td>
<td>Power Supply for PerSci 270 Dual Drive</td>
<td>125.00</td>
</tr>
<tr>
<td>CP206</td>
<td>Power Supply for PerSci 299</td>
<td>120.00</td>
</tr>
<tr>
<td>CP206</td>
<td>Power Supply for Two Siemens or Shugart</td>
<td>120.00</td>
</tr>
<tr>
<td>TBAS-CAS</td>
<td>Tarbell Cassette BASIC on cassette incl manual</td>
<td>48.00</td>
</tr>
<tr>
<td>TBAS-DSK</td>
<td>Tarbell Disk BASIC on CP/M disk incl manual</td>
<td>48.00</td>
</tr>
<tr>
<td>TBAS-LST</td>
<td>Tarbell BASIC source listing on paper</td>
<td>25.00</td>
</tr>
<tr>
<td>TBAS-SRC</td>
<td>Tarbell BASIC source on 2 CP/M disks</td>
<td>25.00</td>
</tr>
<tr>
<td>PTSW+LST</td>
<td>Proc Tech Assembler/Editor with listing</td>
<td>15.00</td>
</tr>
<tr>
<td>EMPL-CAS</td>
<td>EMPL micro APL on cassette &amp; instructions</td>
<td>15.00</td>
</tr>
<tr>
<td>EMPL-DSK</td>
<td>EMPL micro APL on CP/M disk &amp; instructions</td>
<td>20.00</td>
</tr>
<tr>
<td>CPM-2.0</td>
<td>CP/M 2.0 Operating system incl documentation</td>
<td>150.00</td>
</tr>
<tr>
<td>CPM-1.4</td>
<td>CP/M 1.4 Operating system incl BASIC-E on disk</td>
<td>70.00</td>
</tr>
<tr>
<td>CPM-1.4-M</td>
<td>CP/M 1.4 on soft-sectored MINI-FLOPPY disk</td>
<td>70.00</td>
</tr>
<tr>
<td>CPM-MS</td>
<td>CP/M 1.4 Operating system manual set (of six)</td>
<td>25.00</td>
</tr>
<tr>
<td>BASE-MAN</td>
<td>BASIC-E Compiler Manual (works with CP/M)</td>
<td>5.00</td>
</tr>
<tr>
<td>BASE-LST</td>
<td>BASIC-E Source Listing (in PL/M)</td>
<td>15.00</td>
</tr>
<tr>
<td>PBLC-DMN-1</td>
<td>Public Domain Disk #1 - includes DISKTEST, BASIC-E, BIOS, FORMAT, TAPELIB, etc on disk</td>
<td>10.00</td>
</tr>
<tr>
<td>PBLC-DMN-2</td>
<td>Public Domain Disk #2 - includes Double Density FORMAT, DISKTEST, Auto-BIOS for 1.4 &amp; 2.0; FORTH</td>
<td>15.00</td>
</tr>
<tr>
<td>CBAS-DSK</td>
<td>CBASIC-2 disk</td>
<td>85.00</td>
</tr>
<tr>
<td>CBAS-MAN</td>
<td>CBASIC-2 manual</td>
<td>15.00</td>
</tr>
<tr>
<td>SPLR</td>
<td>KLH Systems Spooler for CP/M on disk</td>
<td>70.00</td>
</tr>
<tr>
<td>FAST</td>
<td>FAST! Screen-oriented Editor/Assembler for CP/M</td>
<td>100.00</td>
</tr>
<tr>
<td>TELE-COM</td>
<td>Software to operate D.C. Hayes Modem Remote</td>
<td>195.00</td>
</tr>
<tr>
<td>POLYVUE</td>
<td>Screen-Oriented CP/M Editor</td>
<td>135.00</td>
</tr>
<tr>
<td>PASCAL/MT</td>
<td>Meta-Tech Pascal Compiler for CP/M</td>
<td>99.95</td>
</tr>
</tbody>
</table>

Prices are subject to change without notice.
California residents please add 6% sales tax.
For quick delivery, see your local Tarbell dealer.

(213) 538-4251
950 Dovlen Place, Suite B
Carson, California 90746

(213) 538-2254
AN 8080 DISASSEMBLER
by
William Yarnall
ACG-NJ
1176 Raritan Rd.
Scotch Plains, NJ 07076

A Disassembler program will take object code, in RAM, and yield a source code output. It is an invaluable software utility. This Disassembler program takes up only 1K of memory space and is very easy to use.

This is a very simple and easy to use disassembler program than can be used with virtually any 8080 microcomputer system. Previous to my getting disks I used it with my audio cassette based system. I now use it on my IMSAI with North Star Disk System, in both CP/M and North Star DOS.

I have several versions on disk that are assembled to run at different memory locations. For North Star DOS I have versions that are assembled to run at 0, 2A00H, 5800H and D000H. For CP/M I have versions assembled to run at 100H, 5800H and D000H. I then load the appropriate version for the software I am going to disassemble. For example, to disassemble North Programs that have an origin of 2A00H, I load and run the disassembler at memory location 0. To disassemble a CP/M program that has an origin at 100H, I load and run the disassembler in high memory (e.g. 5800H or D000H depending on where CP/M is located).

The version of disassembler shown here is set up to run with my monitor program that starts at E000H. Therefore the program starts with "global definitions" that define the locations of the console (keyboard and VDM display in my case) I/O driver routines and the return address for the monitor or operating system. These equates should therefore be changed to suit your particular DOS or monitor program.

