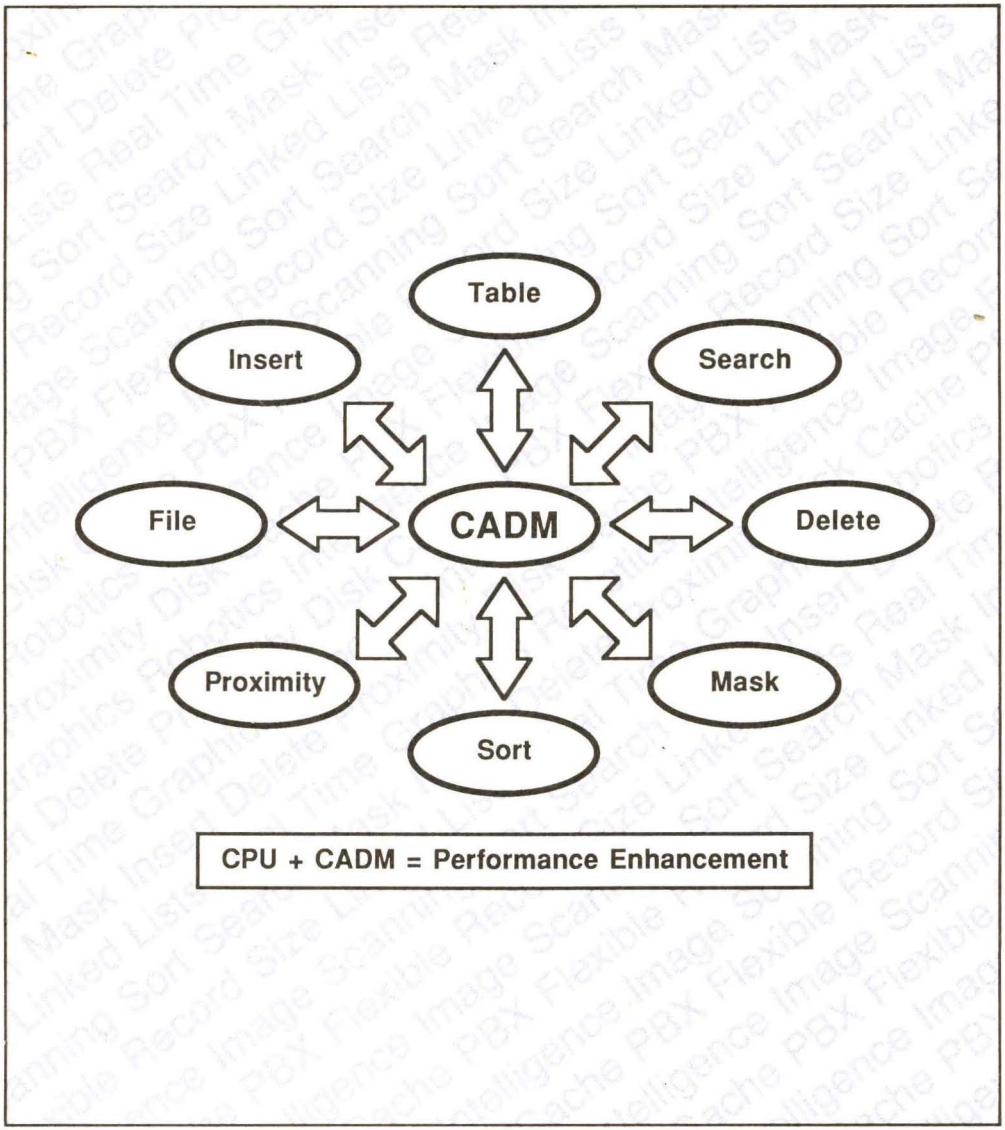
Content Addressable Data Manager Am95C85

Technical Manual



ADVANCED MICRO DEVICES

Advanced Micro Devices

Am95C85 (CADM) Content Addressable Data Manager

Technical Manual

901 Thompson Place, P.O. Box 3453, Sunnyvale, California 94088
(408) 732-2400 TWX: 910-339-9280 TELEX: 34-6306

ACKNOWLEDGEMENTS:

This technical manual was written by Sarosh Vesuna, Headquarters Applications Engineer. The Senior Technical Writer is Erland Kyllonen.

Contributions and assistance were provided by Dave Horton, Product Planning Manager, Rob Oliver, Senior Product Marketing Engineer, and Joseph Brcich, Headquarters Applications Manager.

TABLE OF CONTENTS

1. **INTRODUCTION** ... 1-1

1.1 Overview ... 1-1
1.2 Distinctive Characteristics ... 1-1
1.3 The Hardware Solution ... 1-1
1.4 Applications ... 1-2

2. **FUNCTIONAL DESCRIPTION** ... 2-1

2.1 General Description ... 2-1
2.2 Address Space ... 2-1
2.2.1 Variable-Width Record ... 2-1
2.2.2 The Masking Option ... 2-1
2.2.3 Input Buffer Space ... 2-2
2.2.4 Remaining Space ... 2-3
2.3 Addressing Modes ... 2-3
2.3.1 Content Addressable Array ... 2-3
2.3.2 Auto-Increment Mode ... 2-3
2.3.3 Stack Access Mode ... 2-3
2.4 Sorting ... 2-4
2.5 Cascading Multiple CADMs ... 2-4
2.5.1 Cascading Up To 16 Am95C85s ... 2-4
2.5.2 Cascading More Than 16 Am95C85s ... 2-4
2.6 Pin Description ... 2-6
2.6.1 Data Bus ... 2-6
D0-D7 Data Bus (Input/Output, 3-state) ... 2-6
2.6.2 Interface Control ... 2-7
$\overline{RST}$ Reset (Input, Active LOW) ... 2-7
$\overline{CS}$ Chip Select (Input, Active LOW) ... 2-7
$\overline{RE}$ Read Enable (Input, Active LOW) ... 2-7
$\overline{WE}$ Write Enable (Input, Active LOW) ... 2-7
C/$\overline{D}$ Command/$\overline{Data}$ (Input) ... 2-7
CLK Clock (Input) ... 2-8
$\overline{T}$/R $\overline{Transmit}$/Receive (Output) ... 2-8
$\overline{DONE}$ Done (Input/Output, Active LOW, 3-state) ... 2-8
$\overline{STAT}$ Status (Output, Active LOW, 3-state) ... 2-9
2.6.3 Chip to Chip Communication ... 2-9
TUP Transmit Up (Output, Active HIGH) ... 2-9
TDWN Transmit Down (Output, Active HIGH) ... 2-9
RUP Receive from the Up Direction (Input, Active HIGH) ... 2-9
RDWN Receive from the Down Direction (Input, Active HIGH) ... 2-9
$\overline{GLB}$ Global (Input/Output, 3-state) ... 2-9
$\overline{DIRG}$ Direction of GLB Signal (Output, Active LOW, Open Dr.) ... 2-9
$\overline{DIRD}$ Direction of DONE Signal (Output, Active LOW, Open Dr.) ... 2-9
2.6.4 Supply Pins ... 2-9
Vcc Power Supply ... 2-9
GND Ground ... 2-9

3. Am95C85 INSTRUCTION SET 3-1

3.1 Initialization Instructions 3-1
3.2 Byte-oriented Instructions 3-1
3.3 Record-oriented Instructions 3-1
3.4 Instruction Set 3-1
AIM Auto Increment Mode 3-2
DEC Decrement Address Pointer 3-3
FND Find a Matching Key 3-4
GSF Get Status Full 3-5
KPL Load Length of Key Field, Length of Pointer Field and Last Address Pointer 3-6
LAL Load Address Long 3-7
LAS Load Address Short 3-8
LUD Load Unsorted Data 3-9
NXT Point to Next Record 3-10
PRE Point to Previous Record 3-11
RRB Restore Record Boundary 3-12
RST Reset 3-13
SMB Set Mask Bytes 3-14
SOF Sort Off Line 3-15
SON Sort On Line 3-16
STK Stack Access Mode 3-17

4. PROGRAMMING THE Am95C85 4-1

4.1 Required Software Command Sequences 4-1
4.1.1 Typical Initialization Sequence 4-1
4.1.2 Sorting Off-line 4-1
4.1.3 Search for a Matching Key 4-2
4.1.4 Record-oriented Data Access 4-2
4.2 Command Sequences to be Avoided 4-3
4.3 Byte Boundary to Bit Boundary Conversion 4-4
4.4 Data Manipulation 4-4
4.5 Helpful Hints 4-5
4.5.1 Using the LUD Command 4-5
4.5.2 Keep the Pointer Within Meaningful Data 4-5
4.5.3 Last Address Too High 4-5
4.5.4 Using $\overline{\text{STAT}}$ in Polled Mode 4-5
4.5.5 SMB Declares CADM Data Unsorted 4-5

5. INTERFACE CIRCUIT 5-1

5.1 Introduction 5-1
5.2 DMA Transfer Mode 5-1
5.3 CADM Clock 5-1
5.4 System Bus to CADM Bus Isolation 5-2
5.5 Local CADM Data Bus Bank-to-bank Isolation 5-2
5.6 CADM Status Output 5-2
5.7 Local CADM Signal Buffering 5-2
5.8 CADM Command/$\overline{\text{Data}}$ Select 5-6
5.9 Forcing READY Active 5-6

5.10 The Am95C85 (CADM) Interface to an IBM PC/XT/AT ... 5-7
5.10.1 Synchronizing the Read and Write Signals 5-7
5.10.2 Chip Select Logic 5-7
5.10.3 Generating the Ready Signal 5-10
5.10.4 PAL Device Implementation of the Interface 5-10
5.11 Am95C85 (CADM) Interface to an 8086 Processor ... 5-10
5.11.1 Synchronizing the Read and Write Signals 5-10
5.11.2 Chip Select Logic 5-12
5.11.3 Generating the READY Signal 5-16
5.12 Am95C85 (CADM) Interface to an MC68000 Processor ... 5-16
5.12.1 Synchronizing the Read and Write Signals 5-16
5.12.2 Chip Select Logic 5-20
5.12.3 Generating the READY Signal 5-20

APPENDIX A

Am95C85 CADM SORT PERFORMANCE BENCHMARK SUMMARY A-1

Benchmark Summary A-1
Benchmark Description A-1
Methodology A-3
Input Files A-4
Calculating Sort Times A-4
Maintaining Data Accuracy A-4
System Clock Granularity A-4
Multi-User Systems A-4
Summary A-4
CADM Sort Times vs. Standard Computers A-5

LIST OF ILLUSTRATIONS

Figure 1-1. Am95C85 CADM Block Diagram 1-1
Figure 1-2. Typical System Configuration 1-2
Figure 1-3. Indexed File Using CADM 1-2

Figure 2-1. Am95C85 CADM Address Space 2-1
Figure 2-2. Content-Addressable Array Operations 2-2
Figure 2-3. Auto-Increment Mode 2-3
Figure 2-4. Stack Access Mode 2-4
Figure 2-5. Cascading Up To 16 CADM Devices 2-5
Figure 2-6. Cascading 256 CADM Devices 2-6
Figure 2-7. Buffering Banks of CADMs 2-7
Figure 2-8. Am95C85 Block Diagram 2-8

Figure 4-1. Initialization Sequence 4-1
Figure 4-2. Simplified Off-Line Sort Sequence 4-2
Figure 4-3. Record Search Sequence 4-3
Figure 4-4. Boundary Conversion Example 4-4

Figure 5-1. Unsymmetrical CADM Clock Logic 5-1

Figure 5-2. Cascading Up To 16 CADM Devices 5-3
Figure 5-3. Cascading More Than 16 CADM Devices 5-4
Figure 5-4. Am95C85–IBM PC/XT/AT Interface 5-6
Figure 5-5. 74LS125 Logic Diagram 5-5
Figure 5-6. Am95C85–IBM PC/XT/AT Interface Write Timing 5-8
Figure 5-7. Am95C85–IBM PC/XT/AT Interface Read Timing 5-9
Figure 5-8. Am95C85–IBM PC/XT/AT Interface Using AmPAL16R4A ... 5-11
Figure 5-9. PAL Device Equations for CADM–IBM Interface Ready Circuit ... 5-12
Figure 5-10. Am95C85–8086 Interface 5-13
Figure 5-11. Am95C85–8086 Interface Write Timing 5-14
Figure 5-12. Am95C85–8086 Interface Read Timing 5-15
Figure 5-13. Am95C85–68000 Interface 5-17
Figure 5-14. Am95C85–68000 Interface Write Timing 5-18
Figure 5-15. Am95C85–8086 Interface Read Timing 5-19

Figure A-1. Sort Performance–CADM vs. Standard Computers A-2

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The Am95C85, Content Addressable Data Manager (CADM), is a microprocessor peripheral device capable of both, storing and managing data, thus relieving the host CPU of many time-consuming data manipulation and management tasks. The CADM can perform many sorting and searching operations significantly faster than applications software.

Any computer-based system spends a significant amount of its time performing repetitive tasks associated with data management. As an example, the graphics workstation typically spends a major portion of its CPU time searching and updating virtual memory tables, graphics vector lists, task tables, and directories. A Content Addressable Data Manager can perform the time-consuming details involved in these tasks thus freeing the CPU for other functions and increasing overall system performance. (Refer to the Appendix for CADM Benchmark summary.)

The Am95C85 combines the advantages of a CAM (Content Addressable Memory) with the flexibility of a RAM. It eliminates the need to provide physical addresses to access its memory. It provides automatic record manipulation for operations such as tabular search, index file updates, list sorts, and other iterative tasks. It provides programmable record width and several modes of physical addressing. In addition, an auto-increment mode allows a sequence of reads or writes from consecutive memory locations. A Stack Access Mode allows the insertion or removal of a record at any location without the need for resorting.

1.2 DISTINCTIVE CHARACTERISTICS

Some of the prominent features of the Am95C85, Content Addressable Data Manager (CADM), are:

- On-chip intelligence controls host-independent processing and manipulation
- 1 kbyte of on-chip RAM
- Cascadable to 256 K RAM
- A software programmable field width provides flexibility in managing different data types
- High-performance sorting and searching operations done by hardware without CPU involvement
- CAM (Content Addressable Memory) mode accelerates the searching process
- Stack mode allows the insertion and deletion of a record at any location in the CADM memory
- A short, powerful, yet simple instruction set provides versatility to the user
- Manufactured in low power CMOS technology

1.3 THE HARDWARE SOLUTION

To speed up the sort process, software is replaced by hardware in the Am95C85, Content Addressable Data Manager (CADM). A block diagram of the CADM is shown in Figure 1-1.

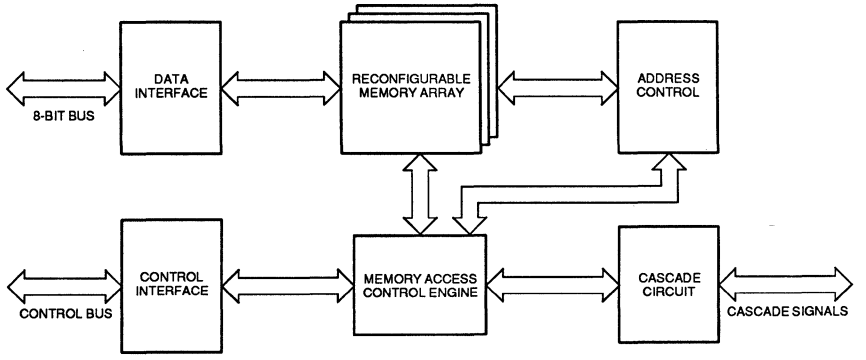


Figure 1-1. Am95C85 CADM Block Diagram

The Am95C85 is capable of data storage and management without CPU intervention. Some of the sort and search operations are orders of magnitude faster when performed by the Am95C85 as compared to the same operations implemented by applications software. Refer to the CADM (Am95C85) Sort Performance Benchmark for details.

1.4 APPLICATIONS

The spectrum of applications which can benefit from the high performance of the CADM include:

- Data base management
- Real-time graphics systems
- Multi-tasking systems
- Robotics
- Artificial intelligence
- Networking and data communications
- Disk and file server systems
- Image scanning devices
- Data acquisition-(Radar)

Virtually any system or sub-system requiring high-speed data structuring and manipulation can be significantly improved by using the CADM.

Figure 1-2 shows a typical system configuration using CADMs. Figure 1-3 shows the relationship of a CADM to main memory. The key fields (keywords) are sorted in the CADM whereas the data records may be randomly located in main memory.

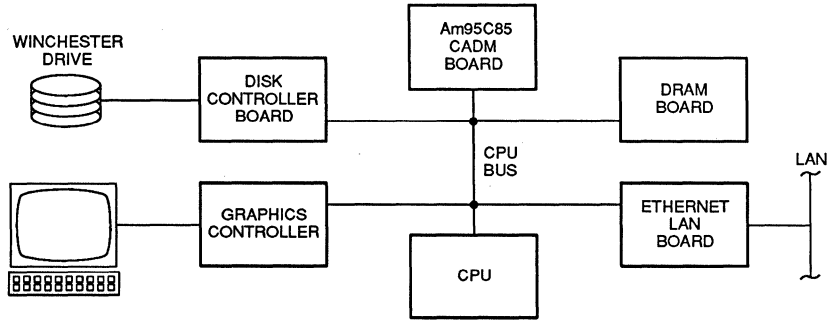


Figure 1-2. Typical System Configuration

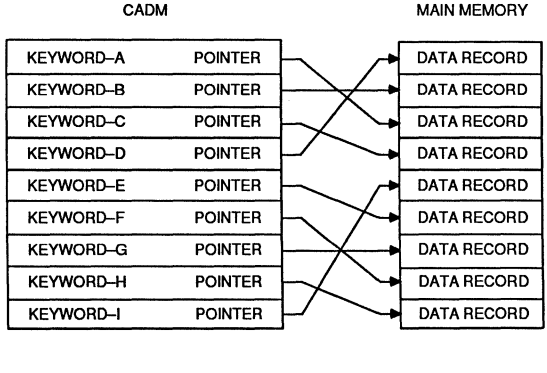


Figure 1-3. Indexed File Using CADM

CHAPTER 2

FUNCTIONAL DESCRIPTION

2.1 GENERAL DESCRIPTION

This chapter describes the Am95C85 and discusses the functional relationships of its control signals. It describes setting the record length, the masking option for the key field, cascading multiple CADMs, the sorting capability, and the addressing modes. The memory is content-addressable. In addition, an auto-increment mode allows a sequence of reads or writes from consecutive memory locations. A Stack Access Mode allows the insertion or removal of a record at any location without the need for resorting. Examples are given to aid in understanding the concepts. This chapter also contains the pin descriptions.

2.2 ADDRESS SPACE

The address space of the CADM consists of a mask area, a record area, and an input buffer area. These areas are shown in Figure 2-1. If the masking option is chosen, the first 'k' bytes of each CADM are reserved for the mask. If the masking option is not chosen, this space is included in the usable record space.

2.2.1 VARIABLE-WIDTH RECORD

The CADM's data management scheme was designed for flexibility in accommodating many types of files. The device's unique internal RAM has an adaptive design that allows the record width to be selected by the user to meet the specific demands of the application.

Each record consists of a key field and an optional pointer field (Figure 2-1). The key field may have from 1 to 255 bytes and the pointer field may have from 0 to 255 bytes. The width of a record can be varied between 1 and 510 bytes, thus providing the versatility to handle a wide range of file types and record sizes.

2.2.2 THE MASKING OPTION

Bits in the key field may be selectively masked by the user before a sort or search operation. When the mask option is used, it must be programmed before data is loaded into the CADM. This is necessary because the Am95C85 allocates the first 'k' bytes of each device to accommodate

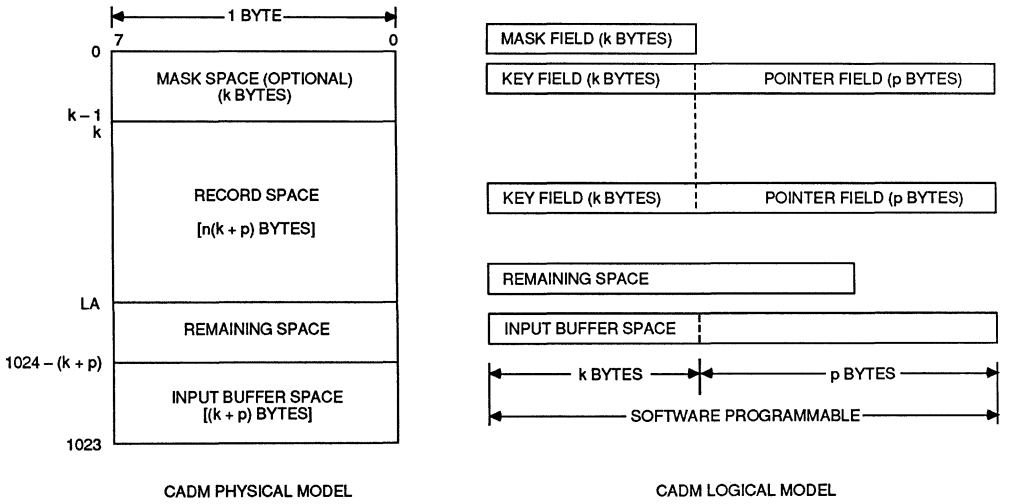


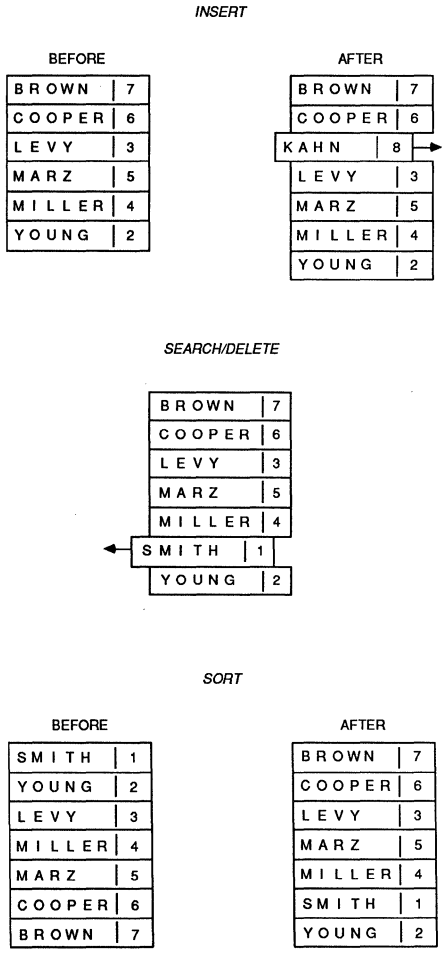
Figure 2-1. Am95C85 CADM Address Space

the mask bytes if selected (where 'k' is the length of the key field). A new masking bit pattern may then be chosen anytime during normal operation. For example, masking bit 5 in each key byte allows for sorting or searching of ASCII text without regard to upper and lower case characters.

