SIEMENS

Microcomputer Components

SAB 82257
High-Performance DMA Controller for 16-Bit Microcomputer Systems

User's Manual 09.86

Siemens Aktiengesellschaft

Straightforward ordering with the catalog "Siemens Components Service, Preferred Products".

Every year, a revised edition of the SCS catalog on Preferred Products (about 800 pages) is published. This catalog comprises preferred products of the entire Siemens components program including their main technical specs.

Orders for components as well as for the above mentioned catalog should be directed to your nearest Siemens Office, Dept. VB, or Distributor.

Published by Siemens AG, Bereich Bauelemente, Produkt-Information, Balanstraße 73, D-8000 München 80

Contents

Contents

Introduction

1 Introduction

1.1 Advantage of DMA

DMA controllers (DMACs) are dedicated to the task of controlling high-speed data block transfers independent of the CPU. Typically, data is transferred between memory and I/O or vice versa, although a few DMA controllers have capabilities to perform other types of transfers normally done by the CPU.

The SAB 82257, for example, can transfer bytes or words between memories or between I/Os on four independent high-speed channels.

In systems without DMA, data transfers must pass through the CPU and must therefore be implemented in software. This normally involves the execution of an instruction sequence for inputting, outputting, and tracking each byte/word of data in the block to be transferred.

Figure 1 illustrates the minimum sequence of instructions that must be fetched from memory and executed by conventional CPUs to transfer a data block by one byte at a time. In fact, most CPUs require much more instructions than shown here.

One disadvantage of this method is that CPU transfers turn out to be comparatively slow and tie up the CPU for a long time. Another disadvantage is that the response time (startup time for the first byte) is usually also slow, because the I/O device typically uses interrupts to signal its readiness and the CPU interrupt service routine causes a significant time lag when transferring the first byte.

A DMA controller performs direct transfers between the source and destination of data without involving the CPU. All the steps illustrated in figure 1 are carried out independently.

In a memory-to-I/O transfer, for example, the starting address in the memory and the length of the block to be transferred are written into the DMA controller by the CPU. This is done prior to the transfer. The DMA controller quickly takes over control and starts to transfer data when enabled by the CPU and the I/O device's ready line becomes active. In most cases the CPU is in idle state during this process. Upon completion of the transfer, the DMA controller informs the CPU and releases control. DMACs are used, therefore, when one or more of the following conditions or requirements are present:

- The CPU is busy with too many I/O operations to perform its other tasks properly.
- The transfer must be faster than the CPU could perform it.
- The transfer response time (startup) must be shorter than the CPU could conveniently provide.

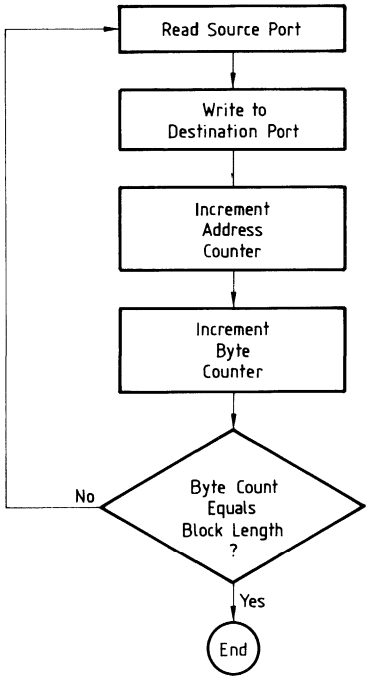
Small and low-performance systems generally run without DMA. Medium-performance systems may also be designed without DMA if the CPU can handle transfers fast enough and still do its other work.
Whenever systems require fast transfers or fast response, DMA controllers are strong candidates for performance enhancement. Not only do they perform the transfers faster than CPUs (the SAB 82257, for example, provides high-speed transfers of up to 8 Megabytes per second), their response time is also significantly better. Here are a few examples where DMA is often the best choice to make:

- hard disk and floppy disk controllers,
- scanning operations, such as CRT I/O,
- data acquisition,

- memory-to-memory transfers,
- backup storage (I/O to I/O),
- fiber optic links,
- block transfers in networking, multiprocessing, or multiprogramming.

The trade-off for this speed is that the CPU typically remains idle and lacks full or partial control of the system bus while the DMA is operating. This can affect not only total system throughput, but also such things as memory refresh and other interrupts.

Figure 1
Typical CPU I/O Sequence



1.2 DMA Characteristics

All DMA controllers are programmable, because the CPU must at least write a block length (byte count) and memory start address to them before they can start to manage a data transfer. The start address is incremented or decremented as the transfer proceeds, and the byte counter is decremented from the specific block length to zero.