Routine LOOP, near the beginning of the program reads the sense switches on the front panel of my IMSAI-8080. If switch 2 is raised then the disassembly is interrupted and control is returned to the operating system. This allows me to interrupt a run at any point before the run is completed, should I desire to do so. If your system does not have a front panel with sense switches and you want to be able to interrupt a run then replace the IN OFFH and ANI 2 instructions with a CALL to the keyboard status routine followed by a jump to the operating system. Then any keypress will interrupt execution and return to the operating system. This works if the keyboard input routine returns with a non-zero. If it returns a zero change JNZ EEND to JZ EEND.

To use the disassembler program load it into memory and then type the starting address of the memory area to be disassembled followed by a comma and the ending address to be disassembled.
MVI D, 0
DAD H
DAD H
DAD H
DAD H
DAD H
JMP GINO

GIN1: XRA A
XCHG
LOOP: IN 0FFH ; SENSE SWITCH
ANI 2 ; NO. 2 TO QUIT
JNZ END ; RETURN TO MONITOR
CALL CLER
CALL PROC
CALL DISP
JMP LOOP

; CONVERT ASCII TO HEX
NYBLE: CPI '0'
RC
CPI 'F' + 1
CMC
RC
SUI '0'
CPI 10
CMC
RNC
SUI 7
CPI 10
RET

; ADD (A) TO H, L
ADDH: ADD L
MOV L, A
RNC
INR H
RET

; GET AND STORE MNEMONIC
; (HL) POINTS TO MNEMONIC TABLE
CARD: PUSH H
MVI A, ' '
CRD0: STA PMNE + 3
MOV A, M
ORA A
JZ CRD2
MOV B, A
RRC
RRC
RRC
RRC
LXI CALL
MOV A, M
RNC
RRC
RRC
RRC
RRC
ANI 1FH
ADI 40H
STA PMNE
INX H
MOV C, M
MOV A, B
ANI 7
RLC
LC
MOV B, A
MOV A, C
RLC
RLC
ANI 3
ORA B
ADI 40H
STA PMNE + 1
MOV A, C
ANI 1FH
JZ CRD1
ADI 40H
STA PMNE + 2
CRD1: POP H
INX H

S-100 MICROSYSTEMS
CR2: LXI H,REG2
CALL ADDH
MOV A,M
STA PARG
POP PSW

CR3:
RLC
JNC CR7

; 1-BYTE CONSTANT ARGUMENT
INX D
LDAX D
MOV H,A
PUSH H
LXI B,PARG+2
CALL C2D
POP H

CR3A:
LXI B,PC2
CALL C2D

CR4:
LDA PARG
CPI ' '
JZ CR5
MVI A,','
STA PARG+1

CR5:
CALL JUST
RET

CR6:
ANI 7
JMP CR2

CR7:
RLC
JNC CR8

; 2-BYTE CONSTANT ARGUMENT
INX D
LDAX D
MOV L,A
INX D
LDAX D
MOV H,A
PUSH H
LXI B,PARG+2
CALL C4D
POP H
LXI B,PC3
CALL C2D
JMP CR3A

CR8:
LDAX D

; TEST FOR "RST" AS SPECIAL CASE
ANI 0CH
CPI 0CH
JNZ CR5
LDA B

S-100 Boards
Video and/or Analog
Data Acquisition
Microcomputer Systems

ANALOG Boards
A/D 16 Channel, $495.
12 Bit, High Speed
D/A 4 Channel, $395.
12 Bit, High Speed

VIDEO DIGITIZATION
Real Time Video $850.
Digitizer and Display
Computer Portrait
System $4950.

8086 Boards
CPU with $650.
Vectored Interrupts
PROM-I/O $495.
RAM $395.
8K x 16/16K x 8

50
C2D: PUSH D
   MVI D,2
   JMP CVD

; CONVERT 2 BYTES TO 4 ASCII CHAR.
CVD: PUSH D
   MVI D,4
   XRA A
   DAD H
   RAL
   DAD H
   RAL
   DAD H
   RAL
   CPI 10
   JC CVD0
   ADI 7

CVD0: ADI '0'
   STAX 8
   INX 8
   DCR D
   JNZ CVD
   POP D
   RET

; DISPLAY LINE
DISP: PUSH H
   LXI H, PLOC
   DISPO: MOV A,M
   CALL OUT8
   CPI 13
   JZ DISP1
   INX H
   JMP DISP0
   DISP1: POP H
   RET

; PROCESS COMMAND (DE POINTS TO COMMAND)
PROC: PUSH D
   LDAX D
   CPI 40H
   JNC PRB

; COMMAND 00 - 3F
   ANI 7
   RLC
   MOV C,A
   RLC
   ADD C
   LXI H,PRA
   CALL ADDH
   PCHL
   LXI H,TB1
   JMP PRA4
   LXI H,TB2
   JMP PRA5
   LXI H,TB3
   JMP PRA1
   LXI H,TB4
   JMP PRA5
   LXI H,TB5
   JMP PRA3
   LXI H,TB6
   JMP PRA3
   LXI H,TB7
   JMP PRA3
   LXI H,TB8
   LXI H,TB9
   MOV A,M
   STA PARG+2
   POP D
   INX D
   RET