If more than one CADM is connected in cascade, the mask space is duplicated in each CADM even though the same mask is used for all of the records.

2.2.3 INPUT BUFFER SPACE

The last (k + p) bytes of each CADM are designated as an Input Buffer Space. These memory locations, equivalent to one record space,



NOTE: IN THIS EXAMPLE, THE KEY FIELD HAS BEEN SPECIFIED AS 6 BYTES IN LENGTH. ALL RECORDS MUST BE LOADED WITH KEYS LEFT JUSTIFIED WITH SPACES FILLED IN TO THE RIGHT. IN THIS EXAMPLE, INFORMATION IS SORTED ALPHABETICALLY.

08035A 2-2

Figure 2-2. Content-Addressable Array Operations

are temporarily used to store a record. After an entire record has been loaded into the input buffer space, one byte at a time, data manipulation on this record begins (e.g., sort, find, insert, etc.).

2.2.4 REMAINING SPACE

Each CADM device has 1024 bytes of memory as mentioned earlier. The first 'k' bytes of this memory space are used to store the mask bytes if masking is desired. The last (k + p) bytes of each CADM are reserved as input buffer space. All other memory space is available to the user to load records for sort and search operations. Unless this record space is an integral multiple of the record length, some record space is left over. Hence, a few bytes are unusable (always less than the length of a record field) in each CADM. These bytes, designated as the remaining space, are located between the last address location and the beginning of the input buffer space (Figure 2-1). A simple formula for calculating the Last Address location is:

- if mask bytes are used,
  $LA = \{INT[(1024-2 \cdot k-p) / (k+p)]\} \cdot (k+p) + k - 1$
- if masking is not used,
  $LA = \{INT[(1024-k-p)/(k+p)]\} \cdot (k+p) - 1$

where,

LA → is the Last Address location (byte) in each chip which can contain meaningful user data

k → is the length of the key field in bytes

p → is the length of the pointer field in bytes

2.3 ADDRESSING MODES

The CADM maintains all the pointers necessary to manage the following three modes of data access: Content Addressing, Auto Increment, and Stack Access. Only one of these pointers is relevant to the user, that being the one to read and write data, the Address Pointer. The user may write an address into this pointer. This feature is provided for diagnostics and testability.

2.3.1 CONTENT ADDRESSABLE ARRAY

As a content-addressable device, the CADM searches the memory array to find a record whose key value matches a particular key designated by the user. If a matching key is found, the Address Pointer contains the address of the first byte of the record which returned the match (Figure 2-2). Multiple matches to a key value are located during subsequent Find operations. If the desired key is not found in the array, the Status line is pulled LOW indicating a 'no match'. The Address Pointer will then contain the address of the first byte of the record with the next higher value key. (This is consistent with the Stack Address mode of data insertion used to place new data in the array.)

Once the address of a required record is determined using the above scheme, data (i.e., key and pointer values) may be read from, or written to the Am95C85 devices. When multiple CADMs are cascaded, the Search works in parallel on all devices. Thus, the performance of the Find operation is independent of file size if at least one CADM is filled up.

2.3.2 AUTO-INCREMENT MODE

The auto-increment mode allows the host to select any particular address location and read or write data at that location. Subsequent reads and writes are easy to execute since the device auto-increments the Address Pointer after each data access. When writing data to a location, any previous data at that location is lost when in the Auto-increment Mode. Refer to Figure 2-3. This facilitates loading and unloading the CADM with DMA.

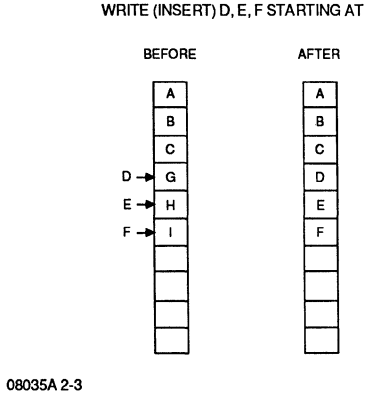


Figure 2-3. Auto-Increment Mode

2.3.3 STACK ACCESS MODE

Operating in the stack access mode allows for immediate insertion or deletion of records. In a previously sorted data array, a record can be inserted or deleted without the need for resorting.

The pointer is set by executing a FND (FIND) instruction, or loading the pointer with an LAS or LAL.

In this mode, the device will insert or delete a record in the array by physically moving all data below the record addressed. A data read 'pops' a byte at the Address Pointer location, moving data below it in the upward direction. Conversely, a data write 'pushes' a byte on the array at the Address Pointer, moving all the data below the pointer downward to make room for the insertion (Figure 2-4). This quick updating of a data base without having to re-sort the entire array delivers amazing performance improvement over traditional software implementations. The CADM can be accessed by DMA.

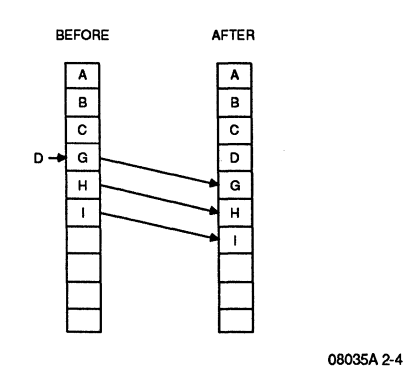


Figure 2-4. Stack Access Mode

2.4 SORTING

Each record in the CADM consists of a key and a pointer. The key may be just one byte or up to 255 bytes in length. The pointer field may vary between 0 and 255 bytes. Data entered into the CADM is sorted by performing a binary search/insert type sort. The user may choose between the On-Line Sort, where data is sorted record by record as it is loaded into the CADM, or an Off-Line Sort, where the host is allowed to quickly load an entire block of unsorted data into the CADM for sorting at a later point in time.

Off-line sorting allows the CPU to perform other tasks while the sorting is taking place. This on-chip intelligence of the Am95C85 is of particular advantage in multiprocessing systems where reducing CPU overhead can significantly increase system performance.

2.5 CASCADING MULTIPLE CADMs

The address space is physically partitioned into several sections to facilitate internal operations of the Am95C85. This address space is expandable up to 256 kbytes by cascading multiple CADM devices. A maximum of 256 devices may be linked together (Figures 2-5, 2-6, and 2-7) so that, from a programmer's perspective, the memory space resembles a single continuous memory block.

The architecture of the Am95C85 is ideal for linking multiple devices in cascade. Four signals, Transmit Up, Transmit Down, Receive Up, and Receive Down, provide the inter-chip control and communication required to successfully complete operations on multi-chip arrays (Figure 2-5).

2.5.1 CASCADING UP TO 16 Am95C85s

All CADM devices in cascade share a common data bus. In addition there are a few control signals that connect to all CADM devices. In order to interface the data bus and these signals to the host system, some form of buffering must be used to isolate the local CADM data bus and control signals from the host. This is necessary so that transactions between CADMs during an off-line operation do not interfere with operations performed by the CPU during the same time period. One of the methods to accomplish this is shown in Figure 2-5.

2.5.2 CASCADING MORE THAN 16 Am95C85s

When cascading more than 16 CADMs, two levels of data buffering are required. In addition some of the control signals that interconnect all CADM devices also need to be buffered. This is necessary because the Am95C85 outputs can drive a maximum of 200 pF capacitive load at the rated maximum frequency (If the capacitive load is larger than 200 pF, the CADMs will work properly but the clock may have to be slowed down). Each CADM has an input capacitance of the order of 10 pF. Taking the bus loading capactance and other stray capacitance into consideration, each CADM is capable of driving about 16 CADMs. A buffering scheme to separate banks of CADMs is shown in Figures 2-6 and 2-7. The CADM device identification register is eight bits wide. This enables a system to have 256 cascaded CADMs (16 banks of CADMs with 16 devices per bank).

Banks of 16 CADMs are isolated from each other and from the host system by control signal buffers and data buffers (Figures 2-6 and 2-7). The

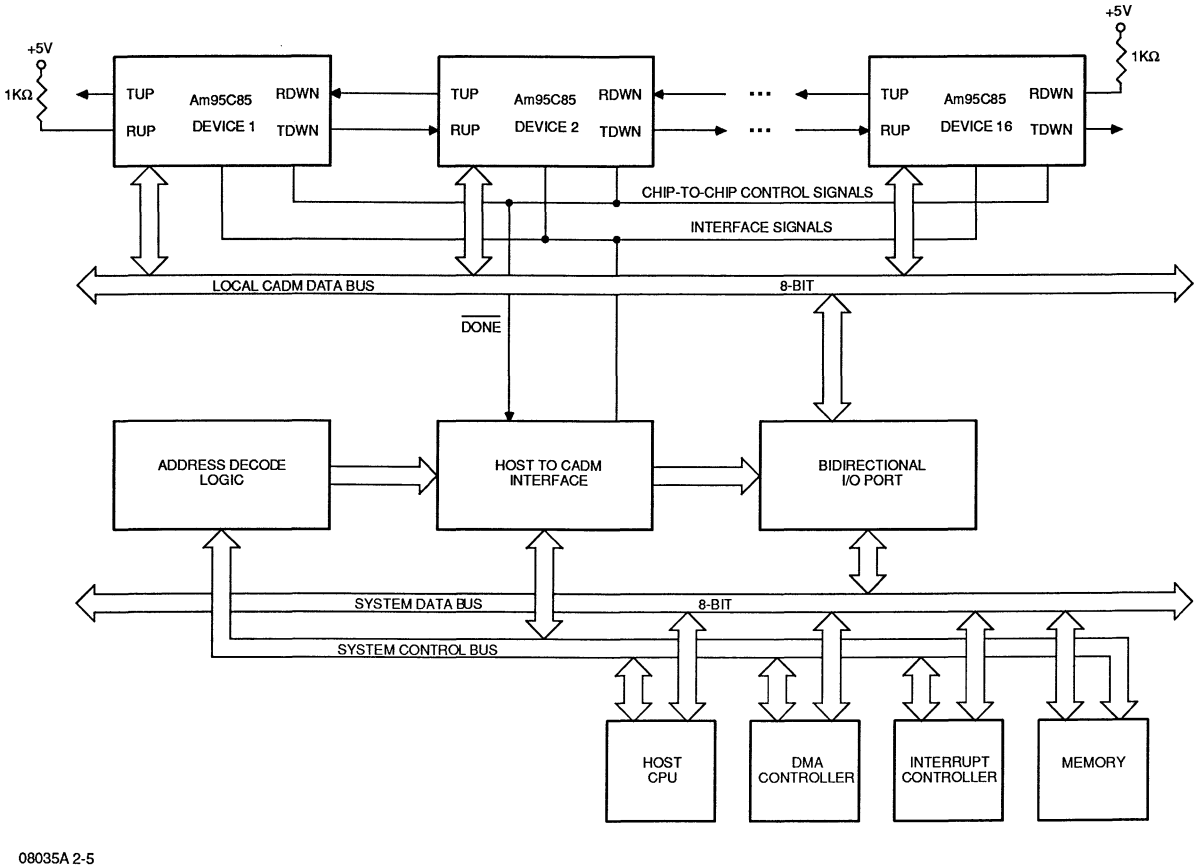


Figure 2-5. Cascading Up To 16 CADM Devices

Am2959 octal buffers isolate the Read, Write, Chip Enable and Command/Data signals. The 74LS125 buffers are connected as shown in Figure 2-7 to control the direction of the Global and Done signals. An eight-input NAND gate combines $\overline{STAT}$ (Status Signal) from up to eight banks of CADMs for an interrupt request to the Am9519A interrupt controller.

2.6 PIN DESCRIPTION

The signal names and the block diagram of the Am95C85 are shown in Figure 2-8.

2.6.1 DATA BUS

D0–D7 Data bus (Input/Output, 3-state)

The eight bidirectional data pins are connected to all Am95C85 devices. These lines are used for information exchanges between Am95C85 CADM devices and the host processor, and between CADM devices themselves. Because the same data pins are used for system interaction and CADM interaction, a transceiver must isolate the CADM array from the system data bus. The pins carry data or command information to and from the Am95C85 devices. A HIGH on a data line

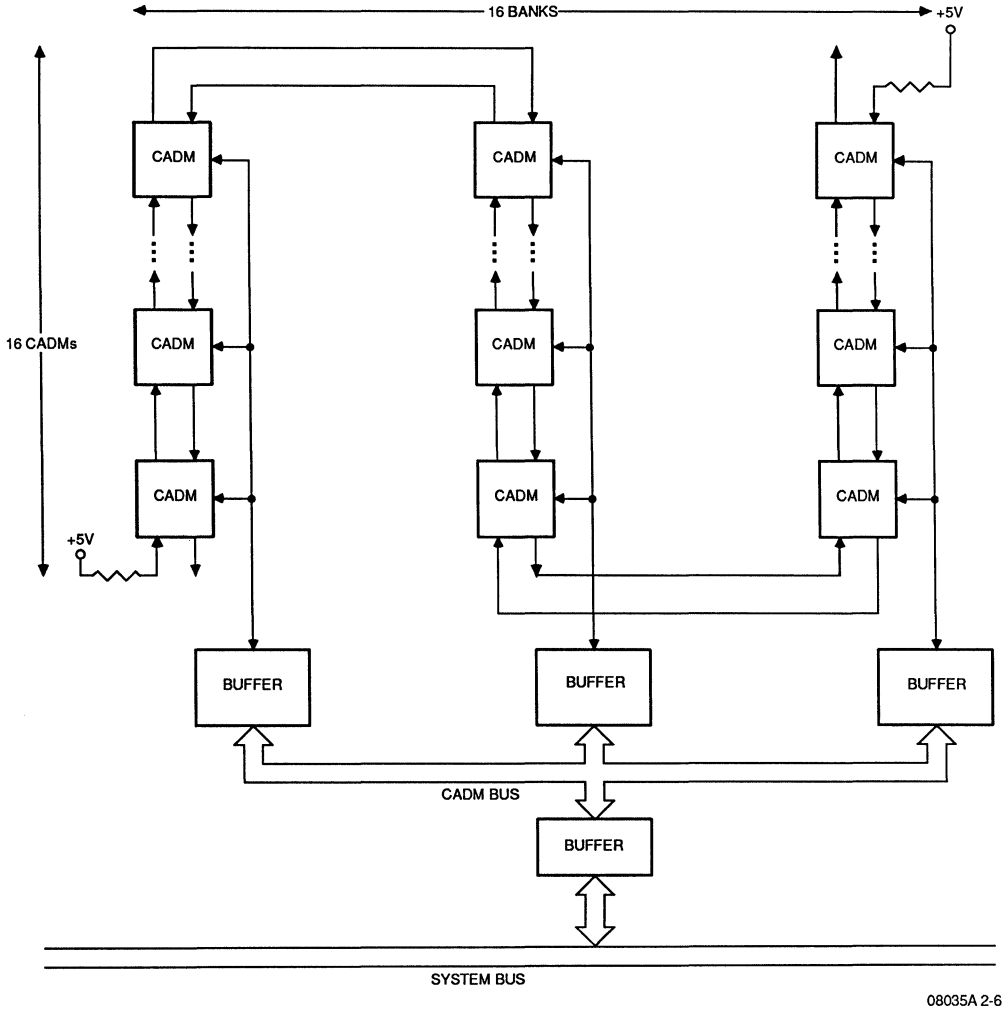


Figure 2-6. Cascading 256 CADM Devices

corresponds to a logic '1' and a LOW corresponds to a logic '0'. These lines act as inputs when $\overline{WE}$ and $\overline{CS}$ are active, and as outputs when $\overline{RE}$ and $\overline{CS}$ are active. D0 is the least significant and D7 the most significant bit position.

2.6.2 INTERFACE CONTROL

The following control signals interface the CADMs to the Host processor.

$\overline{RST}$ Reset (Input, Active LOW)

A chip reset is initiated by pulling this pin LOW for a minimum of four CADM clock cycles. Any command under execution is terminated. $\overline{DONE}$ goes HIGH for the duration of the internal reset operation. Masking is disabled. The Am95C85 device with RUP tied HIGH assumes it has a chip address of 0, the next chip assumes an address of 1, and so on, until all devices enumerate themselves. The device with its RDOWN tied HIGH is the last device in the cascade. The wire-ORed $\overline{DONE}$ pin signals completion of the reset cycle by going LOW. (A software reset ($\overline{RST}$) has the effect of activating the $\overline{RST}$ pin.) A hardware reset is recommended on power-up.

$\overline{CS}$ Chip Select (Input, Active LOW)

The chip select input enables the host CPU to perform read and write operations with the Am95C85 devices. When chip select is HIGH, the read and write inputs are ignored.

$\overline{RE}$ Read Enable (Input, Active LOW)

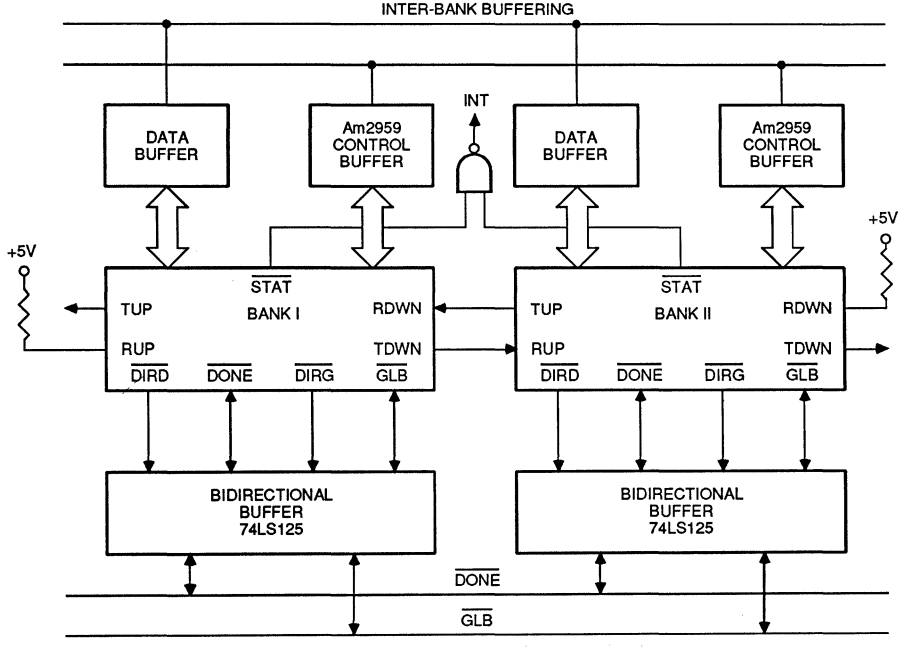
The simultaneous condition of active Read Enable and Chip Select indicates that information internal to the Am95C85 CADM needs to be transferred to the data bus. Read and Write are not allowed to be active simultaneously.

$\overline{WE}$ Write Enable (Input, Active LOW)

The simultaneous condition of active Write Enable and Chip Select indicates that information from the data bus is to be transferred to an internal location.

C/$\overline{D}$ Command/$\overline{Data}$ (Input)

This signal defines the type of information transfer



08035A 2-7

Figure 2-7. Buffering Banks of CADMs

performed by the Am95C85 CADM, i.e., command or data. A command byte is written into the CADM instruction register when this pin is HIGH. Data read and data write operations transfer data from and to the CADM during the period that this pin is LOW.

CLK Clock (Input)

The clock input determines the frequency of operation of the Am95C85. The lower limit on frequency (as specified in the A.C. Spec.) is imposed because of the refresh cycle requirements of the on-chip dynamic circuitry.

$\overline{T}$/R $\overline{\text{Transmit}}$/Receive (Output)

To operate banks of more than 16 Am95C85 devices in cascade without slowing down the clock frequency, bidirectional bus transceivers are required to isolate the data bus between banks of 16 devices each. This pin provides an input to the inter-bus transceiver to control the direction of data flow during a read, write or an off-line operation. Only the device that intends to put information on the Data Bus has its $\overline{\text{Transmit}}$/Receive output pulled LOW.

$\overline{\text{DONE}}$ Done (Input/Output, Active LOW, 3-state)

The $\overline{\text{DONE}}$ signal indicates the termination of an operation. This signal goes HIGH at the beginning of new commands, data writes, or data reads, then goes LOW to indicate that the CADM is ready for subsequent operations.