Beyond this, however, DMA controllers vary substantially in their characteristic and capabilities. The following sections provide a general survey of the characteristics of DMA controllers, with only occasional reference to the SAB 82257 in particular.

1.2.1 Ports and Channels

Every data transfer has a source port and a destination port. For example, in memory-to-I/O transfers, memory is the source and I/O is the destination. The means of controlling and tracking the exchange of data between the two ports is called a "channel". A channel includes the hardware for address and byte counting, bus control, and coordination of the entire transfer process. Each port in a channel has its location specified either by the DMA address generation mechanism or by hardwiring. The SAB 82257, unlike most other DMA controllers, generates addresses for both, memory and I/O ports during each byte/word transfer; in other DMA controllers, the I/O port is hardwired. The ability to generate two addresses means that the DMA controller SAB 82257 can do memory-to-memory transfer on one channel whereas others either cannot do it at all or require two channels to do it.

Some DMA controllers have single channels, others, like the SAB 82257, have multiple channels which means that they can keep track of multiple interleaved transfers.

1.2.2 Transfer Methods

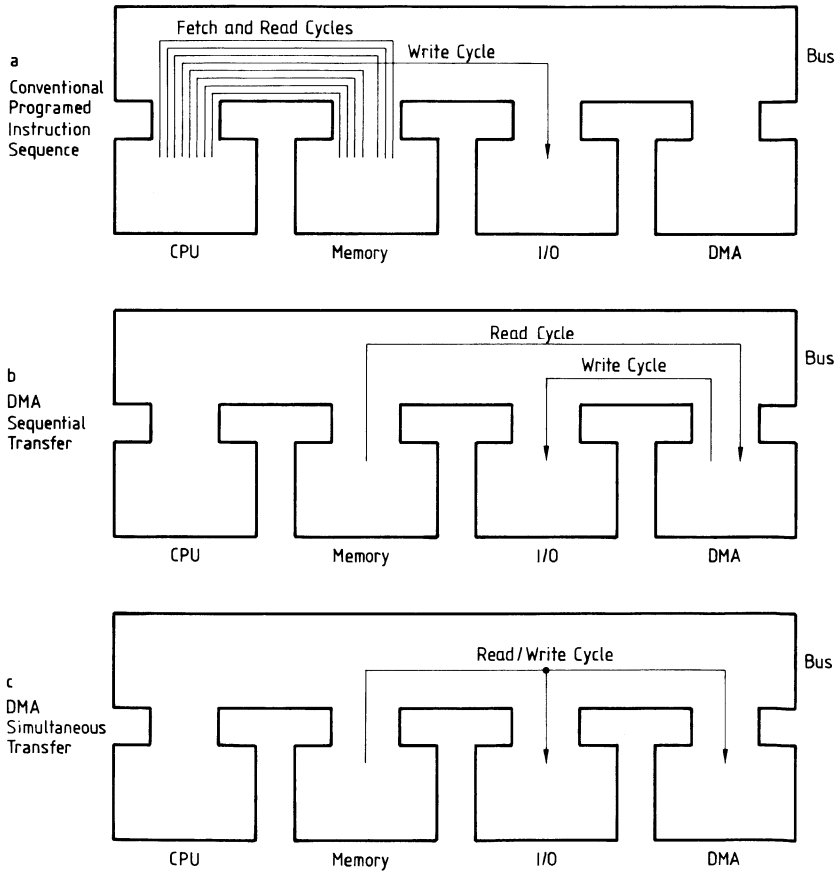
In section 1.1 we've mentioned the difference between handling I/O by conventional CPU instructions and direct memory access. Figure 2 expands this topic by comparing conventional CPU instructions with two different methods of DMA transfer. This figure shows the read and write cycles needed to accomplish the transfer of a single byte/word of data.

Figure 2a illustrates conventional CPU I/O instruction activity. The number of read and write cycles is approximated: many CPUs require more cycles than shown. The CPU instructions at least cause the execution of all steps illustrated in figure 1, plus additional housekeeping such as testing whether the next byte/word is ready for transfer.

Figure 2b illustrates a "sequential" or "two-cycle" DMA transfer in which a byte/word is read from the source port (in figure 2b: memory) into the DMA controller, and is then written to the destination port. Sequential transfer provides speeds that match or exceed the capability of most serial communications processors, and of many other I/O or memory devices. This method can be implemented on each of the four channels of the SAB 82257, and allows a total data rate of up to 4 Mbytes/second.