PRC: CPI OCOH
   JNC PRD

; COMMAND 80 - BF
   LXI H,TB10
   JMP PRA1

PRD: SUI OCOH
   MOV C,A
   RLC
   ADD C
   LXI H,TB11
   JMP PRA2

; REGISTER TABLES
REG1: DB 'B'
   DB 'D'
   DB 'H'
   DB 'S'
   DB 'C'
   DB 'E'
   DB 'H'
   DB 'I'
   DB 'M'
   DB 'A'

REG2: DB 'B'
   DB 'D'
   DB 'E'
   DB 'H'
   DB 'I'
   DB 'M'
   DB 'A'

S-100 MICROSYSTEMS
<table>
<thead>
<tr>
<th>COMMAND PROCESSING TABLES</th>
<th>TB11:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTAINS MNEMONICS &amp; PROCESSING FLAGS</td>
<td>DW 9A93H</td>
</tr>
<tr>
<td>TB1:</td>
<td>DW OD073H</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 922CH</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td>TB2:</td>
<td>DW 966H</td>
</tr>
<tr>
<td></td>
<td>DB 90H</td>
</tr>
<tr>
<td></td>
<td>DW 4420H</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td>TB3:</td>
<td>DW 0</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td></td>
<td>DW 300H</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td></td>
<td>DW 0</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td></td>
<td>DW 300H</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td></td>
<td>DW 600H</td>
</tr>
<tr>
<td></td>
<td>DB 10H</td>
</tr>
<tr>
<td></td>
<td>DW 900H</td>
</tr>
<tr>
<td></td>
<td>DB 10H</td>
</tr>
<tr>
<td></td>
<td>DW 19DH</td>
</tr>
<tr>
<td></td>
<td>DB 10H</td>
</tr>
<tr>
<td></td>
<td>DW 161H</td>
</tr>
<tr>
<td></td>
<td>DB 10H</td>
</tr>
<tr>
<td>TB4:</td>
<td>DW 984BH</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td></td>
<td>DW 0D820H</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td>TB5:</td>
<td>DW 924BH</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td>TB6:</td>
<td>DW OD220H</td>
</tr>
<tr>
<td></td>
<td>DB 80H</td>
</tr>
<tr>
<td>TB7:</td>
<td>DW 896DH</td>
</tr>
<tr>
<td></td>
<td>DB 60H</td>
</tr>
<tr>
<td>TB8:</td>
<td>DW 393H</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 8394H</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 4C90H</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 5290H</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 4120H</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 411BH</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 39DH</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 431BH</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td>TB9:</td>
<td>DW 1443H</td>
</tr>
<tr>
<td></td>
<td>DB 0</td>
</tr>
<tr>
<td></td>
<td>DW 409H</td>
</tr>
<tr>
<td></td>
<td>DB 40H</td>
</tr>
<tr>
<td></td>
<td>DW 309H</td>
</tr>
<tr>
<td></td>
<td>DB 40H</td>
</tr>
<tr>
<td></td>
<td>DW 429DH</td>
</tr>
<tr>
<td></td>
<td>DB 40H</td>
</tr>
<tr>
<td></td>
<td>DW 8298H</td>
</tr>
<tr>
<td></td>
<td>DB 40H</td>
</tr>
<tr>
<td></td>
<td>DW 8108H</td>
</tr>
<tr>
<td></td>
<td>DB 40H</td>
</tr>
<tr>
<td></td>
<td>DW 81C4H</td>
</tr>
<tr>
<td></td>
<td>DB 40H</td>
</tr>
<tr>
<td></td>
<td>DW 817CH</td>
</tr>
<tr>
<td></td>
<td>DB 40H</td>
</tr>
<tr>
<td></td>
<td>DW 501BH</td>
</tr>
<tr>
<td></td>
<td>DB 40H</td>
</tr>
</tbody>
</table>

continued on page 56
**PROC CRTCTL:**

**VIDEO DISPLAY BOARD CONTROL SOFTWARE**

**FOR SDS SALE VDB-8024. JON BONDY, 5/16/79**

**PORTS ASSIGNED TO REGISTERS INSIDE THE SMC 5027 CHIP**

- **REG0**
  - 80H: MODE/HSYNC WIDTH/HSYNC DELAY
- **REG1**
  - 81H: MODE/HSYNC WIDTH/HSYNC DELAY
- **REG2**
  - 82H: VERTICAL DATA START
- **REG3**
  - 83H: ISKewn BIT DATA ROW PER FRAME
- **REG4**
  - 84H: ISKewn LINES PER FRAME
- **REG5**
  - 85H: ISKewn LINES PER FRAME
- **REG6**
  - 86H: LAST DISPLAYED DATA ROW
- **RESET**
  - 8AH: RESET 5027
- **WCHAR**
  - 8CH: WRITE CURSOR CHARACTER/COLUMN ADDRESS
- **WCLINE**
  - 8OH: WRITE CURSOR LINE ADDRESS
- **STIMCH**
  - 8EH: WRITE CURSOR LINE ADDRESS

**CRTIN**
- 20H: INPUT PORT FROM WHICH CHARACTERS FROM THE COMPUTER ARE RECEIVED

**NUMBER**
- 2AH: NUMBER OF ROWS
- 2AH: NUMBER OF COLUMNS
- 2BH: TOP OF STACK
- 2BH: TOP OF DISPLAY MEMORY
- 2BH: ASCII BLANK

**A NOTE ON THE 5027**

In order to scroll the display rapidly, the 5027 does not move data around in the display memory in order to scroll. Rather, it starts reading data for refresh at a different location in the memory, and "wraps around" from the bottom of the buffer to the top when it needs to, because of this use of a circular buffer, the upper left character on the screen need not always be (and most often is not) the initial byte in the refresh memory. A register in the 5027 is used to keep track of where the refresh is to stop (and hence, where it is to start), and the following firmware must keep track of this in order that it always know where the top of the user's display is. This is the main function of the 'B' register, which always maintains a copy of the last line number which the 5027 will display.