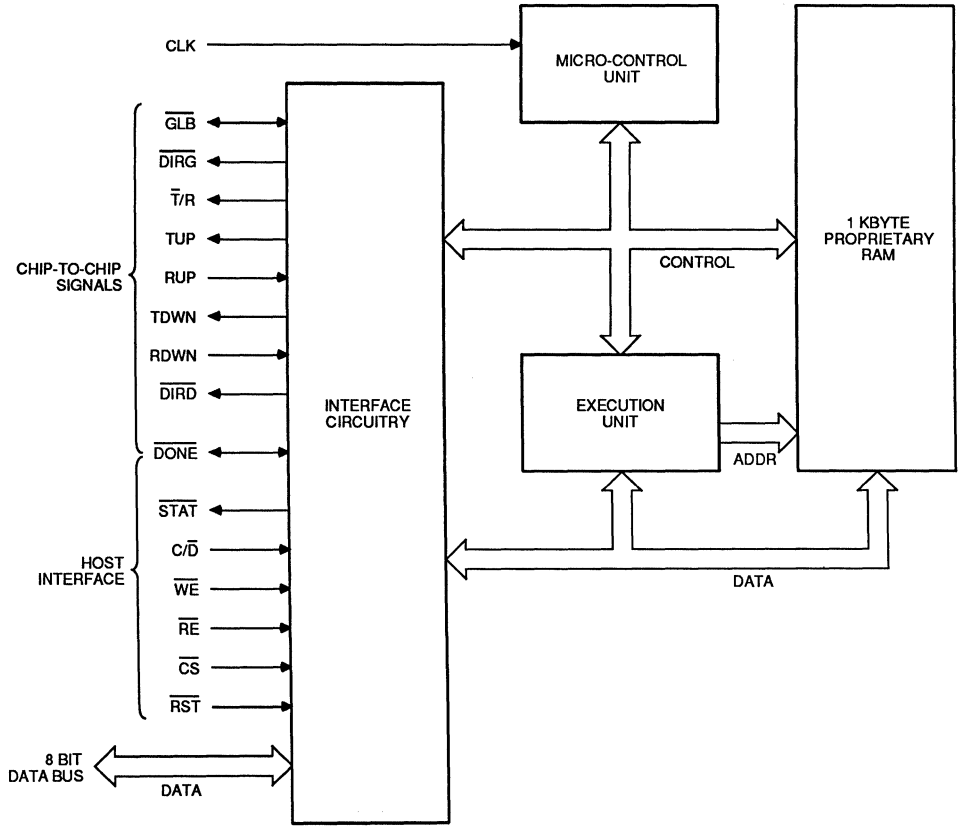


Figure 2-8. Am95C85 Block Diagram

$\overline{\text{STAT}}$ Status (Output, Active LOW, 3-state)

The Status signal indicates an exception condition following either a command or data access. A LOW level on this pin, after $\overline{\text{DONE}}$ signals completion, indicates that further action is needed by the host.

2.6.3 CHIP-TO-CHIP COMMUNICATION

The TUP, TDWN, RUP, and RDWN pins are used in various chip-to-chip communication functions in multiple Am95C85 memory configurations. Following are some typical examples.

TUP Transmit Up (Output, Active HIGH)

This signal, for example, is issued by a lower CADM to its next higher peer in the cascade to indicate that data is available, on the bus, to be latched in the input buffer space.

TDWN Transmit Down (Output, Active HIGH)

This signal, for example, is issued by a higher CADM to its next lower peer in cascade to indicate that data is available on the bus, to be latched in the input buffer space.

RUP Receive from the Up Direction (Input, Active HIGH)

This signal, for example, is received by a CADM from its next higher peer indicating that data is available on the bus to be latched in. This signal is connected to Vcc by a 1 kΩ resistor on the very first device in the CADM cascade.

RDWN Receive from the down Direction (Input, Active HIGH)

This signal, for example, is received by a CADM from its next lower peer indicating that data is available on the bus to be latched in. This signal is connected to Vcc by a 1 kΩ resistor on the last device in the CADM cascade.

TUP is connected to the RDWN and TDWN is connected to RUP on adjacent parts to enable inter-chip data transfers.

$\overline{\text{GLB}}$ Global (Input/Output, 3-state)

The signal is used for part-to-part synchronization during instruction execution.

$\overline{\text{DIRG}}$ Direction of GLB Signal (Output, Active LOW, Open Drain)

This output determines the direction of the GLB pin.

$\overline{\text{DIRD}}$ Direction of $\overline{\text{DONE}}$ signal (Output, Active LOW, Open Drain)

This output determines the direction of the $\overline{\text{DONE}}$ signal.

2.6.4 Supply Pins

Vcc Power Supply

GND Ground

CHAPTER 3

Am95C85 INSTRUCTION SET

This chapter contains detailed information about each of the 16 commands that constitute the Am95C85 instruction set. A summary of the instructions is shown grouped into three categories according to the function performed or the manner in which data is manipulated. These groups are:

Initialization Instructions
Byte-oriented Instructions
Record-oriented Instructions

Following this summary, the sixteen commands are described in alphabetical order.

3.1 INITIALIZATION INSTRUCTIONS

These commands initialize the CADM devices to prepare them for record oriented operation. The operations performed during an initialization sequence specify the number of the chips in cascade, the record length, and the bit masking option. The instructions in this category are:

KPL	Load the Length of the Key field, Length of the Pointer field and Last Address pointer
RST	Reset and enumerate CADM chips
SMB	Set Mask Byte

3.2 BYTE-ORIENTED INSTRUCTIONS

These commands operate on byte boundaries. This enables data transfers between the CADM and the host system and between the CADMs on a byte-by-byte basis. A user-transparent Address Pointer addresses one and only one byte in the entire array of CADM devices. All Reads, Writes, Pushes, and Pops will access data at the location pointed to by the Address Pointer. The byte-oriented instructions are:

AIM	Set Auto-Increment Mode
DEC	Decrement Address Pointer
GSF	Get Status Full
LAL	Load Address Long
LAS	Load Address Short
STK	Set Stack Access Mode

3.3 RECORD-ORIENTED INSTRUCTIONS

These commands operate on record boundaries. The record length must be set before any of the following instructions may be executed:

FND	Find a matching key
LUD	Load Unsorted Data
NXT	Point to next Record
PRE	Point to previous record
RRB	Restore Record Boundary
SOF	Sort Off Line
SON	Sort On Line

3.4 INSTRUCTION SET

All of the instructions are explained in detail in alphabetical order in the following pages.

AIM — Auto Increment Mode

Operation:

Read in Auto Increment Mode

Host System ← Data from Am95C85

Address pointer ← Address Pointer + 1

OR

Write in Auto Increment Mode

CADM Memory ← Data from Host System

Address Pointer ← Address Pointer + 1

Size: One command byte

Category: Byte-oriented Instruction

Description: The AIM command allows the user to access the CADM memory with a post increment of the Address Pointer. This mode allows the user to read from or write to the Am95C85s as if they were in continuous address space without the need to increment the Address Pointer externally. An RST command sets the CADM to the Auto Increment Mode (i.e., default mode).

Command/$\overline{\text{Data}}$: 1

Mnemonic: AIM

Hex value: 06

Machine code: 00000110

$\overline{\text{STAT}}$: The status signal is never asserted by the execution of an AIM command.

DEC — Decrement Address Pointer

Operation:	Address Pointer ← Address Pointer – 1
Size:	One command byte
Category:	Byte-oriented Instruction
Description:	This command decrements the value of the Address Pointer by one. If the current value of the Address Pointer addresses the first byte of the first record in one device, the execution of the DEC command will set the Address Pointer to point to the last byte of the last record in the preceding chip. (This location is set by the user, with the LA field of the KPL instruction.)
Command/$\overline{\text{Data}}$:	1
Mnemonic:	DEC
Hex value:	02
Machine code:	00000010
$\overline{\text{STAT}}$:	The status signal is asserted if the DEC command is executed when the Address Pointer points to the first byte of the first record in the first chip. In this case, the Address Pointer remains unchanged.

FND — Find a Matching Key

Operation: Address Pointer ← Address of First Byte of located record

Size: One command byte + 'k' literal bytes
(where 'k' is the number of bytes in the key field)

OR

One command byte

Category: Record-oriented Instruction

Description: The FND command normally requires k bytes of literal data to follow the FND command. These k bytes contain the key that is being searched for. The key bytes must be loaded in proper sequence, with the most significant byte first. The key bytes are saved in the input buffer space at the end of each chip. When all key bytes are loaded, all of the chips initiate a search to obtain a match for the loaded key in user data space. *The data must be sorted prior to a FND being executed.*

If the CADM finds a match, then the Address Pointer contains the address of the first byte of the located record. If no match was found, then the Address Pointer contains the address of the next higher key that was found. The status line is asserted to indicate this event.

If the CADM array contains more than one record with the desired key, then the first occurrence of the record in the entire set of cascaded devices is located when a FND with key value is executed.

If more records matching a particular key value are to be located, additional FND commands without a key following the command can be issued. In this case, the value of the key contained in the input buffer space from the previous FND is used. The Address Pointer is incremented and the key comparisons are performed. This continues with each subsequent FND. To terminate this mode of operation, for instance to allow a new record to be sought, a command other than FND or RRB should be issued. The CADMs will then expect a subsequent FND command to be followed by a new key for which to search.

Mnemonic:

FND	KEY	KEY	KEY	...

Command/$\overline{\text{Data}}$:

1	0	0	0	...

Hex value: 03

Machine code:

00000011	DDDDDDDD	DDDDDDDD	DDDDDDDD	...

$\overline{\text{STAT}}$: The status line is asserted if no key in the record space matches the key specified.

GSF Get Status Full

Operation: $\overline{STAT}$ ← LOW (if no record space is available)

$\overline{STAT}$ ← HIGH (if record space is available)

Size: One byte command

Category Byte-oriented Instruction

Description: The GSF command allows the user to determine the availability of empty memory space in the on-chip RAM. This command indicates whether or not one more byte of data can be inserted into the user space in the CADM.

Command/$\overline{Data}$: 1

Mnemonic: GSF

Hex value: 0F

Machine code: 00001111

$\overline{STAT}$: The status signal is asserted if and only if the device cannot hold even one more byte of user data (i.e., all devices are full).

KPL — Load Length of Key Field, Length of Pointer Field and Last Address Pointer

Operation:

Key Length ← First Literal Byte

Pointer Length ← Second Literal Byte

Last Address Location ← Third and Fourth Literal Bytes

Size: One command byte + four literal bytes

Category: Initialization Instruction

Description: This command configures the CADM memory such that the record boundaries are well defined. The KPL command also sets the address of the last memory location that can hold user data in each device. This command must be issued by the user before any of the record-oriented commands may be executed.

This command resets the Mask option. (See SMB command.)

The first literal byte of this command contains a value k, where k is the number of bytes in the key field. The key field may vary between 1 byte and 255 bytes. The second literal byte contains a value p, where p defines the length of the pointer field in each record. The third and fourth literal bytes contain a value LA, the address of the last usable byte in each Am95C85. The value of LA depends on whether or not masking is used and can be calculated from the equations in Chapter 2.

Command/$\overline{\text{Data}}$:

1	0	0	0	0

Mnemonic:

KPL	K	P	LA(LSB)	LA(MSB)

Hex value: 08

Machine Code:

00001000	DDDDDDDD	DDDDDDDD	AAAAAAAA	XXXXXXAA

$\overline{\text{STAT}}$: The status signal is asserted if the first literal byte (i.e., the length of the key field) is zero.

LAL Load Address Long

Operation: Address Pointer ← First and Second Literal Bytes

Device Identification ← Third Literal Byte

Size: One command byte + three literal bytes

Category: Byte-oriented Instruction

Description: The LAL command loads an 18-bit address into the Am95C85s which is sufficient to specify exactly one byte of data when a maximum of 256 CADMs are cascaded. The third literal byte contains an 8-bit number which, when compared to the chip identification number specifies the device to be accessed. The second literal byte contains only two bits of meaningful address which effectively becomes the two most significant bits of the byte address. The first literal byte has eight bits of address. These eight bits when combined with the two bits from the second literal byte form a 10-bit address which is common to all CADMs and can point to one of the 1024 bytes of each CADM memory.

Command/$\overline{\text{Data}}$:

1	0	0	0

Mnemonic:

LAL	BYTE ADR	BYTE ADR	CHIP ADR

Hex value: 0D

Machine code:

00001101	AAAAAAAA	XXXXXXAA	AAAAAAAA

where A = a bit of the address

$\overline{\text{STAT}}$: The status signal is asserted if the the device selected does not physically exist (i.e., if the number given in the third literal byte is equal to or exceeds the number of CADM devices in the cascade).

LAS Load Address Short

Operation: Address Pointer ← First and Second Literal Bytes

Size: One command byte + two literal bytes

Category: Byte-oriented Instruction

Description: The LAS command is similar to the LAL, except that no device identification is given in the LAS instruction. Instead, the byte address is used to point to a byte of data in the currently selected device.

Command/$\overline{\text{Data}}$:

1	0	0

Mnemonic:

LAS	BYTE ADR	BYTE ADR

Hex value: 01

Machine code:

00000001	AAAAAAAA	XXXXXXAA

$\overline{\text{STAT}}$: The status signal is never asserted by this command.

LUD Load Unsorted Data

Operation: CADM Memory ← Unsorted Data

Size: One command byte + n Literal Bytes
Where: n is the number of bytes of data to be loaded, and n is an integral multiple of the record size.

Category: Record-oriented Instruction

Description: The LUD command loads a block of unsorted data into the CADM devices. The total number of bytes loaded must be an integral multiple of r = (k + p) bytes, where r is the length of a record field. The CADM assumes that all bytes loaded after the LUD command are data bytes, until the next command is issued by forcing the C/$\overline{D}$ line HIGH. This newly loaded, unsorted data must be sorted by issuing a SOF (Sort Off-Line) command. The LUD shifts the Address Pointer to the end of existing data. The data following a LUD is appended to previously existing meaningful record data if any. *The previously existing data is assumed to be sorted.*

Command/$\overline{Data}$:

1	0	0	0

Mnemonic:

LUD	DATA	DATA	DATA	...

Hex value: 0B

Machine code:

00001011	DDDDDDDD	DDDDDDDD	DDDDDDDD	...

$\overline{STAT}$: The status signal is asserted after the command opcode if the entire bank of CADM memory is full and no more data can be accepted, or after a data byte write cycle if the CADM array just filled up.

NXT — Point to Next Record

Operation: Address Pointer ← Address of the next record's MSB

Size: One command byte

Category: Record-oriented Instruction

Description: The NXT command loads the Address Pointer with the address of the first byte of the next record.

The 'next record' is defined as being that following the last record located by either a FND operation, RRB, PRE, or a NXT operation on sorted data.

Command/$\overline{\text{Data}}$: 1

Mnemonic: NEXT

Hex value: 04

Machine code: 00000100

$\overline{\text{STAT}}$: The status signal is asserted if the execution of a NXT command will leave the Address Pointer pointing to meaningless data. The Address Pointer will be left pointing to the first byte beyond meaningful data.

PRE — Point to Previous Record

Operation: Address Pointer ← Address of previous record's MSB

Size: One command byte

Category: Record-oriented Instruction

Description: The PRE command loads the Address Pointer with the address of the first byte of the previous record. The PRE command will not decrement the Address Pointer, if the Address Pointer points to the first user data byte in the first device.

The 'previous record' is defined as that prior to the record located by the last FND, PRE, RRB, or NXT instruction.

Command/$\overline{\text{Data}}$: 1

Mnemonic: PRE

Hex value: 0E

Machine code: 00001110

$\overline{\text{STAT}}$: The status signal is asserted if the execution of this instruction attempts to load the Address Pointer with an address less than that of the first record in the first CADM. The pointer is loaded with the address of the first record.

RRB Restore Record Boundary

Operation: Address Pointer ← Current Record's MSB

Size: One command byte

Category: Record-oriented Instruction

Description: The RRB command provides an efficient means of restoring the Address Pointer to the current record. The current record is defined as that located by the last FND, RRB, PRE, or NXT instruction.

Command/$\overline{\text{Data}}$: 1

Mnemonics: RRB

Hex value: 05

Machine code: 00000101

$\overline{\text{STAT}}$: The status signal is asserted if the Address Pointer will address a record that does not lie within meaningful data. Monitoring the condition of the status signal after an RRB is executed can verify whether or not the Address Pointer points to meaningful user data.

RST Reset

Operation: Initialize CADM Array and Enumerate Chips

Size: One command byte

Category: Initialization Instruction

Description: This command resets the internal state of the Am95C85. The RUP and RDOWN signals are sampled to locate the first and last device in a bank of Am95C85s. The first device has its RUP tied HIGH, while the last device has its RDOWN tied HIGH. Next the devices are enumerated and the device identification number of each is stored in its device address register.

The reset also logically clears the CADM memory by setting the address of the next free byte to location zero in the first device which indicates that all of memory contains meaningless data. A read issued by the user, immediately after the reset, will indicate the number of devices in cascade.

Note: While RST logically clears the CADM memory, it does not physically clear the memory and therefore, the data can be recovered. *Hence, the RST should not solely be relied upon for security purposes.*

Command/$\overline{\text{Data}}$: 1

Mnemonic: RST

Hex value: 00

Machine code: 00000000

$\overline{\text{STAT}}$: The status signal is never asserted by this command.

SMB Set Mask Bytes

Operation: First k Bytes of CADM Memory ← k Mask Bytes
Where k is the number of bytes in the key field

Size: One command byte + k literal bytes

Category: The SMB command falls into two categories:
(a) Record-oriented Instruction
(b) Initialization Instruction

Description: The SMB command loads k bytes of literal data into the first k locations in each CADM. These k bytes are used as mask bytes to selectively mask out bits in the key field of all records during a sort or find operation by logically ANDing the mask with the key. The most significant mask byte is written first. *The first k bytes in each Am95C85 are reserved for the mask only if the masking option is chosen by issuing an SMB command.*

The SMB command may also be used to simply indicate to the CADMs during initialization, that the user plans to use the masking option later on for record manipulation. In this case the SMB is issued with all literal bytes following it set to zero. In this case, the first k bytes of each CADM are reserved for the mask bytes. The actual masking pattern can be supplied later with the execution of another SMB command.

Command/$\overline{\text{Data}}$:

1	0	0	0

Mnemonic:

SMB	MASK	MASK	MASK	...

Hex value: 09

Machine code:

00001001	DDDDDDDD	DDDDDDDD	DDDDDDDD	...

$\overline{\text{STAT}}$: The status signal is never asserted by the execution of the SMB command.

SOF — Sort Off Line

Operation:	Sort Unsorted Records
Size:	One command byte
Category:	Record-oriented Instruction
Description:	The SOF command follows either a LUD (Load Unsorted Data) command or an SMB (Set Mask Bytes) command, initiating an off-line sort process. The unsorted data in the CADM memory is sorted without any assistance from the host system. In the case of SMB and the first LUD, the SOF command works with the entire record content. In the case of data appended to an existing data array, the SOF command works with recently written records. The action in the latter case is to take each unsorted record and place it in its sorted position within the existing records.
Command/$\overline{\text{Data}}$:	1
Mnemonic:	SOF
Hex value:	0C
Machine code:	00001100
$\overline{\text{STAT}}$:	The status signal is asserted at the end of the sort if the record space within the CADM array is full.

SON — Sort On Line

Operation: Insert records into sorted positions

Size: One command byte + n records

Category: Record-oriented Instruction

Description: The SON command is used to insert records into already sorted data. Following a write of the last byte of each record, this record is inserted into the proper location within the sorted data.

Command/$\overline{\text{Data}}$:

1	0	0	0

Mnemonic:

SON	DATA	DATA	DATA	...

Hex value: 0A

Machine code:

00001010	DDDDDDDD	DDDDDDDD	DDDDDDDD	...

$\overline{\text{STAT}}$: The status signal is asserted immediately after the SON if the CADMs are full or, after a write of the last byte of a record if the insertion of that record has filled the CADM record space.

STK — Stack Access Mode

Operation:

Read in Stack Access Mode

Host system ← Data from CADM memory plus POP Operation

OR

Write in Stack Access Mode

CADM memory ← Data from host system plus Push Operation

Size: One command byte

Category: Byte-oriented Instruction

Description: The STK command allows access of data from the CADM location pointed to by the current value of the Address Pointer. The value of this pointer remains unchanged during subsequent memory accesses, but all bytes below the point of access are moved upward or downward depending on whether the memory access constituted a read or a write.

Command/$\overline{\text{Data}}$: 1

Mnemonic: STK

Hex value: 07

Machine code: 00000111

$\overline{\text{STAT}}$: The status signal is never asserted by the execution of the STK command.

CHAPTER 4

PROGRAMMING THE CADM

This chapter discusses the programming support needed to design a system that uses CADMs. The software guidelines consist of the important command sequences to be followed as well as the command sequences to be avoided.

4.1 REQUIRED SOFTWARE COMMAND SEQUENCES

4.1.1 TYPICAL INITIALIZATION SEQUENCE

After switching on power, a few simple steps must be executed to provide a frame of reference for the Am95C85s. A flow-chart of the initialization sequence is shown in Figure 4-1.

The first step is to reset the devices. The reset may be performed either by an RST (reset) command or by a hardware reset (both have the same effect on the Am95C85s). The hardware reset is initiated by asserting the reset pin LOW for at least four clock cycles. One of the major functions of the reset is to enumerate the cascaded CADM devices. If a read is issued with the Command/$\overline{\text{Data}}$ pin LOW following a reset, the last CADM in the cascade places its device identification number on the data bus. Thus the user can determine the number of devices in cascade.

The second step in the initialization process provides the record size to the CADMs. The execution of the KPL (load key, pointer and last address) command, configures the memory for a fixed number of bytes in the key and pointer fields and sets the location of the last address in each Am95C85.

The last step of the initialization process indicates to the Am95C85s whether or not masking of selected key bits is to be used during sort and search operations. This is accomplished by executing the SMB (set mask bytes) command. This step is optional and the SMB command need not be issued if none of the key bits are to be masked during data manipulation.

If masking is to be used at some future time, it is best to reserve space for the mask with the SMB command. If data is loaded before the mask area is reserved and masking is to be used, then the data has to be reloaded after the SMB is issued. Unless the user is at the limit of usable space, it is a good practice to reserve the mask bytes during initialization.

Hence, the initialization sequence consists of:

(a) Reset
(b) Load Key, Pointer and Last Address
(c) Set Mask Bytes (optional)

4.1.2 SORTING OFF LINE

The Am95C85 capability to Sort Off Line ensures that the host CPU is not disturbed until the entire off-line operation is completed, at which time the CADM informs the CPU by asserting the $\overline{\text{DONE}}$ signal. The CADM's co-processing capabilities not

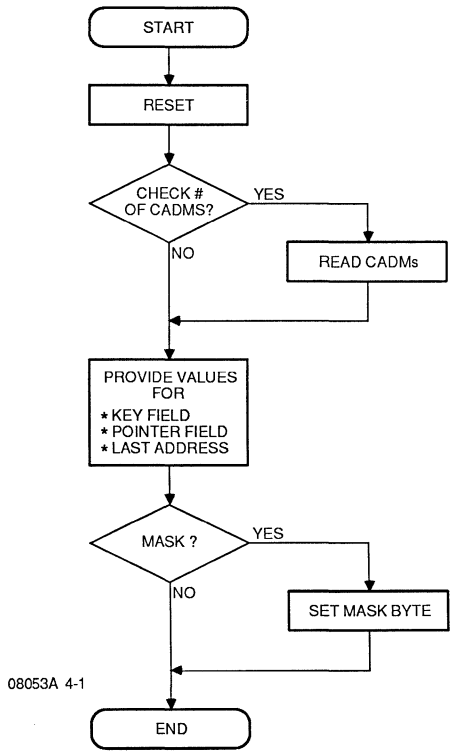


Figure 4-1. Initialization Sequence

only free the host from the chore of performing repetitive, time-consuming software-intensive tasks, but also significantly improve the overall performance.

Unsorted data is first loaded into the CADMs (using the Load Unsorted Data (LUD) command) either by the CPU or by DMA. The number of bytes loaded must always be an integral multiple of the number of bytes per record to ensure that complete records are loaded.

The SMB and LUD commands must be followed by a Sort Off Line (SOF) command so that future searches on the newly loaded records are meaningful. An exception to this rule is explained in Section 4.5.1.

Hence, an off-line sort operation (Figure 4-2) normally consists of:

(a) Set Mask Bytes (optional)
(b) Load Unsorted Data
(c) Sort Off Line

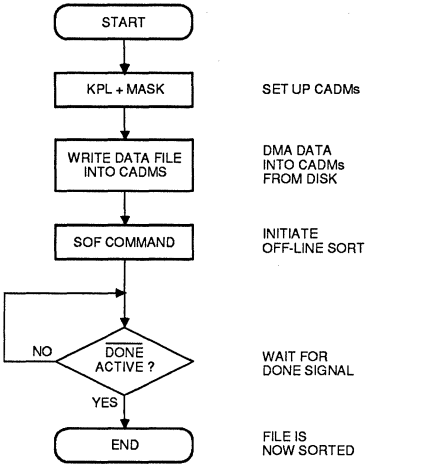


Figure 4-2. Simplified Off-Line Sort Sequence

4.1.3 SEARCH FOR A MATCHING KEY

In order to locate a particular record the Am95C85 must match the user- supplied key with a key value in its content-addressable memory. This is accomplished by executing the FND instruction consisting of the FND command followed by 'k' bytes of the key value required to be matched. These 'k' bytes are stored in the input buffer space at the end of each device. When a match is found, the Address Pointer is set to the first byte of the located record. The entire record can then be read.

Multiple records with identical key values are located in the CADMs in consecutive record locations. If no command other than RRB has been issued since the last FND, the CADMs interpret the next FND as 'find next', and no key data is permitted. The key value stored in the input buffer space will be used for a search.

This operation internally performs a NXT instruction followed by a comparison of the key. Multiple FND commands may be issued without key values if no other command except RRB is issued between them. Issuing any command (except RRB or FND) after a FND logically clears the input buffer space of the current key value. The next time a FND is issued it must be followed by 'k' key bytes.

From the previous discussion, it follows that if FND commands are to be issued for different key values, they must be separated by at least one command other than RRB or FND. Any command except the RRB or FND can be used for this purpose. The safest command to use would be the GSF (Get Status Full) instruction since this instruction does not alter the state of the data or Address Pointers in any of the devices.

The sequence to be followed for a record search is shown by the flowchart in Figure 4-3.

4.1.4 RECORD-ORIENTED DATA ACCESS

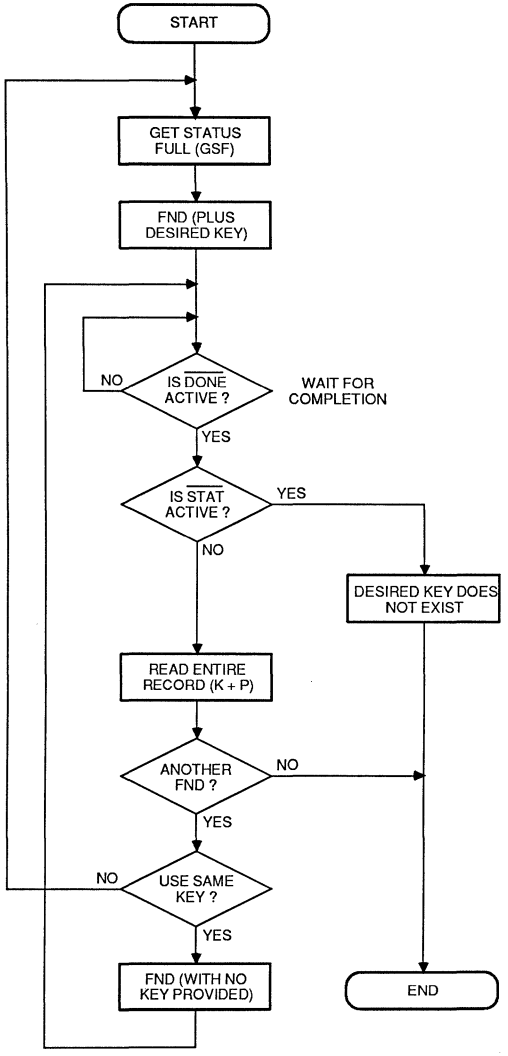
The CADM provides several features enabling the user to access data by record content rather than physical address. In reality, this is a means of setting the Address Pointer to the first byte of a record of interest. The FND instruction does this automatically. The user may then wish to use byte-oriented commands, such as DEC, LAL, or LAS, to move elsewhere within the CADM record space, prior to returning to the record of interest. Three (3) instructions provide record-oriented pointer control; RRB, PRE, and NXT. They all load the Address Pointer with the location of the first byte of a record, the one adjacent to, or the one itself most recently selected by FND, RRB, PRE, or NXT.

In this way, the user can manipulate the pointer anywhere in the array and still have a means of returning to the record of interest.

4.2 COMMAND SEQUENCES TO BE AVOIDED

Certain command sequences must be avoided to ensure predictable results from the Am95C85. If proper logic is used during software development, these sequences will never occur, but sometimes they may not be absolutely obvious. The command sequences to be avoided are as follows:

1. Attempting to execute any record-oriented command before issuing the KPL command. The results from the CADM would be meaningless. It may even cause the devices to hang up by not returning the $\overline{\text{DONE}}$ signal. Since the KPL command defines the record boundaries and the location of the last user-available byte in each device, it is imperative that this command be issued at the beginning of the software routine. Also, if new data (sorted or unsorted) with a different record size from the previous data is to be loaded into the CADMs, the KPL instruction must be issued to reflect the new record boundaries before this data is loaded.



08053A 4-3

Figure 4-3. Record Search Sequence

2. The SOF (Sort off Line) command should be preceeded by an SMB or LUD command.

3. If a FND command is attempted on unsorted data, then the devices may not return the $\overline{\text{DONE}}$ signal. In the case of a user error causing the $\overline{\text{DONE}}$ signal to remain High, the contents of the CADM are not destroyed but the outcome of the last operation is not defined.

4. The Set Mask Bytes command must never be issued after the devices have valid data if the masking option was not chosen before loading the data. Doing so would guarantee loss of data in the first 'k' bytes of all the CADMs and also cause the devices to lose their frame of reference of the record boundaries. The mask area must be reserved before loading of any data if masking of selective key bits is desired for any subsequent finds and sorts.

   If the mask needs changing, and new data needs to be appended to a sorted array with the LUD command, the order of the commands is important. First, append the data, then change the mask.

   A good way to avoid this problem is to always select the mask option during initialization with no bits masked. Then if it is necessary to mask something later, the SMB command can be re-issued with mask bits. The only penalty in doing this is that each CADM loses k bytes (length of the key field) of its memory space if no mask were to be used for the sort and search process.

4.3 BYTE BOUNDARY TO BIT BOUNDARY CONVERSION

The CADM is designed to interpret a Key and Pointer only in terms of byte boundaries.

The user model in the example shown in Figure 4-4 indicates that the user needs a 12-bit key field and a 12-bit pointer field. It would appear that the user would have to specify two bytes for the key field and two bytes for the pointer field leaving four unused bits in each field. However, with a slight manipulation of the CADM, the device can be used to respond to key and pointer fields on bit boundaries. This is where the SMB (Set Mask Byte) is useful.

In this example, the user can define the key field as two bytes long and the pointer field as one byte long giving a record length of 24 bits. The records are loaded into the CADM as 24-bit records. The mask is specified as shown in Figure 4-4 to mask off the rightmost four bits of the key field. Subsequent Sort and Find commands then sort and find records with respect to the leftmost 12 bits of each record which is the user's key field. When the desired record is found by the CADM, it can be read into the user's memory. The user reads the record from the CADM one byte at a time. The user software must know that the pointer field of the record consists of 12 bits (i.e., the rightmost four bits of the second byte and all eight bits of the third byte). Hence, as long as the user software keeps track of the key and pointer boundaries, substantial CADM memory space is saved by splitting a byte between a key and pointer as shown in this example.

This example shows the flexibility of the CADM in adapting to user records in which the key and pointer fields are not on byte boundaries.

4.4 DATA MANIPULATION

Chapter 2 discusses cascading multiple CADM devices to provide enough CADM memory to meet system file size requirements. In some cases, economic or logic constraints could restrict the available CADM memory to be less than the maximum length of an index file to be manipulated.

For example, a CADM system is initially designed to handle a finite length of data considered adequate at the time of design. But as time progresses, the CADM system is found to be suitable for many other application areas requiring manipulation of larger data bases than the physical memory available and an alternative solution must be developed to up-grade the system without changing the hardware.

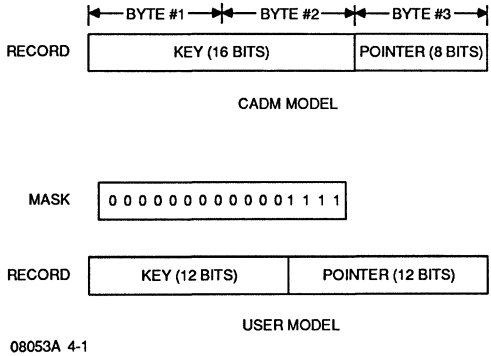


Figure 4-4. Boundary Conversion Example

This section shows how Quicksort can be used with the CADMs for maximum efficiency. Consider an example wherein the physical CADM memory available is 4 Kbytes but the data to be sorted requires 6 Kbytes.

Layer 1 of Quicksort, when executed, selects a key at random from the records (usually the first key) and separates the data into two parts around that key value. One part has all of the keys that are smaller than the selected key and the other part has all of the keys that are larger than the selected key.

The two parts of this data file are loaded into the CADMs one part at a time and an Off-line Sort is performed on each part. The result is two sorted files, one containing all the lower key values and the other containing all the higher key values in the file. These two files are combined and stored in system memory as one sorted file.

Sometimes, it may happen that the key picked at random by Quicksort may divide the data file in such a way that one part of the file is much larger than the other. For example, in a 6 Kbyte file, 5 Kbytes may have key values lower than the value picked by Quicksort and consequently be placed into the same part. The 5 Kbyte part would not fit into the CADMs of this example. The solution is to run layer 1 of Quicksort once more on the 5 Kbyte part dividing it into two parts. Then there would be three parts that need to be sorted by the CADMs.

Using this method, the best case timing takes *nlogn* iterations and the worst case takes n^2 iterations, where:

$$n = \frac{\text{Number of records in data file}}{\text{Available CADM record space}}$$

4.5 HELPFUL HINTS

4.5.1 USING THE LUD COMMAND

When data is loaded into the CADMs using the LUD command, the previously existing data in the CADMs is assumed to be sorted. This feature can be useful if the data being loaded into the CADMs has been pre-sorted. Assuming that no other meaningful data existed in the CADMs before the LUD is issued, issuing a Sort Off-line would be a waste of time since the data is already sorted. After the data is loaded with the first LUD, issuing a second LUD indicates to the CADM that the data existing in it is sorted.

LUD | data | data | data | ..... (presorted data)

LUD (declares that data is sorted)

4.5.2 KEEP THE POINTER WITHIN MEANINGFUL DATA

Although the LAL and LAS commands can be used to point to any location within CADM memory, the pointer must not point to the Mask Space or Input Buffer Space while performing PUSHES or POPs in Stack Mode. It is important that this pointer always point to meaningful data space within the CADMs.

4.5.3 LAST ADDRESS TOO HIGH

The value of the Last Address Pointer (set by the KPL command) must never be higher than the value 1023 minus the record length in bytes. The input buffer in each CADM is located at the high end of the address space and requires enough space for one record. For example, if a record within the CADM is only one byte long (a record must have at least one byte of key data), the value of the Last Address Pointer must be set no higher than 1022. Setting the pointer at 1023 would leave no space in the CADM memory for the input buffer. If a record is 100 bytes long, any Last Address Pointer value higher than 923 is wrong. The equation given in Section 2.2.4 gives the correct Last Address Pointer value to use for any allowable record length.

4.5.4 USING STAT IN POLLED MODE

The $\overline{\text{STAT}}$ (status) output is valid only when the CADM has completed an operation indicated by the $\overline{\text{DONE}}$ signal going active. Hence, $\overline{\text{STAT}}$ should be polled only after $\overline{\text{DONE}}$ goes active. If $\overline{\text{STAT}}$ is active, then branch to the current command's error recovery routine.

4.5.5 SMB DECLARES CADM DATA UNSORTED

Issuing the SMB (Set Mask Bytes) command to set or change a mask makes any previously sorted data appear unsorted because a new set of bits in the key field are now masked. This means that data already in the CADMs must be re-sorted with respect to the new mask bit patterns. Remember: SMB always declares the contents of the CADM unsorted.

CHAPTER 5

INTERFACE CIRCUITS

5.1 INTRODUCTION

This chapter discusses, in detail, various aspects of the hardware interface of the CADM that are common to many processors. In addition, the unique aspects of the hardware interface to three specific processors or systems are described.

The common topics discussed include:

DMA Transfer Mode
CADM Clock
CADM Status Output
CADM Bus to System Bus Isolation
CADM Data Bus Bank to Bank Isolation
CADM Local Signal Buffering
CADM Command/Data Select

The three specific application interfaces are:

A CADM interface to an IBM PC XT/AT
A CADM interface to an 8086 processor
A CADM interface to an MC68000 processor

5.2 DMA TRANSFER MODE

The DMA mode of data transfer is used to move a large block of data to or from the CADM memory. The two instructions which use DMA to write to the CADMs are:

1. The Load Unsorted Data (LUD) instruction which requires a fixed number of bytes to be dumped to the CADMs

2. The Sort On-Line (SON) instruction which also requires a fixed number of bytes to be loaded into the CADMs, but allows the CADMs to sort this data into an existing data base while it is being loaded

DMA can also be used while reading a large block of sorted data from the CADMs into system memory.

In either case (DMA read or write), the starting address in memory and the length of the data base to be transferred is loaded into the registers of the appropriate DMA controller channel which is then enabled. The single byte transfer mode or demand transfer mode may be used so that a DMA transfer is initiated by the simultaneous occurrence of three factors. They are:

1. An active DMA request (DREQ) input
2. The requested channel must be enabled
3. The word count for the requested channel must be non-zero

5.3 CADM CLOCK

The CADM can operate at the frequency range specified in the data sheet. In this interface, the clock that drives the CADM is deliberately unsymmetrical in order to provide higher performance. The minimum LOW time for the clock is specified in the data sheet. When multiple banks of CADMs are cascaded, the data bus buffers that isolate the banks from each other introduce additional delays in the data paths. When CADMs have to move data among themselves, the transmitting device places data on the local CADM bus on the rising edge of the clock. This data is latched in by the receiving device on the immediately following falling edge. Propagation delays through buffers during data transfers can be taken care of by lengthening the clock HIGH time while leaving the LOW time fixed as required by the A.C. specifications of the device.

The scheme used to implement this unsymmetrical clock involves a D-type flip-flop, a delay line, and an EX-OR gate. Refer to Figure 5-1. A 12 Mhz clock

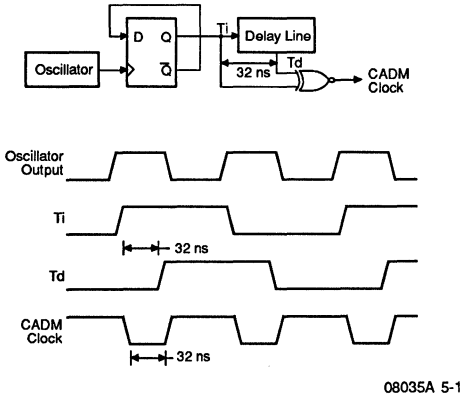


Figure 5-1. Unsymmetrical CADM Clock

is fed from an oscillator to D3 which is connected in a divide-by-two configuration and yields a 50% duty cycle. Output Q3 is fed into a delay line at T_i which is tapped at the 32 ns delay output, T_d. T_i and T_d are gated giving a 12 MHz output with a LOW time fixed at 32 ns and a HIGH time of 52 ns. If the propagation delay in the local data paths is increased, the oscillator frequency must be decreased. This is done by keeping the LOW time fixed, and increasing the HIGH time substantially, thus making the most efficient use of the extended time between the rising and the falling edge.

5.4 SYSTEM BUS TO CADM BUS ISOLATION

The CADM is different from most peripherals because it also acts as a coprocessor. While the CADM is performing search and sort operations off line without any intervention from the host, the host processor can be busy with other data transfer operations not involving the CADMs. This means that a system with multiple CADM devices must also have more than one data bus which can be electrically isolated from or connected to each other at will. Refer to Figures 5-2 and 5-3.

The two levels of buffering shown in Figures 2-6, 2-7, and 5-3 implement the data bus isolation. In addition, all the control signals need to be buffered between banks to provide sufficient drive capability. The Am2952A 8-bit bidirectional registered I/O port serves to isolate the host system data bus from the CADM data bus (Figure 5-4).

While writing to the CADMs, the output of the Am2952A is enabled by the write pulse to the CADMs and remains enabled for the duration of the write. The data is frozen in the registers on the falling edge of clock prior to the write enable going inactive. The CADM samples data during the LOW time of clock as long as its write is active, and it latches in the data that was sampled during the clock LOW time just before the write goes inactive.

During a CPU read from the CADMs, the output of the Am2952A is enabled by the read from the processor, qualified by an active $\overline{CS}$. The processor read is used in this case to enable the Am2952A ouputs because data may be latched by the processor after the read to the CADM has gone inactive. Output data from the CADMs is valid within 20 ns (worst case) after the falling edge of clock. Considering the propagation delay through the Am29863 and the set-up time required for the Am2952A, the data will not be valid at the Am2952A input port at the following rising edge. Hence, during a read, the data is clocked into the Am2952A by the falling edge of the CADM clock.

5.5 LOCAL CADM DATA BUS BANK-TO-BANK ISOLATION

The CADM outputs are designed to drive a maximum of about 200 pF capacitive load (i.e., about 16 CADM inputs). Hence, banks of 16 CADMs have to be buffered from each other. The Am29863 serves this purpose for the local data bus.

The $\overline{T}$/R signal which is normally HIGH is used to control the direction of data flow through the Am29863 data buffers. In the idle state, all CADMs are set up ready to receive data from either the host system or from any other CADM. When the CADMs have an active read signal, the $\overline{T}$/R signal is pulled LOW thus allowing any CADM to transmit data to the host. During an off-line operation, the CADMs have to transfer data among themselves. This involves one CADM transmitting and another receiving the data. The transmitting CADM places the data buffer at its bank in the transmit mode by forcing its $\overline{T}$/R signal LOW, while the buffers on all other banks are in the receive mode. This enables inter- chip communication involving inter-bank data transfers.

5.6 CADM STATUS OUTPUT

The Status ($\overline{STAT}$) output of the CADMs is pulled active (LOW) if an exception condition occurs when the CADMs are either executing an instruction or transfering data. The occurrence of an exception condition indicates that CPU attention is needed. After a $\overline{STAT}$ signal interrupt, the CPU looks at the current CADM instruction being executed and branches to an appropriate exception handling routine, which clears the fault. All $\overline{STAT}$ lines from up to eight banks of CADMs form the inputs to the 74LS30 NAND gate which outputs a single Interrupt signal. See Figure 5-3.

5.7 LOCAL CADM SIGNAL BUFFERING

The Am2959 supplies enough drive capability for the $\overline{CS}$, $\overline{RE}$, $\overline{WE}$, C/$\overline{D}$ and CLK inputs to the CADMs for each bank. These signals from the host system are buffered on every bank in the cascade of CADMs.

In addition, the Am2959 buffer on the first bank is the only one that buffers the $\overline{DONE}$ signal to the interface logic circuit. All $\overline{DONE}$ pins are tied together and have a 1 kohm pull-up resistor to Vcc.