In the following code, two similar tests occur frequently; one to see if the cursor has moved outside of the buffer (and hence should be moved to the top of the buffer in order to wrap around), and one to see if the cursor has moved outside of the screen (i.e., below the last line or above the first line). If the reader allows these two tests to become confused with each other, the code will make no sense at all.

**REGISTER USE**

- A: SCRATCH
- B: LAST DISPLAYED LINE -- SEE ABOVE DISCUSSION ON 5027
- C: SCRATCH
- D: COLUMN NUMBER FOR CURSOR
- E: LINE NUMBER FOR CURSOR

**POWER UP INITIALIZATION**

```
POF 1Y ;SAVE RETURN ADDRESS
INC A,(CRTIN) ;IS THIS A CURSOR CONTROL CHAR ?
JP NZ,DISPLAY ;YES
JR Z,CTR ;TEST AGAIN FOR CONTROL CHAR
JR 5,A ;TEST FOR OTHER CONTROL CHAR
LD A,067H ;SET UP DISPLAY MEMORY
INC CP
INC JR
OUT
HLT SCF
EI
HALT
```

**DISPLAY CHARACTER**

```
LD C,A ;CHAR SAVED IN C REG
LD A,D ;PLAY WITH COLUMN COUNTER
INC A ;IS NEXT CHAR ON SAME LINE ?
CP NUMCOL
JR Z,HLT
LD (HL)+C ;PUT CHAR IN DISPLAY MEMORY
INC D ;CLEAN UP COLUMN POINTER
INC HL ;CLEAN UP MEMORY POINTER
OUT (WCHAR)+A ;REPOSITION CURSOR ON SCREEN
HLT
SCF
EI
HALT
```

**INITIALIZE CRT 5027 CHIP**

```
INIT OUT (RESZT)+A ;SEE 5027 DATA SHEET FOR MEANING OF THESE
LD A,065H
OUT (REGO)+A
LD A,04BH
OUT (REG1)+A
LD A,044H
OUT (REG2)+A
LD A,007H
OUT (REG3)+A
LD A,003H
OUT (REG4)+A
LD A,08H
OUT (REG5)+A
LD A,NUMROW-1
```

**MODIFYING THE SDS VDB-8024**

Continued from page 14
LD B,A
OUT (REG6),A
LD A:O
OUT (REG7),A
OUT (WCHAR),A
OUT (WCLINE),A
OUT (STIMCH),A
OUT (RESET),A
LD D:O
RET

;LAST LINE DISPLAYED

LD IY,HLT
CALL LF
CALL CR

;SET UP TO RETURN TO HALT INSTRUCTION

BRANCH ON CONTROL CHARACTERS

CR
LD D:O
LD A:B
OUT (WCHAR),A
OUT (WCLINE),A
OUT (STIMCH),A
OUT (RESET),A
LD D:O
RET

;SET COLUMN NUMBER TO ZERO

CRLF
CALL LF
CALL CR

;MOVE CURSOR RIGHT

LFRT
LD E:O
OUT (WCHAR),A
OUT (WCLINE),A
LD D:O
LD A:B
OUT (WCHAR),A
OUT (WCLINE),A
OUT (STIMCH),A
OUT (RESET),A
LD D:O
RET

;SET UP TO COMPARE CURRENT LINE NUMBER WITH LAST LINE IN BUFFER

;WITH LAST LINE IN BUFFER

CALL Z:SCRUP
IF THEY MATCH, MUST SCROLL UP
INC A
OTHERWISE, MOVE TO NEXT LINE
LD A:O
LD B:A
JR NZ,LFRT
INC A
YES -- SEEN NUMBER TO TOP OF BUFFER
ADD A:L
;

;SET UP TO FILL NEW LINE WITH BLANKS

;CALCULATE ADDRESS OF LAST CHAR...

;...IN LINE FOR LOOP TERMINATION

;IF "E" CONTAINS LINE NO.

;CONVERT ROW AND COLUMN INTO ADDRESS IN...

;CLEAN UP REFRESH MEMORY ADDRESS

;SET UP TO RETURN TO HALT INSTRUCTION

;PUSH RETURN ADDR

;TAB

;LINE FEED

;CARRIAGE RETURN

;MOVE CURSOR LEFT

;BACK SPACE

;IF COLUMN WAS ZERO...

;ATTEND TO MOVE OFF OF SCREEN

;DEC H: TO DECREMENT COLUMN NUMBER

;DEC H: IF MEMORY POINTER CORRECT ALSO

;TELL 5027 WHAT WE'RE DOING SO THAT...

;CURSOR WILL BE CORRECT

;COPY COLUMN COUNTER AND...

;INCREMENT COPY IN 'A'

;DID WE MOVE OFF OF SCREEN?

;YES -- FORGET THE MODIFICATION

;INCREMENT COLUMN AND...

;MEMORY POINTERS

;KEEP 5027 INFORMED

;INCREMENT LAST DISPLAYED LINE

;SET UP TO COMPARE

;IS NEW ROW LAST ROW IN MEMORY?