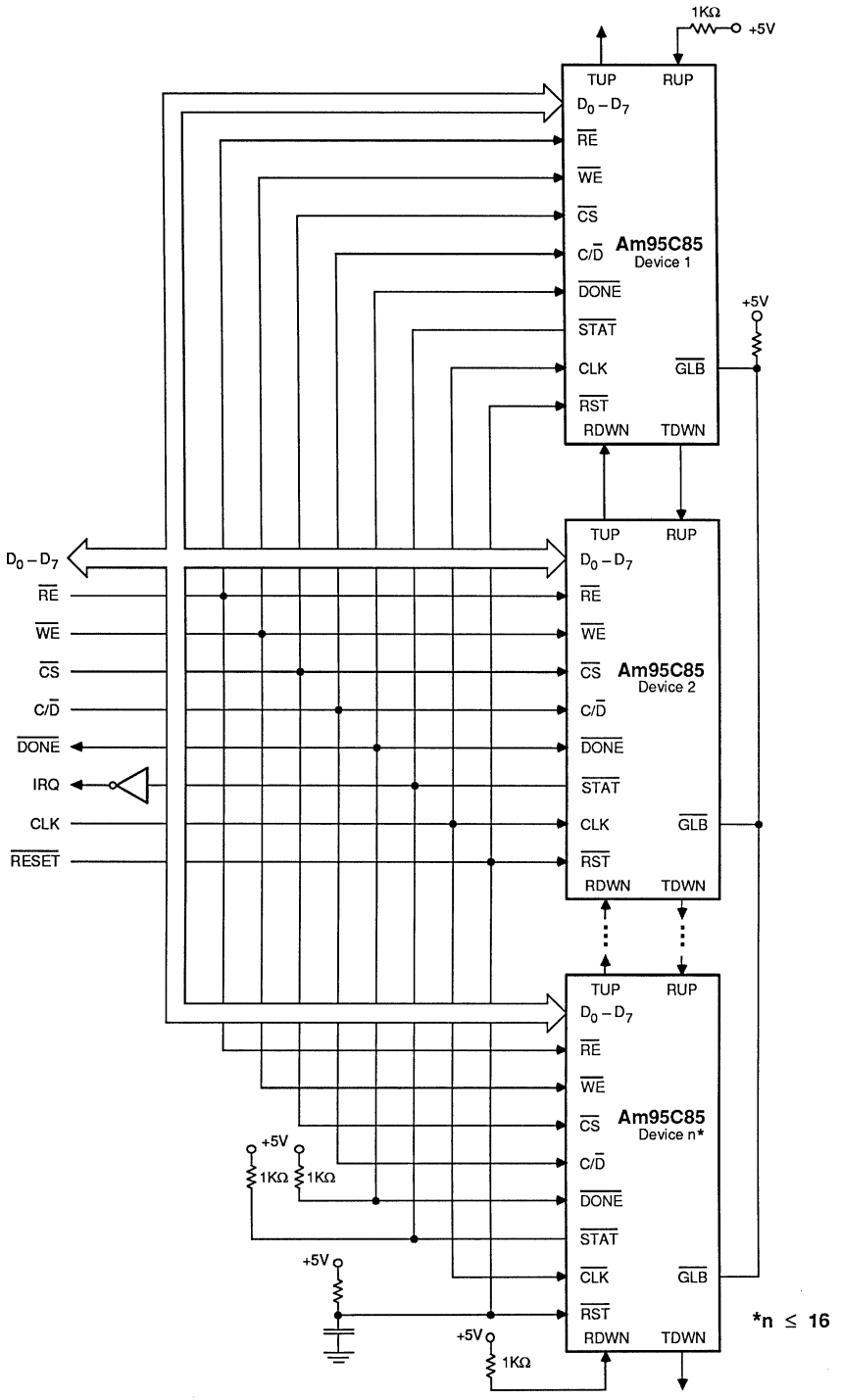
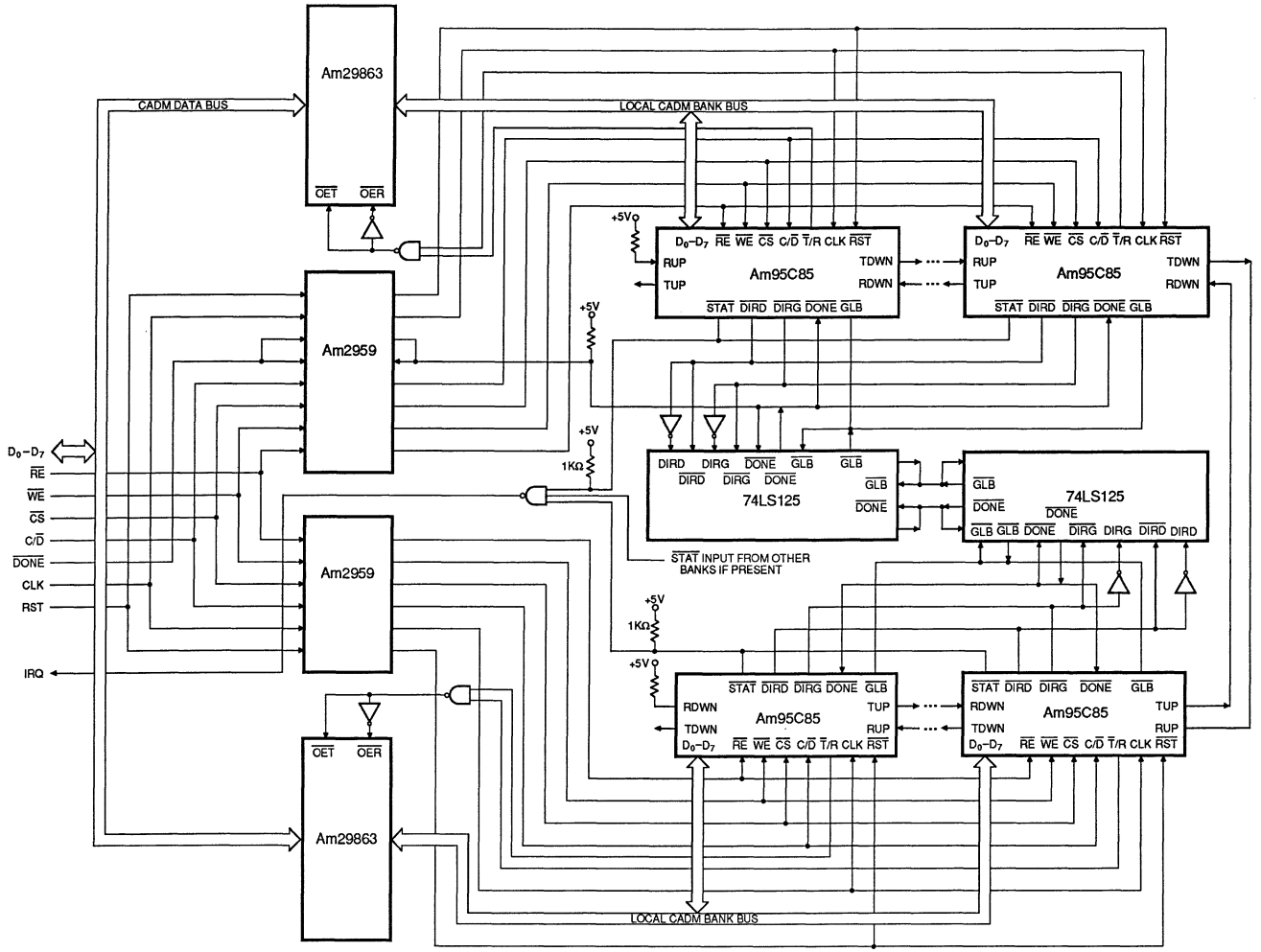
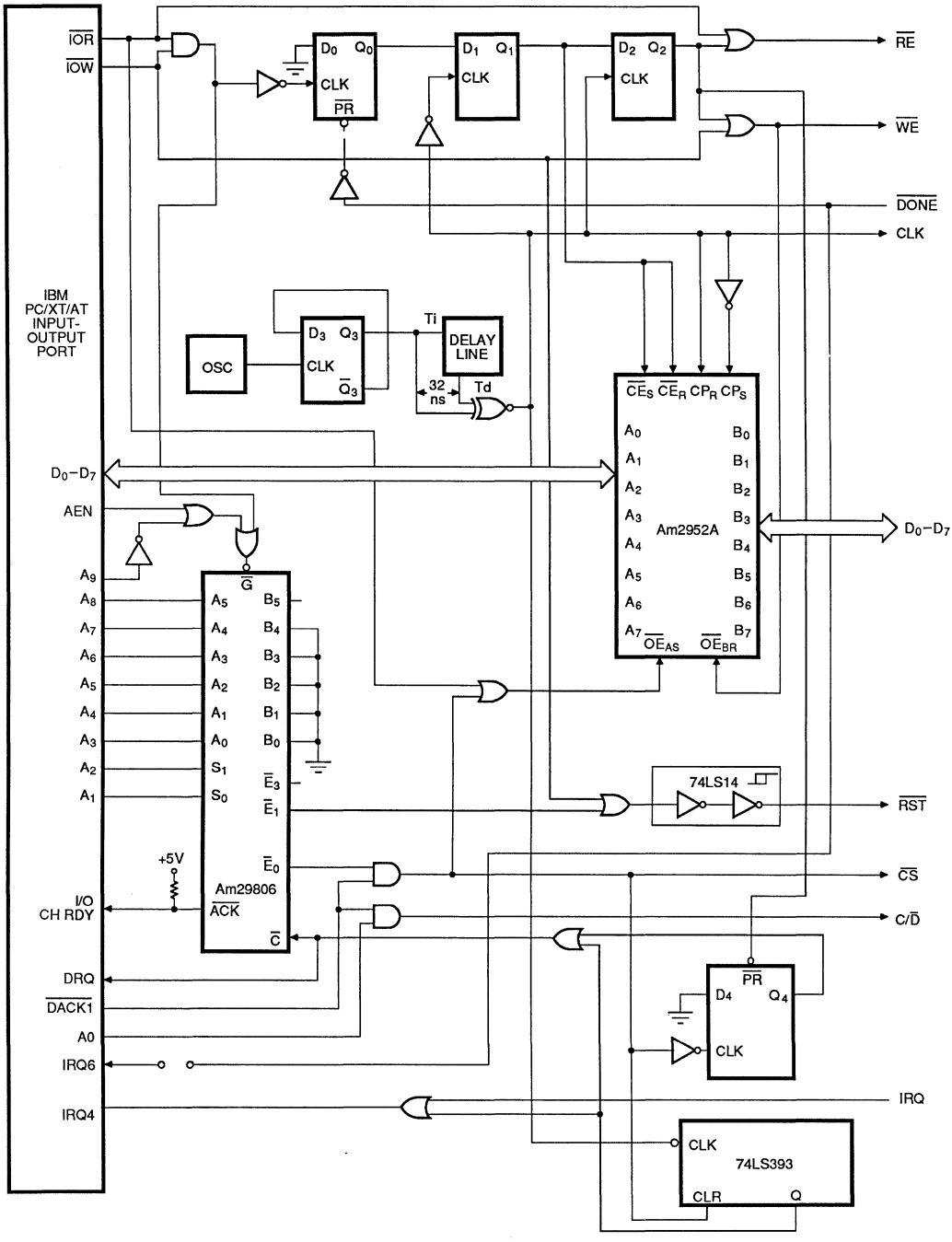


Figure 5-2. Cascading Up To 16 CADM Devices



08035A 5-3

Figure 5-3. Cascading More Than 16 CADM Devices



08035A 5-4

Figure 5-4. Am95C85–IBM PC/XT/AT Interface

The $\overline{\text{DONE}}$ and $\overline{\text{GLOBAL}}$ signals are bidirectional and also provide inter-chip communication. These are buffered between banks by the 74LS125 buffers. The Direction of Done ($\overline{\text{DIRD}}$) and Direction of Global ($\overline{\text{DIRG}}$) signals control the direction of these buffers. The detailed connection for the 74LS125 is shown in Figure 5-5.

The other inter-chip signals include RUP, RDWN,TUP, TDWN, daisy-chain from chip-to-chip as shown in Figure 5-2 and Figure 5-3 and are transparent to the user. The RUP on the first device and the RDWN of the last device are connected to Vcc through a pull-up resistor to allow the CADMs to enumerate and configure themselves during a reset operation. The reset can be performed either in software by the Reset ($\overline{\text{RST}}$) command or in hardware during a power up or by forcing the reset input LOW.

5.8 CADM COMMAND/$\overline{\text{DATA}}$ SELECT

Both the Command port and the Data port of the CADM are accessed through a single 8-bit data bus. The two ports are differentiated by the use of the Command/$\overline{\text{Data}}$ (C/$\overline{\text{D}}$) pin. During CPU transfers to the CADM, address line A0 distinguishes whether the current byte on the system data bus is a command to the CADM or simply data to be stored in the on-chip memory. If a DMA controller is to be used to transfer data, DMA Acknowledge ($\overline{\text{DACK}}$) may be used to force C/$\overline{\text{D}}$ pin LOW so that data can be written to or read from CADM memory.

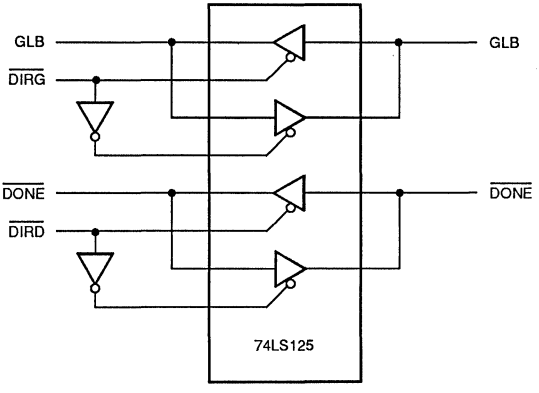
5.9 FORCING READY ACTIVE

During the development stages of the software for the CADM, the programmer may write an invalid sequence of commands that may cause the $\overline{\text{DONE}}$ signal from the CADMs to remain inactive (HIGH). The design of this hardware interface is such that access to CADMs is prohibited while $\overline{\text{DONE}}$ is HIGH. Hence, some other mechanism must be developed to externally force $\overline{\text{DONE}}$ active (LOW) following an invalid command sequence that cause $\overline{\text{DONE}}$ to remain HIGH.

The following discussion is based on the IBM PC/XT/AT. However, this discussion also applies to the 8086 and 68000 interfaces. The RDY signal follows the $\overline{\text{C}}$ input on the Am29806 whenever an address match exists. For the 68000 interface, the RDY is replaced by $\overline{\text{DTACK}}$. If the $\overline{\text{DONE}}$ signal remains inactive (HIGH) the RDY is prevented from going active (HIGH) and releasing the CPU. If the system is held in the WAIT state longer than allowed by system specification, (e.g., the IBM PC/XT/AT cannot be held in the WAIT state longer than 2.6 µsec as this will prevent a refresh of its dynamic memory), then the system may hang up.

The 74LS393 counter is enabled by the falling edge of $\overline{\text{CS}}$ and reset by the rising edge of $\overline{\text{CS}}$. During normal operation (with the exception of FND, KPL, SON, and SOF), the $\overline{\text{CS}}$ pulse width is shorter that 16 clock pulses (the count-down time on the counter). Hence, the counter is reset before it can count down 16 clock pulses to activate the RDY line.

If one of the operations (FND, SON, SOF, or KPL) is being performed or if the $\overline{\text{DONE}}$ signal remains HIGH because of an illegal command sequence, then the $\overline{\text{DONE}}$ signal may not return to LOW or may take too long. If the CADM is accessed while the $\overline{\text{DONE}}$ is HIGH, Q_2 is prevented from going LOW which in turn forces RDY to remain inactive (i.e., inserting WAIT states). In this case, the



08035A 5-5

Figure 5-5. 74LS125 Logic Diagram

counter counts down the 16 pulses and forces a RDY signal to the system. It also sends an interrupt, IRQ4, to the system to inform the processor that this is a special case. The processor then checks the command issued before the current device access.

If it was one of the four commands (FND, SON, SOF, or KPL), the processor tries to access the device after waiting a predetermined amount of time or it polls the $\overline{DONE}$ before the next access.

If the $\overline{DONE}$ remains HIGH because the user issued an illegal sequence of commands such as a FND on unsorted data (this may happen during debugging), the CADMs must be reset to force $\overline{DONE}$ LOW.

This counter provides a mechanism to prevent a hang-up of the entire system caused by a software error. The 16 clock cycle count-down is appropriate for a 10 MHz CADM clock. The number of clock pulses needed for the counter to force a RDY depends on the CADM clock frequency and the requirements of the system.

5.10 The Am95C85 (CADM) INTERFACE TO AN IBM PC/XT/AT

The interface of the Am95C85 CADM to an IBM PC/XT/AT is shown in Figure 5-4 combined with either Figure 5-2 or Figure 5-3. If 16 or fewer CADMs are used, Figure 5-4 is combined with Figure 5-2. If more than 16 CADMs are used in the application, Figure 5-4 is combined with Figure 5-3. This discussion deals with the generation of the hardware interface signals. This interface consists of three separate data and control buses that can be isolated from each other. The three buses are:

1. The System Bus
2. The CADM Bus
3. The Local CADM Bank Bus

The Write and Read timing diagrams for this interface is shown in Figures 5-6 and 5-7. The clock mentioned in this discussion is the CADM clock unless otherwise stated.

This section discusses in detail how and why each CADM interface control signal is generated. Due to the unique nature of this device, some of the control signals have special timing requirements.

5.10.1 SYNCHRONIZING THE READ AND WRITE SIGNALS

Whereas, most peripherals accept asynchronous control signals, the Am95C85 requires that its Read ($\overline{RE}$) and Write ($\overline{WE}$) inputs be synchronized with its clock. The I/O Read ($\overline{IOR}$) and I/O Write ($\overline{IOW}$) signals from the input/output port are ANDed together. This signal is then qualified by the $\overline{DONE}$ control output from the CADM to ensure that the CADM has completed the previous operation before any further access is allowed. At this point we shall assume that the $\overline{DONE}$ output from the CADM is active (LOW). The case when $\overline{DONE}$ is inactive (HIGH) at a point in time when $\overline{IOR}/\overline{IOW}$ goes active (LOW) is dealt with separately in this discussion.

The falling edge of the clock following the read/write being active at the D1 input passes the signal to Q1. The next rising edge generates the read/write signal for the CADMs which is then appropriately gated with the $\overline{IOR}/\overline{IOW}$ from the system port to separate the $\overline{RE}$ and $\overline{WE}$ signals. After the CADMs receive the read/write signal, they 3-state the $\overline{DONE}$ signal (which has a 1 kohm pull-up to Vcc) on the next falling edge. On the subsequent rising edge, the CADMs force the $\overline{DONE}$ line to the inactive (HIGH) state to indicate that they are busy with the current device access.

The $\overline{DONE}$ now drives the D1 input to the HIGH state. Q1 will now go HIGH on the next falling edge of clock. Flip-flop Q1 is clocked on the falling edge because $\overline{DONE}$ is stable in the HIGH state before the falling edge, whereas its state is undefined at the rising edge. The subsequent rising edge of clock drives Q2 HIGH thus de-activating read/write to the CADMs. This mechanism of generating the read/write signals for the CADMs meets two requirements:

1. The read/write signals to the CADMs shall meet the A.C. specifications of set up time with respect to CADM clock.

2. The read/write signals shall be active (LOW) for at least two clock cycles. This scheme guarantees that these signals are active for exactly two clock cycles, for any operating frequency of the CADM.