;HOFER

;YES -- WRAP 'B' REG AROUND

;SET UP MEMORY POINTER FROM ROW,COL

;SET UP TO FILL NEW LINE WITH BLANKS

;CALCULATE ADDRESS OF LAST CHAR...

;...IN LINE FOR LOOP TERMINATION
SCRLP LD (HL),C "FILL WITH BLANKS"
INC HL
CP
JR NZ, SCRLP
LD A, B
OUT (REG6), A "CHANGE LAST DISPLAYED LINE ADDRESS IN 5027"
RED OUT (WCCHAR), A "LINE FEED ROUTINE EXPECTS THIS SETUP"

CLEAR SCREEN
CLR SCREEN
LD A, 08H "THIS IS FUNNY HARDWARE HACK TO DISABLE DMA..."
OUT (10H), A "...FOR 5027 (I THINK)"
LD HL, STARTING ADDRESS
LD C, BLANK "BLANK CHARACTER"
CLR2 LD A, 050H "FLOOR TERMINATION TEST VALUE"
INC HL "FOR FIRMWARE BOOKKEEPING, SO MUST BE..."
CP L ...PROTECTED FROM CLEAR
JR NZ, CLR2
LD A, 01H "NOW, WE TEST 'H', FINALLY"
CLR CALL HOME "HOME -- HOME CURSOR"
LD A, 20H "IF NOT DONE, START AGAIN AT 'CLR1'"
OUT (10H), A "SET UP 5027 DMA AGAIN (I THINK)"

CURSOR CONTROL ROUTINES
JF' M, XCNTL "'<' IS X-CONTROL, '>' IS Y-CONTROL"
SUB 210 "THIS IS Y-CONTROL CHAR"
CP NUMROW "CHECK VALUE < NUMBER OF ROWS"
JP HLT "INVALID, SO IGNORE"
ADD A, B "ADD ROW 0 TO LAST DSP ROW"
INC A "IS IT OUTSIDE OF BUFFER (WRAP AROUND)"
CP NUMROW "IS IT OUTSIDE OF BUFFER (WRAP AROUND)"
JP M, SKIP2 "YES -- SUBSTITUTE VALUE"
SUB NUMROW "USE NEW ROW VALUE"
CALL PLACE "PLACE CURSOR CORRECTLY"
JP HLT "FINISHED"
XCTLN SUB 128 "COLUMN CODE STARTS HERE"
CP NUMCOL "GREATER THAN VALID COLUMN"
JP HLT "NO OTHER CHECKS REQUIRED -- PLACE CURSOR"
CALL PLACE "PLACE CURSOR AT LOCATION IN D/E"
JP HLT "PLACE CURSOR AT LOCATION IN D/E"

CURSOR UP ONE LINE
UPLINE LD A, E "DO WE HAVE TO MOVE UP IN WRAP AROUND MODE?"
CP D "NOPE -- NORMAL NUMBER"
JR NZ, ULP LP "YES -- SUBSTITUTE LARGEST LINE NUMBER"
DEC A "DEC INCREMENT IN EITHER CASE"
RET Z "YES -- IGNORE ATTEMPT TO MOVE OFF OF SCREEN"
LD E, A "VALUE IS D.K. -- SAVE IT"
CALL PLACE "PLACE CURSOR AT LOCATION IN D/E"
RET "PLACE CURSOR AT LOCATION IN D/E"

CURSOR DOWN ONE LINE
DL LINE LD A, E "ARE WE TRYING TO MOVE DOWN FROM LAST LINE?"
CP B "YES -- IGNORE ATTEMPT TO MOVE OFF OF SCREEN"
RET Z "DO WE HAVE TO WRAP AROUND BUFFER?"
CP NUMROW-1 "DO WE HAVE TO WRAP AROUND BUFFER?"
JR NZ, DL NLP "YES -- PRE-LOAD DUMMY VALUE"
INC A "INCREMENT ROW IN ANY CASE"
LD E, A "SAVE NEW VALUE"
CALL PLACE "PLACE CURSOR AT LOCATION IN D/E"
RET "PLACE CURSOR AT LOCATION IN D/E"

PLACE CURSOR AT POSITION SPECIFIED IN 'D' AND 'E'
'D' IS COLUMN NUMBER, 'E' IS LINE NUMBER

CURSOR CONTROL ROUTINES
JF' M, XCNTL "'<' IS X-CONTROL, '>' IS Y-CONTROL"
SUB 210 "THIS IS Y-CONTROL CHAR"
CP NUMROW "CHECK VALUE < NUMBER OF ROWS"
JP HLT "INVALID, SO IGNORE"
ADD A, B "ADD ROW 0 TO LAST DSP ROW"
INC A "IS IT OUTSIDE OF BUFFER (WRAP AROUND)"
CP NUMROW "IS IT OUTSIDE OF BUFFER (WRAP AROUND)"
JP M, SKIP2 "YES -- SUBSTITUTE VALUE"
SUB NUMROW "USE NEW ROW VALUE"
CALL PLACE "PLACE CURSOR CORRECTLY"
JP HLT "FINISHED"
XCTLN SUB 128 "COLUMN CODE STARTS HERE"
CP NUMCOL "GREATER THAN VALID COLUMN"
JP HLT "NO OTHER CHECKS REQUIRED -- PLACE CURSOR"
CALL PLACE "PLACE CURSOR AT LOCATION IN D/E"
JP HLT "PLACE CURSOR AT LOCATION IN D/E"