5.10.2 CHIP SELECT LOGIC

The Chip Select ($\overline{CS}$) signal is generated by the Am29806 comparator. Address lines A1-A8 from the system bus are fed to the A- inputs of the Am29806. This device has internal pull-up resistors on the comparator B-inputs for easy connection to SPST switches. The comparator function is defined by:

$$\overline{E}_{OUT} = \overline{(A0 \odot B0)(A1 \odot B1)(A2 \odot B2)\ldots(Ai \odot Bi)G}$$

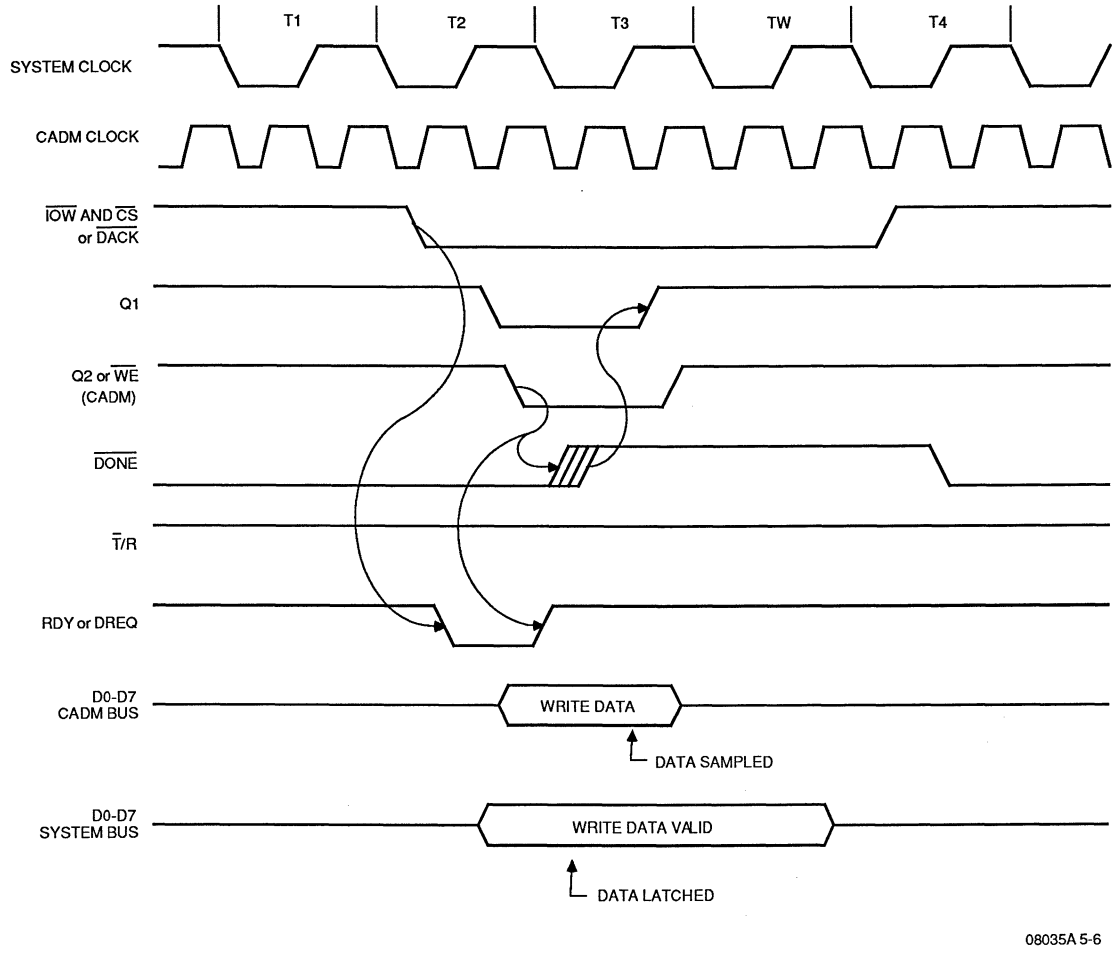


Figure 5-6. Am95C85–IBM PC/XT/AT Interface Write Timing

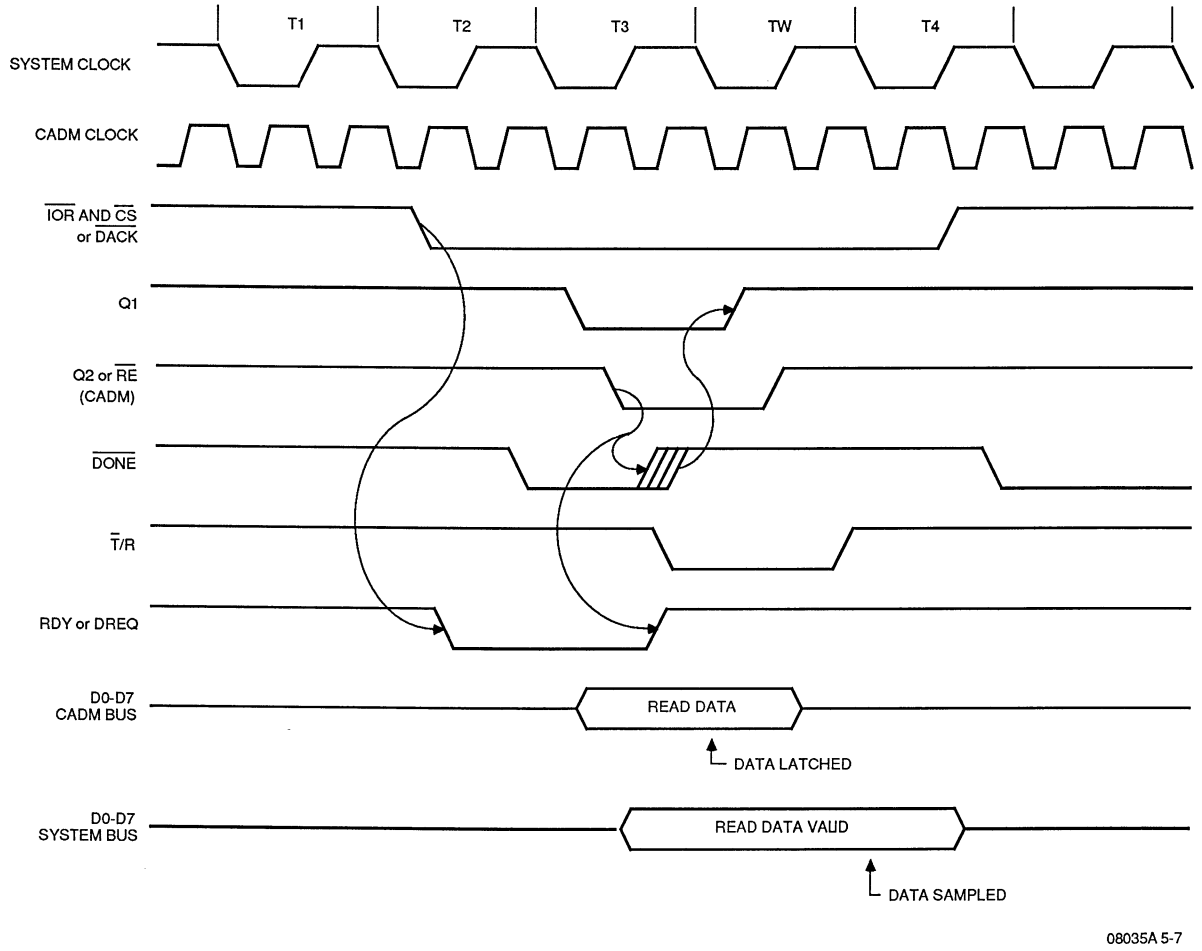


Figure 5-7. Am95C85–IBM PC/XT/AT Interface Read Timing

As seen from this equation $\overline{E}_{OUT}$ is qualified by $\overline{G}$, the enable input to the comparator. In this interface, $\overline{G}$ is provided by gating $\overline{IOR}$, $\overline{IOW}$, A_G, and AEN from the system, to avoid spurious $\overline{E}_{OUT}$ active signals caused by a memory access that matches the address of this input/output port. The $\overline{CS}$ to the CADMs is generated by ANDing the $\overline{E}_{OUT}$ address qualifier from the Am29806 and the DMA Acknowledge from channel 1 ($\overline{DACK1}$) of the DMA controller, i.e.,

$$\overline{CS} = \overline{E}_{OUT} \cdot \overline{DACK1}$$

Gating the $\overline{DACK1}$ signal provides an active $\overline{CS}$ during DMA transfers of data to and from the CADM.

5.10.3 GENERATING THE READY SIGNAL

The I/O Channel Ready (I/O CH RDY) input on the IBM PC/XT/AT must not be held inactive (LOW) for more than 2.5 us. This requirement of the IBM PC (XT or AT) dictates the generation of the Ready signal. Also to maximize system performance, each I/O access should insert the smallest possible number of Wait States. The I/O CH RDY signal is normally held active (HIGH) and is forced inactive (LOW) when $\overline{CS}$ goes active. This is then held LOW until Q2 goes LOW, i.e., a valid read/write is available for the CADMs at which point the I/O CH RDY is driven to the active (HIGH) state. This is shown in Figures 5-6 and 5-7.

The hardware implementation of RDY is shown in Figure 5-4. The normally HIGH output from Q4 is clocked LOW when $\overline{CS}$ goes active, i.e., a valid Chip Select is available to the CADMs. This places the processor in a Wait state. If $\overline{DONE}$ is active at this point, the read/write is clocked through D1 and D2 as explained earlier. When Q2 goes LOW it forces Q4 HIGH thus releasing the processor from the Wait state.

If the $\overline{DONE}$ is inactive (HIGH) when $\overline{CS}$ goes active, then the read/write from the system is temporarily prevented from being clocked through D1 and D2 until $\overline{DONE}$ goes active. If this condition occurs, extra Wait states are inserted. Refer to Section 5.9.

5.10.4 PAL DEVICE IMPLEMENTATION OF THE INTERFACE

Figure 5-8 shows the Am95C85 - IBM PC/XT/AT Interface using a PAL device. The PAL16R4 shown in Figure 5-8 serves as an alternative source to generate the $\overline{RE}$, $\overline{WE}$, C/$\overline{D}$ and CLK, thus eliminating some of the discrete logic. The difference of the PAL device implementation as compared to the discrete implementation lies in the manner in which D1 is clocked. The PAL device can only use the rising edge of clock; flip-flop D1 is clocked by the rising edge. Since $\overline{DONE}$ is forced HIGH by the rising edge of clock, output Q1 will go HIGH on the next rising edge thus adding an extra clock cycle to the CADM read/write active pulse. Figure 5-9 is a listing of the PAL device equations.

5.11 Am95C85 (CADM) INTERFACE TO AN 8086 PROCESSOR

The interface of the Am95C85 CADM to an 8086 processor is shown in Figure 5-10 combined with either Figure 5-2 or Figure 5-3. If 16 or fewer CADMs are used, Figure 5-10 is combined with Figure 5-2. If more than 16 CADMs are used in the application, Figure 5-10 is combined with Figure 5-3. This discussion deals with the generation of the hardware interface signals.This interface consists of three separate data and control buses that can be isolated from each other. The three buses are:

1. The System Bus
2. The CADM Bus
3. The Local CADM Bank Bus

The Write and Read timing diagrams for this interface are shown in Figures 5-11 and 12. The clock mentioned in this discussion is the CADM clock unless otherwise stated.

This section discusses in detail how and why each CADM interface control signal is generated. Because of the unique nature of this device, some of the control signals have special timing requirements.

5.11.1 SYNCHRONIZING THE READ AND WRITE SIGNALS

Whereas, most peripherals accept asynchronous control signals, the Am95C85 requires that its Read ($\overline{RE}$) and Write ($\overline{WE}$) inputs be synchronized with the rising edge of its clock. The Read ($\overline{RE}$) and Write ($\overline{WE}$) signals from the 8086 are ANDed to form one read/write signal and qualified by chip select from the decoder. This signal is then gated by the $\overline{DONE}$ control output from the CADM to ensure that the CADM has completed the previous operation before any further access is allowed. At this point we shall assume that the $\overline{DONE}$ output from the CADM is active (LOW). (The case when $\overline{DONE}$ is inactive (HIGH) when $\overline{RD}/\overline{WR}$ goes active

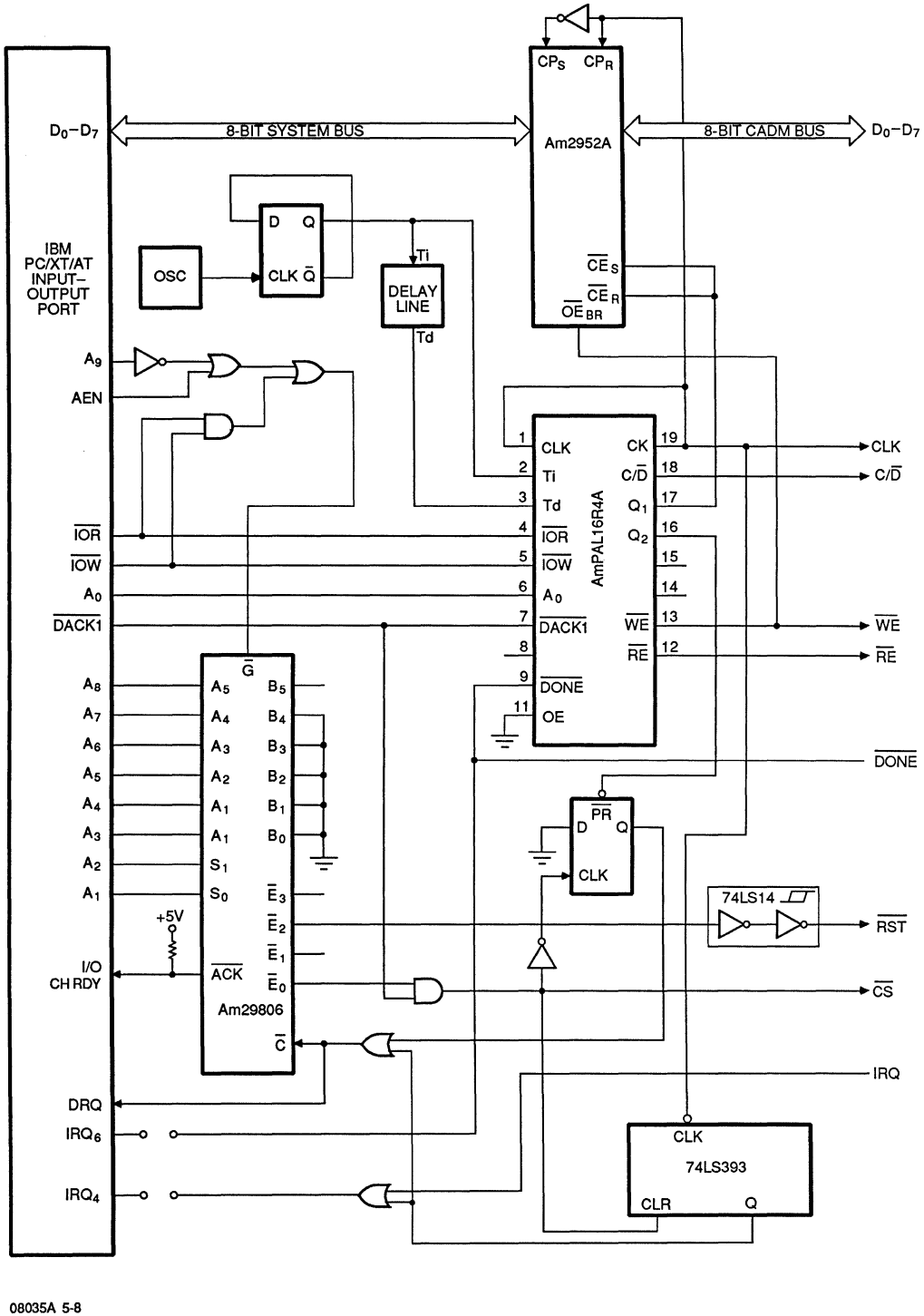


Figure 5-8. Am95C85–IBM PC/XT/AT Interface Using AmPAL16R4A

(LOW) is dealt with separately, in this discussion).

The falling edge of clock following the read/write being active at the D1 input passes the signal to Q1. The next rising edge generates the read/write signal for the CADMs which is then appropriately gated with the $\overline{RD}/\overline{WR}$ from the processor to separate the synchronized $\overline{RE}$ and $\overline{WE}$ signals. After the CADMs receive the read/write signal, they 3-state the $\overline{DONE}$ signal (which has a 1 kohm pull-up to Vcc) on the next falling edge. On the subsequent rising edge, the CADMs force the $\overline{DONE}$ line to the inactive (HIGH) state to indicate that they are busy with the current device access.

The $\overline{DONE}$ now drives the D1 input to the HIGH state. Q1 will go HIGH on the next falling edge of clock. Flip-flop Q1 is clocked on the falling edge because $\overline{DONE}$ is stable in the HIGH state before the falling edge, whereas its state is undefined at the rising edge. The subsequent rising edge of clock drives Q2 HIGH thus de-activating read/write to the CADMs. This mechanism of generating the read/write signals for the CADMs meets two requirements:

1. The read/write signals to the CADMs shall meet the A.C. specifications of set up time with respect to CADM clock.

2. The read/write signals shall be active (LOW) for at least two clock cycles. This scheme guarantees that these signals are active for exactly two clock cycles, for any operating frequency of the CADM.

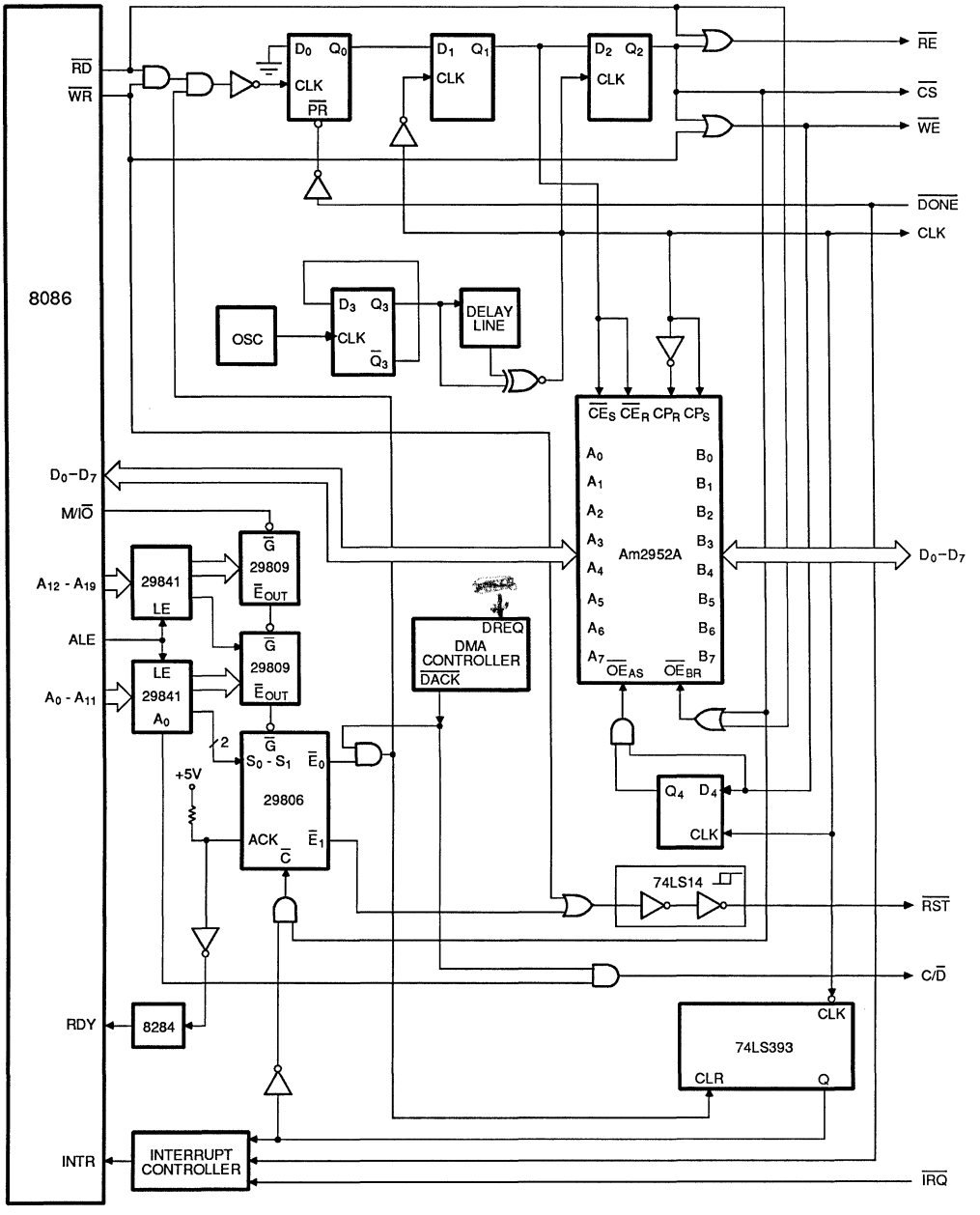
5.11.2 CHIP SELECT LOGIC

The Chip Select ($\overline{CS}$) signal is primarily generated by two Am29809, 9-bit comparators, and one Am29806. These devices have internal pull-up resistors on the comparator B inputs for easy

---

```
                                                    PAL DESIGN SPECIFICATION
IBM PC AT - Am95C85 INTERFACE                       SAROSH VESUNA 3-11-86
ADVANCED MICRO DEVICES
CLK   TIN   TD    /IOR   /IOW   /DONE   A0   /DACK   NC    GND
/OE   /RE   /WE    Q2     Q1     NC     NC     CD    CK    VCC

/Q1 := IOR *DONE + IOW *DONE

/Q2 := /Q1

RE    = /Q2*IOR

WE    = /Q2*IOW

/CD = /A0 + DACK

/CK = TIN*/TD + /TIN*TD

FUNCTION TABLE
CLK    TIN     TD   /IOR   /IOW   /DONE    A0   /DACK   /RE    /WE     Q2    Q1    CD    CK

 P      H      L     H      H      L       L      H      H      H      H     H     L     L
 X      H      H     H      L      L       L      H      H      H      H     H     L     H
 C      H      H     H      L      L       L      H      H      H      H     L     L     H
 X      L      H     H      L      L       L      H      H      H      H     L     L     L
 X      L      L     H      L      L       L      H      H      H      H     L     L     H
 C      L      L     H      L      L       L      H      H      L      L     L     L     H
 X      H      L     H      L      H       L      H      H      L      L     L     L     L
 X      H      H     H      L      H       L      H      H      L      L     L     L     H
 C      H      H     H      L      H       L      H      H      L      L     H     L     H
 X      L      H     H      L      H       L      H      H      L      L     H     L     L
 X      L      L     H      L      H       L      H      H      L      L     H     L     H
 C      L      L     H      L      H       L      H      H      H      H     H     L     H

DESCRIPTION:
THE ABOVE FUNCTION TABLE TESTS THE CADM WRITE CYCLE, WITH THE CPU PROVIDING
THE WRITE AND CHIP SELECT.
```

Figure 5-9. PAL Device Equations for CADM–IBM Interface Ready Circuit



08035A 5-10

Figure 5-10. Am95C85–8086 Interface

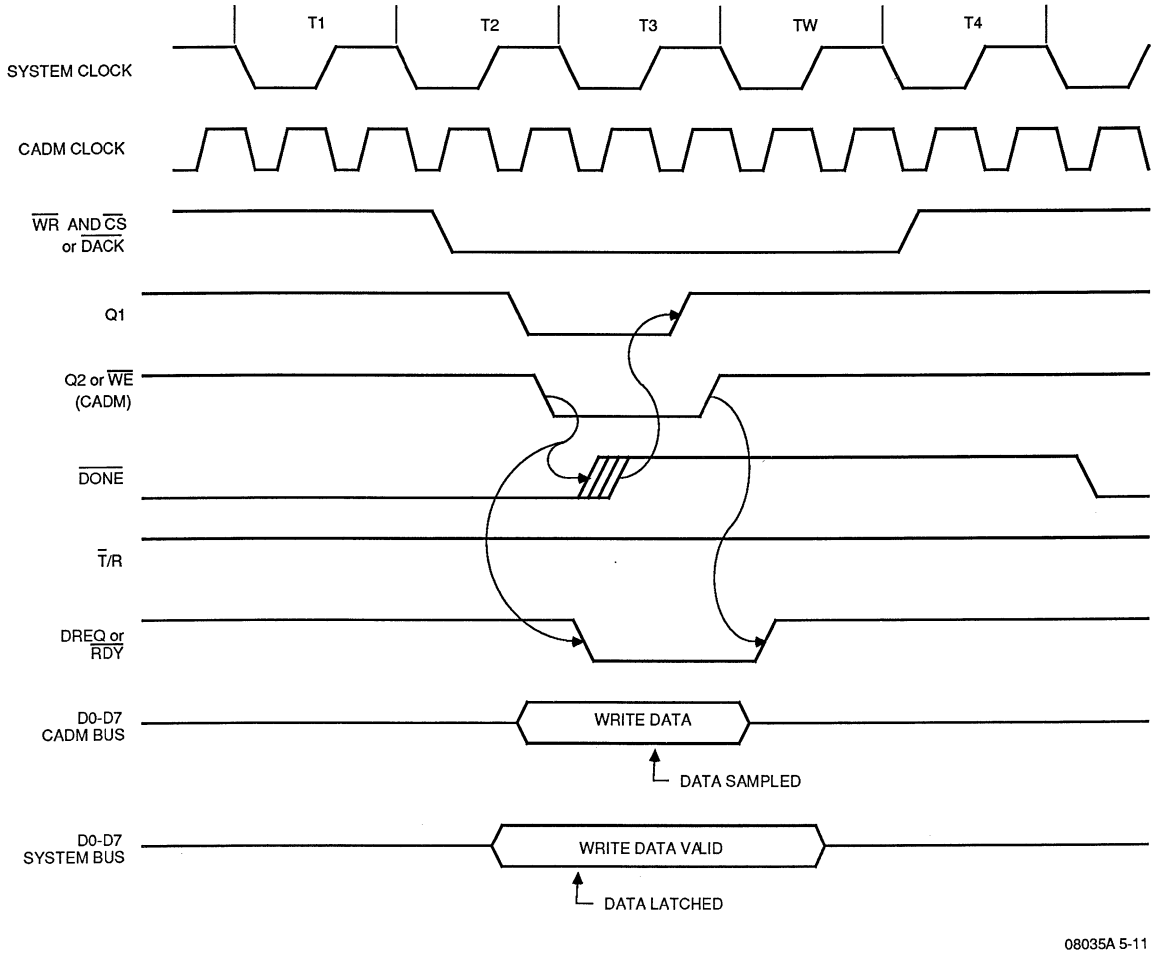


Figure 5-11. Am95C85–8086 Interface Write Timing

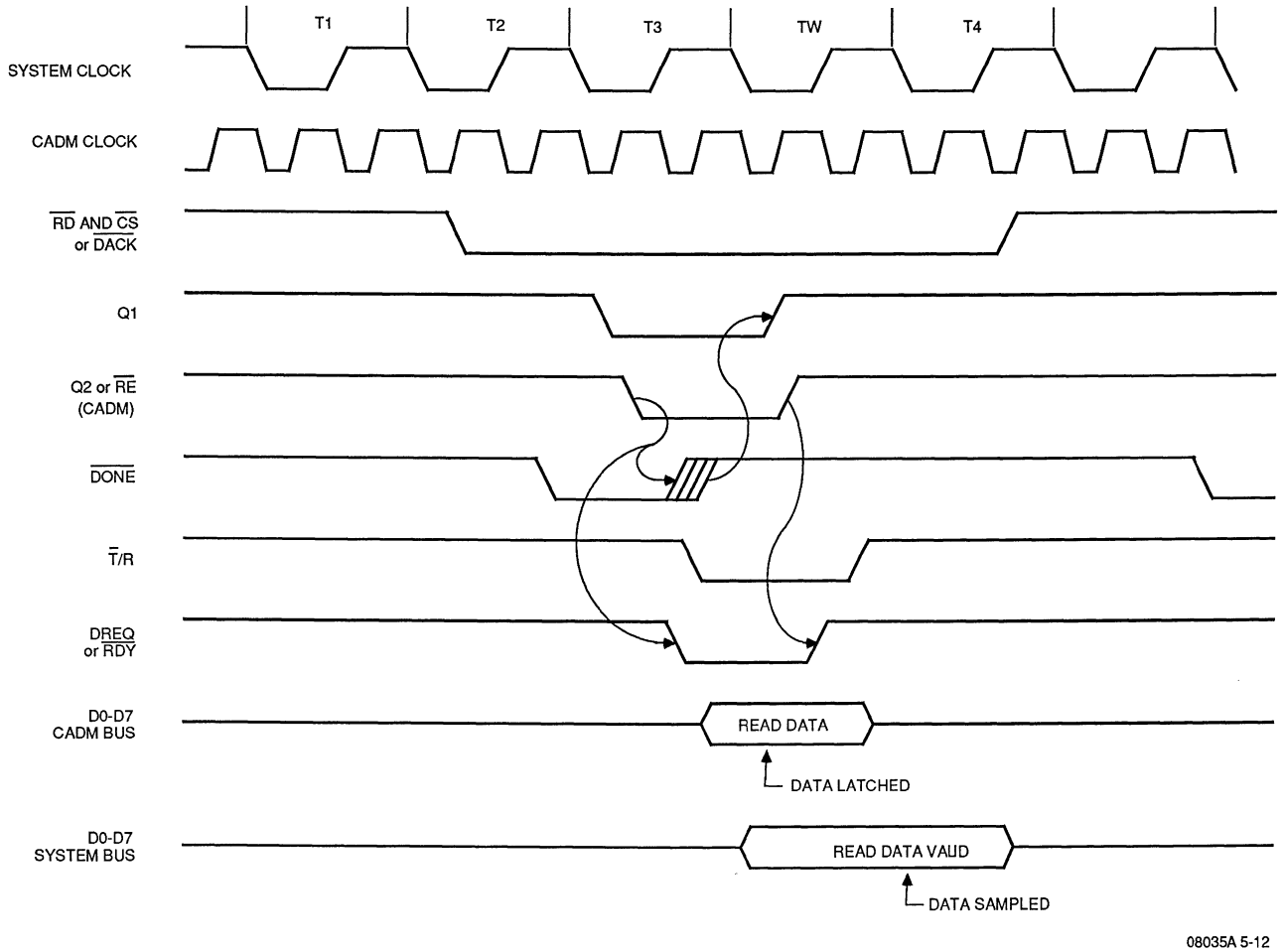


Figure 5-12. Am95C85–8086 Interface Read Timing

connection to SPST switches to ground selected inputs. The address lines are latched by the Am29841s on the falling edge of ALE. The comparator function is defined by:

$$\overline{E}_{OUT} = \overline{(A0 \odot B0)(A1 \odot B1)(A2 \odot B2)\ldots(Ai \odot Bi)\overline{G}}$$

As seen in this equation, $\overline{E}_{OUT}$ is qualified by $\overline{G}$, the enable input to the comparator. In this interface, $\overline{G}$ is provided by M/$\overline{IO}$ from the processor so as to avoid spurious $\overline{E}_{OUT}$ active signals caused by a memory access that matches the address of this input/output port. The $\overline{CS}$ is generated by ANDing the $\overline{E}_0$ address qualifier from the Am29806 with the DMA Acknowledge from channel 1 of the DMA controller, i.e.,

$$\overline{CS} = \overline{E}_{OUT} \cdot \overline{DACK}$$

Gating the $\overline{DACK}$ signal provides an active $\overline{CS}$ during DMA transfers of data to and from the CADM. This $\overline{CS}$ signal is gated with the read/write and passed through the synchronizing logic to obtain a synchronized $\overline{CS}$ for the CADMs from Q2.

E1 is selected when there is a valid address on A1–A18 and A1 is HIGH. This provides an alternative source of a hardware reset for the CADMs.

5.11.3 GENERATING THE READY SIGNAL

To maximize system performance, each I/O access should insert the smallest possible number of WAIT States. The Ready signal is normally held inactive (LOW) and is forced active (HIGH) when Q2 goes active, i.e., a valid read/write is available to the CADMs. This is held HIGH until Q2 goes inactive as shown in Figures 5-11 and 12.

The hardware implementation of RDY is shown in Figure 5-10. If $\overline{DONE}$ is active when $\overline{RD}/\overline{WR}$ goes active, the read/write is clocked through D1 and D2 as explained earlier. When Q2 goes LOW it forces C LOW on the Am29806. If an address match exists (i.e., E_0 is active), the $\overline{ACK}$ output goes LOW. The invertor at the RDY input of the 8284 causes it to go active (HIGH) thus releasing the processor from the Wait state.

If the $\overline{DONE}$ is inactive (HIGH) when $\overline{RD}/\overline{WR}$ goes active, then the read/write from the system is temporarily prevented from being clocked through D1 and D2 until $\overline{DONE}$ goes active. If this condition occurs, extra Wait states are inserted. Refer to Section 5-9.

5.12 Am95C85 (CADM) INTERFACE TO AN MC68000 PROCESSOR

The interface of the Am95C85 CADM to a MC68000 processor is shown in Figure 5-13 combined with either Figure 5-2 or Figure 5-3. If 16 or fewer CADMs are used, Figure 5-13 is combined with Figure 5-2 If more than 16 CADMs are used in the application, Figure 5-13 is combined with Figure 5-3. This discussion deals with the generation of the hardware interface signals. This interface consists of three separate data and control busses that can be isolated from each other. The three busses are:

1. The System Bus
2. The CADM Bus
3. The Local CADM Bank Bus

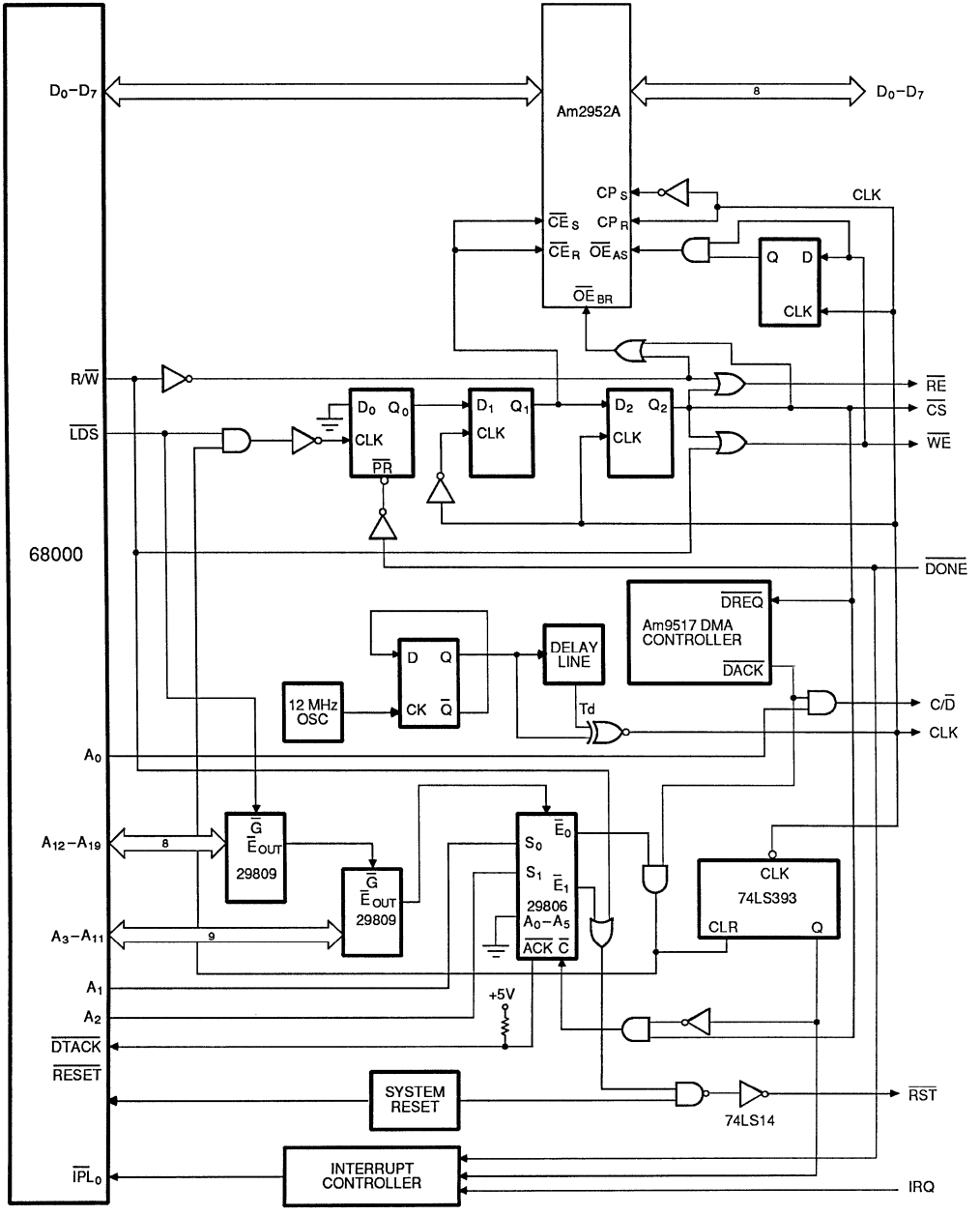
The Write and Read timing diagrams for this interface are shown in Figures 5-14 and 15. The clock mentioned in this discussion is the CADM clock unless otherwise stated.

This section discusses in detail how and why each CADM interface control signal is generated. Because of the unique nature of this device, some of the control signals have special timing requirements.

5.12.1 SYNCHRONIZING THE READ AND WRITE SIGNALS

Whereas, most peripherals accept asynchronous control signals, the Am95C85 requires that its Read ($\overline{RE}$) and Write ($\overline{WE}$) inputs be synchronized with the rising edge of its clock. The $\overline{LDS}$ (Lower Data Strobe) is gated by the $\overline{DONE}$ control output from the CADM to ensure that the CADM has completed the previous operation before any further access is allowed. At this point we shall assume that the $\overline{DONE}$ output from the CADM is active (LOW). The case when $\overline{DONE}$ is inactive (HIGH) when $\overline{R/W}$ goes active is dealt with separately, in this discussion.

The falling (trailing) edge of clock following the $\overline{LDS}$ being active at the D1 input passes the signal to Q1. The next rising edge generates the read/write signal for the CADMs which is then appropriately gated to separate the $\overline{RE}$ and $\overline{WE}$ signals. After the CADMs receive the synchronized read/write signal, they 3-state the $\overline{DONE}$ signal (which has a 1 kohm pull-up to V_{cc}) on the next falling edge. On the subsequent rising edge, the CADMs force the $\overline{DONE}$ line to the inactive (HIGH) state to indicate that they are busy



08035A 5-13

Figure 5-13. Am95C85–68000 Interface

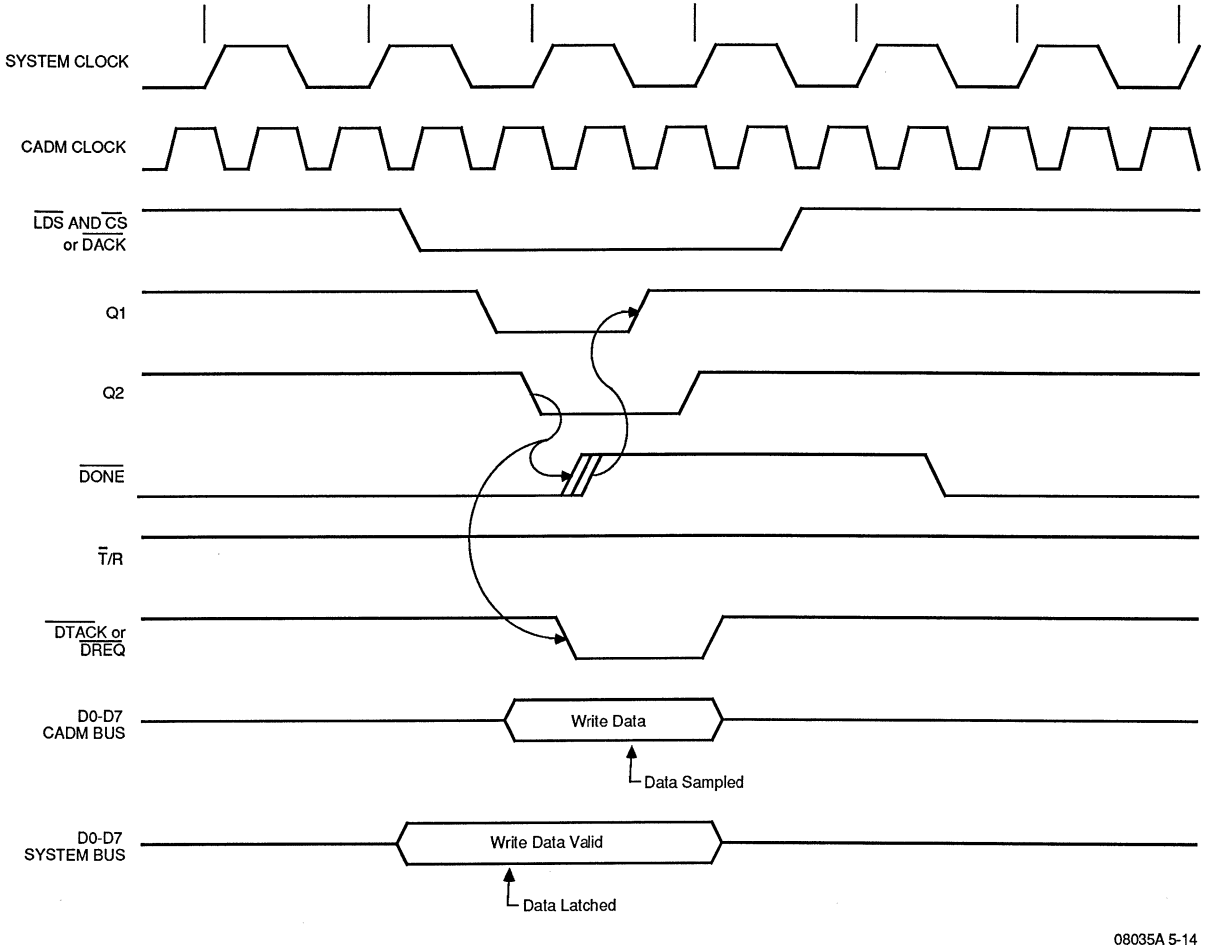


Figure 5-14. Am95C85–68000 Interface Write Timing

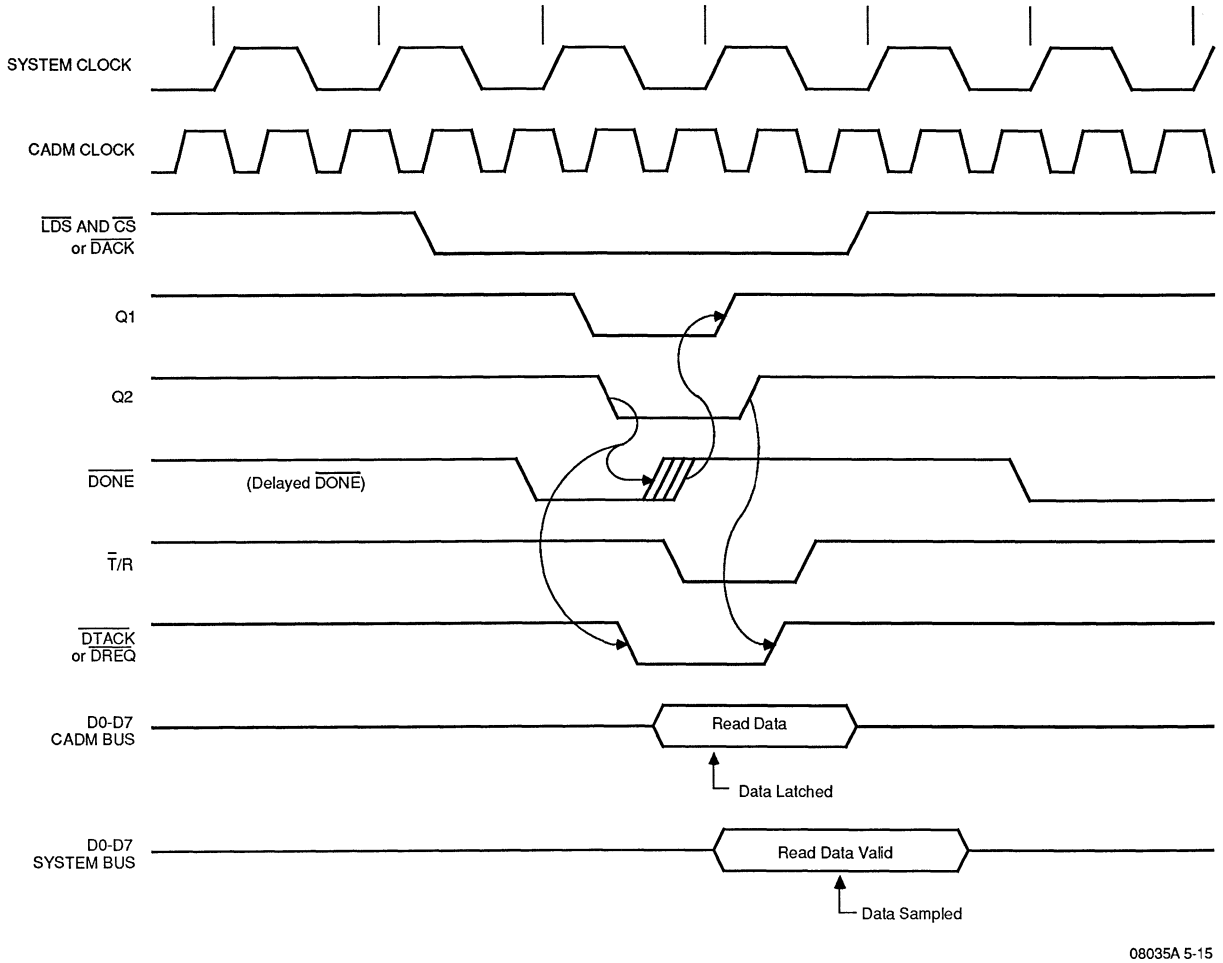


Figure 5-15. Am95C85–8086 Interface Read Timing

with the current device access.