CURSOR UP ONE LINE
UPLINE LD A, E "DO WE HAVE TO MOVE UP IN WRAP AROUND MODE?"
CP D "NOPE -- NORMAL NUMBER"
JR NZ, ULP LP "YES -- SUBSTITUTE LARGEST LINE NUMBER"
DEC A "DEC INCREMENT IN EITHER CASE"
RET Z "YES -- IGNORE ATTEMPT TO MOVE OFF OF SCREEN"
LD E, A "VALUE IS D.K. -- SAVE IT"
CALL PLACE "PLACE CURSOR AT LOCATION IN D/E"
RET "PLACE CURSOR AT LOCATION IN D/E"

CURSOR DOWN ONE LINE
DL LINE LD A, E "ARE WE TRYING TO MOVE DOWN FROM LAST LINE?"
CP B "YES -- IGNORE ATTEMPT TO MOVE OFF OF SCREEN"
RET Z "DO WE HAVE TO WRAP AROUND BUFFER?"
CP NUMROW-1 "DO WE HAVE TO WRAP AROUND BUFFER?"
JR NZ, DL NLP "YES -- PRE-LOAD DUMMY VALUE"
INC A "INCREMENT ROW IN ANY CASE"
LD E, A "SAVE NEW VALUE"
CALL PLACE "PLACE CURSOR AT LOCATION IN D/E"
RET "PLACE CURSOR AT LOCATION IN D/E"

PLACE CURSOR AT POSITION SPECIFIED IN 'D' AND 'E'
'D' IS COLUMN NUMBER, 'E' IS LINE NUMBER
AN 8080 DIASSEMBLER
Continued from page 52

```
DB 0
DW 0F1CH
DB 10H
DW 0C00H
DB 80H
DW 890BH
DB 20H
DW 0D494H
DB 0
DW 594H
DB 0
DW 1B00H
DB 0
DW 554H
DB 10H
DW 1500H
DB 0
DW 51CH
DB 10H
DW 922CH
DB 0
DW 89C4H
DB 20H
DW 0D494H
DB 0
DW 94H
DB 0
; 4-CHARACTER MNEMONIC TABLE
DW 0D083H
TBX:  DW 9D58H
DB 80H
DB 54H
DW 10H
DB 0
DB 0
DB 4022H
DB 401BH
DB 922CH
DB 91CH
DB 20H
DB 0D494H
DB 0
DB 9D58H
DB 1
DB 6158H
```

S-100 MICROSYSTEMS
CHARTER SUBSCRIPTION OFFER!

S-100 MICRO SYSTEMS

THE ONLY MAGAZINE BY AND FOR S-100 SYSTEM USERS!
TAKE ADVANTAGE OF THIS LIMITED TIME CHARTER SUBSCRIPTION OFFER!

At last there is a magazine written exclusively for S-100 system users. No other publication is devoted to supporting S-100 system users. No longer will you have to hunt through other magazines for an occasional S-100, CP/M* or PASCAL article. Now find it all in one publication. Find it in S-100 MICROSYSTEMS.

SOFTWARE
CP/M*
Assembler
BASIC
PASCAL
applications
and lots more

HARDWARE
8 bit & 16 bit CPUs
interfacing
hardware mods
bulletin board systems
multiprocessors
and lots more

SYSTEMS
Cromemco
North Star
IMSAI
SOL
Polymorphics

Every issue of S-100 MICROSYSTEMS brings you the latest in the S-100 world. Articles on applications, tutorials, software development, letters to the editor, newsletter columns, book reviews, new products, etc. Material to keep you on top of the ever changing microcomputer scene.

*Registered Trademark Digital Research

Take advantage of our low charter subscription offer right now! This offer runs out on April 30, 1980.

Published every other month
Edited by Sol Libes and Russell Gorr

Send me ______Issues of
S-100 MICRO SYSTEMS

□ Bill me with first issue
□ Payment enclosed

Name __________________________

Address __________________________

City __________________________

State __________________________ Zip __________

CPU __________________________ Disk System __________________________

Clip and mail to: S-100 MICROSYSTEMS
P.O. Box 1192 Mountainside, NJ 07092

SUPER OFFER!
...for charter subscribers only

<table>
<thead>
<tr>
<th>No. of Issues</th>
<th>Bill Me</th>
<th>Payment enclosed</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 issues</td>
<td>$8.00</td>
<td>$7.50</td>
</tr>
<tr>
<td>12 issues</td>
<td>$15.00</td>
<td>$14.00</td>
</tr>
<tr>
<td>18 issues</td>
<td>$23.00</td>
<td>$21.50</td>
</tr>
</tbody>
</table>

Canada: $9.00 per year
Foreign: $20 per year
Foreign Air Mail add $12.
NEW S-100 ROM-RAM-I/O BOARD AVAILABLE

ELECTRONIC CONTROL TECHNOLOGY, 763 Ramsey Ave, Hillside NJ 07205 has available the model R2I/O board, S-100 Computer Systems. It features 3 serial I/O ports, 1 parallel I/O port, 4 status ports, 2K of ROM and 2K of RAM. The R2I/O provides a means of interfacing several I/O peripherals such as CRT terminals, line printers, modems or other devices, to an S-100 Computer.