The $\overline{DONE}$ now drives the D1 input to the HIGH state which triggers the transition of Q1 HIGH on the next falling edge of clock. Flip-flop Q1 is clocked on the falling edge because $\overline{DONE}$ is stable in the HIGH state before the falling edge, whereas its state is undefined at the rising edge. The subsequent rising edge of clock drives Q2 HIGH thus deactivating read/write to the CADMs. This mechanism of generating the read/write signals for the CADMs satisfies two requirements:

1. The read/write signals to the CADMs shall meet the A.C. specifications of set up time with respect to CADM clock.

2. The read/write signals shall be active (LOW) for at least two clock cycles. This scheme guarantees that these signals are active for exactly two clock cycles, for any operating frequency of the CADM.

5.12.2 CHIP SELECT LOGIC

The Chip Select ($\overline{CS}$) signal is primarily generated by two Am29809, 9-bit comparators, and the Am29806. These devices have internal pull-up resistors on the comparator B-inputs for easy connection to SPST switches to ground selected inputs. The comparator function is defined by:

$$\overline{E}_{OUT} = \overline{(A0 \odot B0)(A1 \odot B1)(A2 \odot B2)...(Ai \odot Bi)G}$$

As seen from the above equation $\overline{E}$OUT is qualified by $\overline{G}$, the enable input to the comparator. In this interface, $\overline{G}$ to the first Am29809 is provided by $\overline{LDS}$ from the processor to avoid spurious $\overline{E}$OUT active signals. The $\overline{CS}$ to the CADMs is generated by ANDing the $\overline{E}_0$ address qualifier from the Am29806 and the DMA Acknowledge from channel 1 of the DMA controller, i.e.,

$$\overline{CS} = \overline{E}_{OUT} \cdot \overline{DACK}$$

Gating the $\overline{DACK}$ signal provides an active $\overline{CS}$ during DMA transfers of data to and from the CADM. The E_0 output of the Am29806 is selected when address lines A1 and A2 are both LOW.

$\overline{E}_1$ is selected when there is a valid address on A3-A19, A1 is HIGH, and A2 is LOW. This provides an alternative source of a hardware reset for the CADMs.

5.12.3 GENERATING THE READY SIGNAL

To maximize system performance, each I/O access should insert the smallest possible number of Wait States. The $\overline{DTACK}$ (Data Transfer Acknowledge) signal is normally held inactive(HIGH) and is forced active (LOW) when Q2 goes active, i.e., a valid read/write is available to the CADMs. This is held LOW until Q2 goes inactive (HIGH), as shown in Figure 5-13.

The hardware implementation of RDY is shown in Figure 5-13. If $\overline{DONE}$ is active when $\overline{LDS}$ goes active, the read/write is clocked through D1 and D2 as explained earlier. When Q2 goes LOW, it forces $\overline{C}$ LOW on the Am29809. If an address match exists (i.e., $\overline{E}_{OUT}$ is active), the $\overline{ACK}$ output goes LOW thus releasing the processor from the Wait state.

If the $\overline{DONE}$ is inactive (HIGH) when LDS goes active, then the read/write from the system is temporarily prevented from being clocked through D1 and D2 until $\overline{DONE}$ goes active. If this condition occurs, extra Wait States are inserted. Refer to Section 5-9.

APPENDIX A

Am95C85 CADM SORT PERFORMANCE BENCHMARK SUMMARY

BENCHMARK SUMMARY

One measure of the performance of the Am95C85 Content Addressable Data Manager (CADM) device is it's ability to sort data. When compared to the QSort, QPoint, and Tree sort software algorithms, the CADM, simulated at 16 MHz, performed as follows:

- Up to 50-times faster than the VAX 11/785
- Up to 116-times faster than the Valid ScaldStar Workstation
- Up to 154-times faster than the IBM PC-AT
- Up to 424-times faster than the IBM PC-XT

(Improvement Factor = CADM Load & Sort Time / Software Sort Time)

	CADM Performance Improvement Factors	
Computer	**Minimum**	**Maximum**
VAX 11/785	12.49	50.16
VALID ScaldStar	43.77	116.41
IBM PC-AT	25.43	154.37
IBM PC-XT	77.52	424.43

Figure 1 illustrates the performance ranges of the Am95C85 CADM with respect to each of the four computers for 100-, 400-, and 1000-record file sizes.

This comparison represents real-world reflections of expected performance gains of the CADM since the time required to both load and sort data by the CADM is compared to the time required to sort data in software.

Furthermore, performance improvement multiples are delineated for all combinations of six data types, three file sizes, three sort algorithms and four processing machines.

BENCHMARK DESCRIPTION

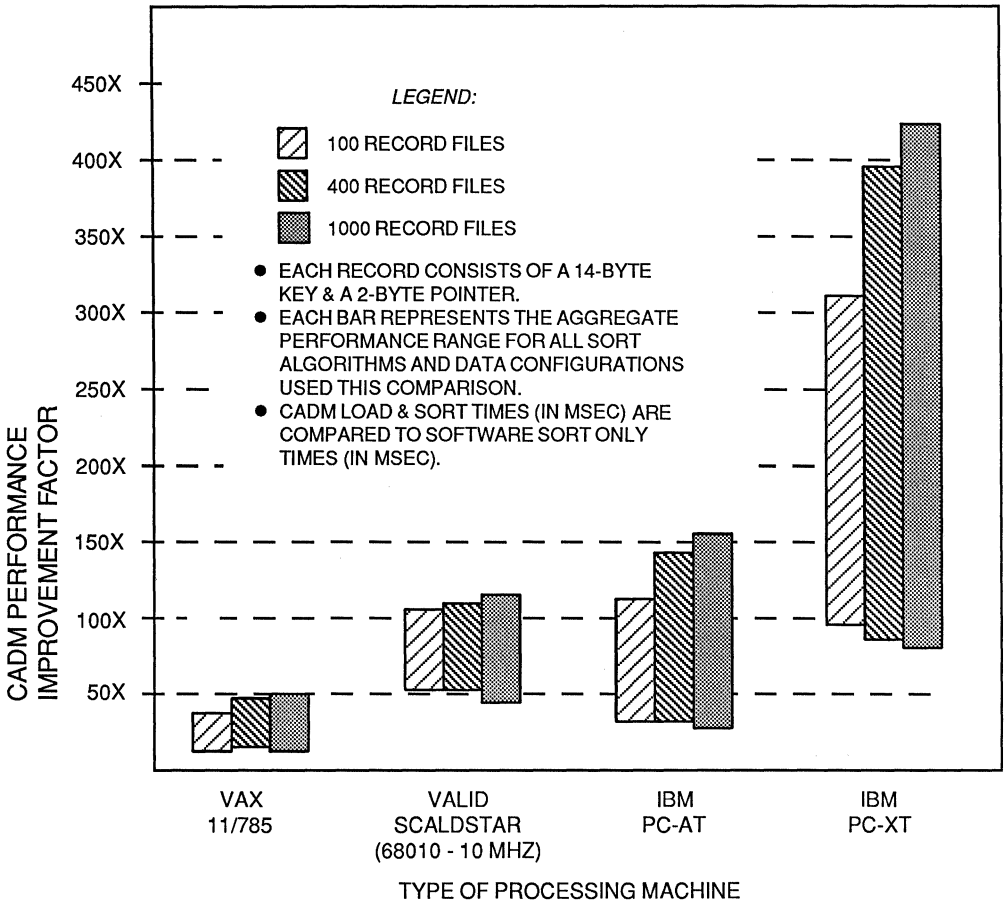
The Am95C85 Content Addressable Data Manager (CADM) is a unique CMOS peripheral device designed to perform high-performance data sorting, searching and updating. The device is capable of accelerating by orders-of-magnitude many of the time consuming, repetitive data manipulation tasks which are found in operating systems and application level software.

The purpose of this benchmark is to document these performance advantages in an objective framework so that greater understanding of the device's capabilities may be obtained. In designing the analysis, care has been taken to represent performance data in a manner as close to actual system conditions as possible. To this end, the comparison of the CADM to software sort performance is intended to represent an "apples to apples" reflection of the performance advantages of the device.

While the CADM is also able to perform content addressable searching, insertions, deletions and other data manipulative tasks, the sort operation has been selected as the comparative element since this task is germane to most operating systems and applications software and must be performed by the CADM prior to searching and other operations.

Types of Input Data[1]	File Sizes	Sort Algorithms[2]	Processing Machines
Random	100 Records	QSort	VAX 11/785
Reverse sorted	400 Records	QPoint	Valid ScaldStar
Identical	1000 Records	Tree sort	IBM PC-AT
Pre-sorted			IBM PC-XT
98% Pre-sorted			
90% Pre-sorted			

Note: All software sort times presented in this analysis are accurate to within +/- 2.5%.
[1]All records contained in the input data files consist of a 14-byte key field plus a 2-byte pointer field.
[2]The sort algorithms used are given in the CADM Benchmark publication.



Note: All computer sorting is conducted in main memory, without disk accesses.

08035A A-1

Figure A-1. Sort Performance CADM vs. Standard Computers

The sort performance of the Am95C85 was compared to three common software sort algorithms each run on four industry standard processing machines. The comparison assumes that the data to be sorted is resident in main memory and does not involve disk accesses. The benchmark compares the time required for the data to be loaded into the Am95C85 from main memory and then sorted versus the time needed for the computer to perform the software sort alone. In both cases, the analysis begins with the data to be sorted located in system memory and ends with a sorted file ready for the next task. The processing machines and sort algorithms used are:

Processing Machines	Sort Algorithms
1. VAX 11/785	1. QPOINT
2. VALID SCALDSTAR (68010-10MHz)	2. QSORT
3. IBM PC-AT (80286-6MHz)	3. TREE SORT
4. IBM PC-XT (8088-4.8MHz)	

Sort programs were written in "C" based upon each of the three standard sort algorithms found in the literature (Wirth, Niklaus. Algorithms + Data Structures = Programs, Pentice-Hall, Englewood Cliffs, N.J., 1976.). These algorithms are available in the CADM Benchmark publication.

Six types of ASCII data files were sorted. These are:

- Random data (3 sets)
- Identical data
- Reverse sorted data
- Pre-sorted data
- 98% pre-sorted data (2 sets)
- 90% pre-sorted data (2 sets)

From each of these data types, three lengths of input files were generated: 100 records, 400 records and 1000 records (where each record consists of a 14-byte key field plus a 2-byte pointer field). Thus, 30 total input files of varying length and data type were sorted using three different sort algorithms. Each combination of data type, length and sort algorithms were run on four processing machines. The same input files were also loaded and sorted by a 16 MHz CADM sort simulation routine which represented an array of 16 CADM devices in cascade.

METHODOLOGY

Input Files

Ten input files were sorted, with the sort time results categorized into six groups based on type of data. Performance results were then charted based on these six data categories. The 10 ASCII input files and their resulting data categories are:

Input Files	Resultant Data Categories
3 Random data files	Random data (average of 3 files)
1 Identical data file	Identical data
1 Reverse-sorted file	Reverse-sorted data
1 Pre-sorted file	Pre-sorted data
2 98% pre-sorted files	98% pre-sorted data (average of 2 files)
2 90% pre-sorted files	90% pre-sorted data (average of 2 files)
10 Total	

Type of input data file	Description
Random	Data in each key field has no specific pattern or sequence.
Identical	Data in each key field is identical.
Reverse-sorted	Data records are pre-sorted in reverse order based on key field.
Pre-sorted	Data records are pre-sorted based on key field.
98% pre-sorted	Data records are pre-sorted. Last 2% of file is then removed and scattered randomly throughout the remainder of the file.
90% pre-sorted	Data records are pre-sorted. Last 10% of file is then removed and scattered randomly throughout the remainder of the file.

In preparing the analysis, each of the 10 input files were run independently and the results recorded. For the sake of simplicity and graphic illustration, the results of the three random data files were averaged as were the two 98% pre-sorted and the

two 90% pre-sorted files. Figures 2, 3, 4 and 5 demonstrate these averaged performance values of the CADM compared to sort algorithms run on the four processing machines used in this analysis. A complete listing of the results for each of the 10 input files, prior to averaging, is available in Appendix B.

Calculating Sort Times

The internal system clock was used in determining the time required for computer sorting. Conceptually, a source program, written in "C", was generated which called the appropriate sort algorithm. Immediately prior to the start of the sort, the system clock was polled and the time recorded by the program. Upon termination of the software sort, the time was immediately sampled again and compared to the start time. A simple subtraction then produced the time required by the computer to sort the given file using the given algorithm.

Maintaining Data Accuracy

System Clock Granularity

It is important to note that the granularity of the real-time clock within the computers can impose limits on the accuracy of sort times. In cases where the sort times are small (such as 100 record data files), the time required to complete a sort can approach the incremental graduations of the computer clock. The resulting inaccuracy would be particularly acute with the 55 msec granularity of the IBM PC-XT and PC-AT clocks.

To avoid this source of inaccuracy, multiple sort operations were consecutively performed and the total time recorded. This total was then divided by the number of sort passes involved to accurately determine the sort time. Precautions were also taken to assure that clock sampling times were not included in the sort time. As a result of these measures, the sort times performed by the four computers are accurate to within ± 2.5%.

Multi-User Systems

The VAX 11/785 and Valid ScaldStar provide for multi-user support by sharing CPU time among the users involved. To assure sort times were not inflated due to the time-sharing process, the comparison was made by measuring actual CPU time devoted to the sort process rather than comparing elapsed time. Each machine was dedicated exclusively to the sort calculations without competing with other time-shared tasks.

An interesting effect of multi-user systems occurs when using the Tree sort algorithm. As shown in Figures 2 and 3, the sort time required by the Tree sort for pre-sorted and nearly-sorted data is proportionately much greater than equivalent data types on the PC-XT and PC-AT. This is due to the mechanism of the Tree sort algorithm of allocating additional memory as each record is compared and sorted. Since requests for additional memory allocation in multi-user systems must be granted by the operating system, this added time is rightfully reflected in the Tree sort results.

SUMMARY

The sort comparison benchmark illustrates that very significant performance gains can be expected by the user with respect to standard software sorting routines. Effort has been made to assure the analysis was cast in a practical, user-oriented environment. Such conservatism is apparent since software sort times of the computers are compared to the time required by the CADM to both load data into the device from system memory and conduct the sort. This bias against the Am95C85 was included in order to represent an "apples-to-apples" comparison of traditional sort techniques in software to higher performance sorting in hardware.

The benchmark conducted an objective comparison involving four industry standard computers and three commonly-used sort algorithms on a spectrum of data types and file sizes. Every combination of the above elements were compared and the results illustrated both graphically and in tabular form.

CADM SORT TIMES VS. STANDARD COMPUTERS

(All sort times are in milliseconds)

	RANDOM DATA (set 1)	RANDOM DATA (set 2)	RANDOM DATA (set 3)	RANDOM DATA (avg.)	IDENTICAL DATA	REVERSE-SORTED DATA
Am95C85 CADM						
100 records	1.77	1.76	1.77	1.77	2.59	1.81
400 records	8.15	8.17	8.20	8.17	10.59	8.83
1000 records	23.63	23.60	23.51	23.58	26.67	27.37
VAX 11/785						
QP.100	40	37	37	38.0	42	27
QP.400	183	186	177	182.0	218	137
QP.1000	566	554	540	553.3	597	365
QS.100	51	49	50	50.0	61	35
QS.400	238	244	236	239.3	313	166
QS.1000	714	703	703	706.7	858	342
TR.100	62	61	62	61.7	77	62
TR.400	323	320	319	320.7	382	296
TR.1000	1015	1019	1014	1016.0	1069	824
VALID SCALDSTAR						
QP.100	141	134	133	136.0	150	97
QP.400	631	642	618	630.3	761	469
QP.1000	1847	1808	1818	1824.3	2021	1198
QS.100	186	180	181	182.3	218	127
QS.400	848	867	842	852.3	1112	592
QS.1000	2467	2434	2440	2447.0	2955	1466
TR.100	134	132	134	133.3	196	138
TR.400	663	660	659	660.7	982	684
TR.1000	1894	1891	1877	1887.3	2806	1933
IBM PC-XT						
QP.100	223	215	208	215.3	360	162
QP.400	1105	1032	991	1042.7	1885	785
QP.1000	3003	2920	2948	2957.0	5148	1985
QS.100	522	512	530	521.3	807	356
QS.400	2435	2517	2462	2471.3	4195	1573
QS.1000	7018	6908	6974	6966.7	11319	3729
TR.100	197	194	197	196.0	424	207
TR.400	1010	991	991	997.3	2242	1041
TR.1000	2986	2948	2986	2973.3	6578	3250
IBM PC-AT						
QP.100	76	74	70	73.3	128	55
QP.400	348	352	338	346.0	680	270
QP.1000	1037	1009	1023	1023.0	1862	696
QS.100	186	180	186	184.0	292	127
QS.400	854	888	877	873.0	1515	563
QS.1000	2494	2494	2490	2492.7	4117	1348
TR.100	68	68	68	68.0	153	72
TR.400	349	342	343	344.7	788	359
TR.1000	1009	1009	1009	1089.0	2329	1033

	PRE-SORTED DATA	98% PRE-SORTED DATA (set 1)	98% PRE-SORTED DATA (set 2)	98% PRE-SORTED DATA (avg.)
Am95C85 CADM				
100 records	1.63	1.63	1.63	1.626
400 records	6.64	6.75	6.74	6.743
1000 records	16.87	17.27	17.27	17.266
VAX 11/785				
QP.100	27	27	27	27.0
QP.400	136	137	135	136.0
QP.1000	363	364	364	364.0
QS.100	32	32	32	32.0
QS.400	153	151	152	151.5
QS.1000	399	401	403	402.0
TR.100	62	62	62	62.0
TR.400	304	310	308	309.0
TR.1000	834	862	865	863.5
VALID SCALDSTAR				
QP.100	96	96	96	96.0
QP.400	467	461	461	461.0
QP.1000	1194	1195	1192	1193.5
QS.100	117	117	117	117.0
QS.400	550	542	542	542.0
QS.1000	1371	1372	1375	1373.5
TR.100	139	139	139	139.0
TR.400	687	709	709	709.0
TR.1000	1949	2010	2003	2006.5
IBM PC-XT				
QP.100	162	162	162	162.0
QP.400	771	757	771	764.0
QP.1000	1985	1996	1996	1996.0
QS.100	293	299	299	299.0
QS.400	1316	1280	1280	1280.0
QS.1000	3069	3085	3096	3090.5
TR.100	208	207	208	207.5
TR.400	1060	1078	1078	1078.0
TR.1000	3058	3344	3344	3344.0
IBM PC-AT				
QP.100	55	55	55	55.0
QP.400	270	265	265	265.0
QP.1000	696	696	696	696.0
QS.100	105	107	106	106.5
QS.400	475	462	470	466.0
QS.1000	1147	1128	1147	1137.5
TR.100	70	73	70	71.5
TR.400	359	372	371	371.5
TR.1000	1050	1078	1078	1078.0

	90% PRE-SORTED DATA (set 1)	90% PRE-SORTED DATA (set 2)	90% PRE-SORTED DATA (avg.)
Am95C85 CADM			
100 records	1.65	1.65	1.654
400 records	7.02	7.02	7.019
1000 records	18.39	18.38	18.389
VAX 11/785			
QP.100	27	27	27.0
QP.400	137	137	137.0
QP.1000	373	374	373.5
QS.100	32	32	32.0
QS.400	156	156	156.0
QS.1000	419	421	420.0
TR.100	62	62	62.0
TR.400	317	716	316.5
TR.1000	909	906	907.5
VALID SCALDSTAR			
QP.100	97	97	97.0
QP.400	468	468	468.0
QP.1000	1229	1227	1228.0
QS.100	119	118	118.5
QS.400	556	556	556.0
QS.1000	1440	1439	1439.5
TR.100	140	140	140.0
TR.400	715	717	716.0
TR.1000	2051	2061	2056.0
IBM PC-XT			
QP.100	162	162	162.0
QP.400	785	771	778.0
QP.1000	2079	2051	2065.0
QS.100	304	298	301.0
QS.400	1335	1358	1346.5
QS.1000	3399	3399	3399.0
TR.100	207	212	209.5
TR.400	1078	1096	1087.0
TR.1000	3388	3388	3388.0
IBM PC-AT			
QP.100	55	55	55.0
QP.400	270	270	270.0
QP.1000	718	718	718.0
QS.100	107	107	107.0
QS.400	493	490	491.5
QS.1000	1238	1238	1238.0
TR.100	72	72	72.0
TR.400	371	371	371.0
TR.1000	1105	1092	1098.5

The International Standard of Quality guarantees a 0.05% AQL on all electrical parameters, AC and DC, over the entire operating range.

INT·STD·500

ADVANCED MICRO DEVICES, INC.
901 Thompson Place
P.O. Box 3453
Sunnyvale,
California 94088
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL FREE
(800) 538-8450

Order #08035A

IH-MU-7.5M-7/86-0