It also provides computer control from a terminal with a ROM-based Monitor program containing Executive commands and I/O routines. It can also be used in dedicated control applications with a minimum number of boards since it contains ROM, RAM and I/O. Baud rates are individually selectable from 75-9600 baud and serial I/O ports feature RS-232 compatibility. All ICs are socketed and the board is available for $295 assembled and tested.

HIGH SPEED FOUR CHANNEL 12-BIT DAC BOARD

A high speed four channel 12-bit Digital-to-Analog Converter board has been introduced by TECMAR INC., 23414 Greenlawn Avenue, Cleveland Ohio 44122. The board features four completely independent high speed DACs and associated latches. The DACs have a typical conversion time of 3 usecs. The use of latch driven inputs allows the DAC to hold its previous input until an entirely new word is presented to it. The board features user-selectable output voltage ranges 0 to +5V, 0 to +10V, +2.5V, +5V, +10V. The board may be addressed as I/O ports or memory mapped. It comes assembled, tested and completely socketed and is $395. The technical manual alone is $15.

fcc approved s-100 modem board

POTOMAC MICRO-MAGIC INC., First Lincolnia Building, Suite B1, 4810 Beauregard Street, Alexandria, Va 22312 had introduced an S-100 compatible modem board. The MM-103, allows commercial quality communications at an affordable price. The board utilizes sophisticated modem IC and associated circuitry. Electronic features of the MM-103 include: positive carrier detection and loss of carrier disconnect, 5MHz operation without wait states, variable baud rate of over 200 values for 61 to 600 baud, 50 db sensitivity, 10 DB S/N and allows for speed dialing when available from phone company. The board features the most extensive software support of any data communications interface for S-100 Computer Systems. Included is the source code for over 30 programs as well as diskettes for North Star DOS, North Star CP/M, 8" CP/M, and Micropolis II. The MM-103 is the first S-100 Modem approved by the FCC for direct connection to the phone system without the use of DAA device (CBS or CBT). The modem is available assembled and tested for $395.
THE DATABASE SPECIALIST

HDBS - HIERARCHICAL DATA BASE MANAGEMENT SYSTEM
MDBS - OUR FULL NETWORK DATA BASE MANAGEMENT SYSTEM

<table>
<thead>
<tr>
<th>HDBS FEATURES</th>
<th>ADDITIONAL FEATURES IN MDBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>• HIERARCHICAL DATA STRUCTURES</td>
<td>• HIERARCHICAL AND FULL NETWORK DATA STRUCTURES (CUBESYL ORIENTED)</td>
</tr>
<tr>
<td>• FIXED LENGTH RECORDS</td>
<td>• FIXED AND VARIABLE LENGTH RECORDS</td>
</tr>
<tr>
<td>• READ/WRITE PROTECTION AT FILE LEVEL</td>
<td>• MULTIPLE LEVELS OF READ/WRITE PROTECTION AT ITEM, RECORD, SET AND FILE LEVELS</td>
</tr>
<tr>
<td>• ONE-TO-MANY SET RELATIONSHIPS ALLOWED</td>
<td>• EXPLICIT REPRESENTATION OF ONE-TO-ONE, ONE-TO-MANY, MANY-TO-ONE AND MANY-TO-MANY SETS</td>
</tr>
<tr>
<td></td>
<td>• RECORD TYPES MAY OWN OTHER OCCURRENCES OF THE SAME RECORD TYPE</td>
</tr>
<tr>
<td></td>
<td>• A SINGLE SET MAY HAVE MULTIPLE OWNER AND MEMBER RECORD TYPES</td>
</tr>
</tbody>
</table>

FEATURES COMMON TO HDBS and MDBS

• STRAIGHT FORWARD USE OF ISAM-LIKE STRUCTURES
• SORTED, FIFO, LIFO, NEXT AND PRIOR SET ORDERING PROVIDED
• COMMANDS TO ADD, DELETE, UPDATE, SEARCH AND TRAVERSE THE DATA BASE
• NAMES OF DATA ITEMS, RECORDS, SETS AND FILES ARE WHOLLY USER DEFINABLE
• RECORDS CAN BE MAINTAINED IN A NUMBER OF SORTED ORDERS
• ROUTINES ARE CALLABLE FROM BASIC, PASCAL, FORTRAN, COBOL OR MACHINE LANGUAGE
• WRITTEN IN MACHINE LANGUAGE FOR MAXIMAL EXECUTION EFFICIENCY AND MINIMAL MEMORY USAGE
• SUPPORTS DATA BASE SPREAD OVER SEVERAL DISK DRIVES (MAX. 8 DISKS MAY BE MINI OR FULL SIZED FLOPPIES OR HARD DISKS

Also

• UP TO 256 RECORD-TYPES MAY BE DEFINED IN THE DATA BASE
• EACH RECORD-TYPE MAY CONTAIN UP TO 256 ITEM-TYPES
• EACH ITEM-TYPE MAY BE UP TO 9,999 BYTES IN LENGTH

REQUIREMENTS

• Z-80 APPROXIMATELY 16K MEMORY –
• 8080 APPROXIMATELY 20K MEMORY –
• IN ADDITION TO THE OPERATING SYSTEM, HOST LANGUAGE,
  USER'S PROGRAM AND SOME BUFFER AREA
• 6502 APPROXIMATELY 26K MEMORY

HDBS and MDBS PACKAGES INCLUDE

DDL DATA DEFINITION LANGUAGE ANALYZER/EDITOR. The user specifies data structures to be used in a Concise Data Definition Language (DDL). The Data Definition Language Analyzer/Editor allows the user to interactively create and edit DDL Specifications and to initialize the data base for use based on these specifications.

260 PAGE USERS MANUAL with extensive documentation of the Data Base Management System.

MDBS DATA MANAGEMENT ROUTINES. These are the routines callable from the host language (BASIC, PASCAL, etc.) which perform the data base operations of finding, adding, and deleting records; fetching and storing data items; and traversing the (possibly complex) data structure.

SAMPLE APPLICATION PROGRAM AND DDL FILES

RELOCATOR TO RE-ORG ALL ROUTINES

SYSTEM SPECIFIC MANUAL to show how to bring up our software on your computer

NEW DYNAMIC RESTRUCTURING SYSTEM

MDBS-DRS FEATURES

• ALLOWS ITEM, RECORD AND/OR SET TYPES TO BE ADDED TO OR DELETED FROM AN EXISTING DATA BASE. THIS ALLOWS THE USER TO RE-DESIGN A DATA BASE AFTER IT IS ALREADY ON-LINE.

THIS FEATURE CAN ONLY BE ADDED TO THE MDBS SYSTEM.

RUNS UNDER

• CP/M WITH MICROSOFT BASIC, COBOL AND FORTRAN
• NORTHSTAR DOS WITH NORTHSTAR BASIC
• TRSDOS* AND DISK BASIC
• APPLE* DOS AND APPLE* SOFT BASIC
• MACHINE LANGUAGE CALLABLE FORMS
• OTHER FORMS IN DEVELOPMENT

MICRO DATA BASE SYSTEMS, INC
P.O. BOX 248 LAFAYETTE, IN 47902
(317) 742-7388

CP/M is a registered trademark of Digital Research Corp.
TRS-80 is a registered trademark of Radio Shack/Tandy Corp.
Apple is a registered trademark of Apple Computers, Inc.
COMPUPRO™ CONTINUES TO DELIVER.

Whether it's memory, motherboards, I/O boards, enclosures, or any of our family of products, COMPUPRO delivers what you want at prices you can afford.

Looking for memory? Our boards are fully static, low power, run at 4 or 5 MHz, support a number of popular busses, include a 1 year limited warranty, and generally come in 3 configurations to suit your exact needs. For lowest cost, choose an "unkit" with sockets and bypass caps pre-soldered in place for easy assembly. When you can't wait to get going, order one of our assembled versions. For critical systems, specify boards qualified under our Certified System Component (CSC) high reliability program. These boards are extensively tested, burned in for at least 200 hours, and are immediately replaced in event of failure within 1 year of invoice date.

Looking for other peripherals? We mix leading edge technology, design savvy, and volume buying to deliver the right product at the right price. See our list below for pricing.

Memory name | Bus & Notes | Unkit | Assm | CSC
---|---|---|---|---
8K Econoram* | S-100 | $149 | $179 | $239
16K Econoram | S-100 | $289 | $339 | $429
16K Econoram VIIA-16 | S-100 | $329 | $349 | $439
24K Econoram VIIA-24 | S-100 | $419 | $499 | $605
16K Econoram IX-16 | Dig Grp | $319 | $379 | n/a
32K Econoram IX-32 | Dig Grp | $559 | $639 | n/a
32K Econoram X | S-100 | $549 | $669 | $789
32K Econoram XI | SBC/BLC | n/a | n/a | $1050
16K Econoram XIIIA-16 | S-100 (1) | $349 | $419 | $519
24K Econoram XIIIA-24 | S-100 (1) | $469 | $539 | $649
32K Econoram XIIIA-32 | S-100 (1) | $579 | $699 | $849
16K Econoram XIV | S-100 (2) | $299 | $359 | $459
16K Econoram XV-16 | H8 (3) | $329 | $395 | n/a
32K Econoram XV-32 | H8 (3) | $599 | $729 | n/a

*Econoram is a trademark of Godbout Electronics.

(1) Compatible with all bank select systems (Cromemco, Alpha Micro, etc.): addressable on 4K boundaries.
(2) Extended addressing (24 address lines), Single block addressable on 4K boundaries.
(3) Bank select option for implementing memory systems greater than 64K.

S-100 MEMORY MANAGER BOARD

$59 unkit
$85 assm
$100 CSC

Add bank select and extended addressing to older S-100 machines (Altair, IMSAI, Solo, and others). Use with our new extended addressing boards, or retrofit our high density Econorams for use with the Memory Manager to get added memory space for your computer.

MULLEN S-100 I/O BOARD

$49 kit

Includes logic probe and general purpose breadboard section, ideal for troubleshooting and analysis.

3P PLUS S "Interfacer II"

$199 unkit
$249 assm
$324 CSC

Incorporates 1 channel of serial I/O (RS-232 with full handshake), along with 3 full duplex parallel ports plus a separate status port. The parallel section uses Tri-State (tm National Semiconductor) octal latches for latched data, input and output with 24 mA drive current, attention/enable/strobe bits for each parallel port, and separate connectors for power for each channel.

2S "Interfacer I"

$199 unkit
$249 assm
$324 CSC

Dual RS-232 ports with full handshake; use EIA232C line drivers and receivers. Supports both 150 and 9600 baud, and provides diagnostics on errors, parity and signal quality.

TERMS: Cal res add tax. Allow 5% for shipping excess refunded. VISA/Mastercharge calls our 24 hour order desk at (415) 522-0636. COD OK with street address for UPS. Prices good through cover month of magazine.