Figure 2c illustrates a "simultaneous" or "single-cycle" DMA transfer in which a byte/word is transferred directly from the data source to the data destination in a single bus cycle. This fastest transfer method can also be implemented on each of the four SAB 82257 channels, and allows a total data rate of up to 8 Mbytes/second as single channel data rate or as cumulative data rate of multiple channels (simultaneous operation of several channels in single-cycle mode).

Figure 2
Comparison of Various I/O Transfer Methods



Another method also used on certain DMACs is called "cycle-stealing" transfer. This technique works in a manner similar to the one shown in figures 2b and 2c, except instead of taking control of the bus it causes the DMA data transfers to be interleaved with CPU cycles where dynamic memory otherwise would be refreshed. This method inhibits memory refresh when DMA occurring and, additionally, reduces DMA throughput in some cases.

All DMA transfers interrupt dynamic memory refresh by the CPU, and most of them obstruct the CPU. Therefore, it is important to consider these implications when making the trade-off for higher DMA transfer speed.

1.2.3 Bus Control

Most DMA controllers do not control the system bus (processor bus) in the way it would be controlled by a CPU. For example, many DMA controllers neither have a straightforward interface to the system data bus (but rather multiplex a portion of the memory address to the data bus from which it must be latched by external logic) nor do most of them generate all of the bus control and status signals that a CPU would generate, and therefore they lack some degree of bus control.
The SAB 82257 has an adaptive bus interface, which means that it directly fits – per hardware and/or software configuration – into all the conventional 16 (and 8)-bit microprocessors like

- SAB 8086
- SAB 8088
- SAB 80186
- SAB 80188
- SAB 80286

without requiring additional TTL glue chips (i.e. all status and bus timing information is identical with that of the above CPUs if the SAB 82257 operates in the corresponding interface mode). Therefore it can use the same support components such as latches, transceivers, controllers and arbiters as are used by the corresponding processor. This property considerably simplifies design and reduces the number of required parts.

1.2.4 Programmability

The way a DMA controller starts, transfers data, and stops is determined by control information which the CPU writes to the DMA controller prior to the transfer. Status registers, which can be read by the CPU to determine the transfer condition after the DMA controller stops transferring are typically provided as well. The degree of programmability is directly related to the DMA controller's flexibility of handling a variety of transfer tasks. Most DMACs are quite limited in their programmability.

In contrast, the SAB 82257 provides more than 80 bytes of on-chip, user-accessible, logically ordered registers used to tailor the device (and retailor it between operations) to a wide variety of tasks and environments. Additionally the CPU does not only communicate with the SAB 82257 through the registers as it is the case with conventional DMA controllers. Communication is also memory-based. The CPU generates a "command block" somewhere in memory, informs the SAB 82257 where it is and issues a "start channel" command. On getting the start command, SAB 82257 autonomously loads the entire command block,

which contains all the information necessary to execute the DMA operation (source, destination pointers, byte count, each 24 bits long), from memory into its on-chip channel register and starts execution. This topic, as well as the other ones described earlier, are detailed in the following chapters. We have introduced them here to simply give a generalized framework that can be used to launch a more detailed discussion of the SAB 82257 DMA controller.

Overview

2 Overview

2.1 General Information

The SAB 82257 is a high-performance, general-purpose, four-channel DMA controller tailored for efficient high-speed data transfer between peripheral devices and memories. It is either coupled tightly with a companion CPU (local mode) or working in stand-alone applications using the remote mode. The SAB 82257 provides a complete direct interface to the SAB 80286 as well as to SAB 80186/188/86/88 bus. For example, in an SAB 80286 CPU environment the SAB 82257 generates the same signal levels and timings that the SAB 80286 CPU would generate to accomplish a transfer. Unlike with most other DMA controllers no external TTLs are required. For this reason, and because of its extensive programmability for operations on data and data flow, the SAB 82257 unburdens not only the CPU but also the system designer.

The SAB 82257 is also outstanding with respect to performance. Its four superfast DMA channels – transfer rates up to 8 megabytes per second are possible – make it one of the most powerful monolithic 16 (and 8)-bit controllers to service fast devices. Interrupt signals can be generated under several conditions. If multiple DMA controllers are needed, multiple SAB 82257 DMACs can be integrated very easily by using a simple local bus arbiter logic. Finally, the SAB 82257 is one of a few DMA controllers that provide source or destination-oriented data chaining (two chaining modes) as well as conditional command chaining (jumps within channel program based on several conditions).

A brief survey of the SAB 82257's features is given in the following. Each of the points is described more thoroughly in this and other chapters.

Features

- 16-bit DMA controller for
  16-bit family processors
  - SAB 80286
  - SAB 80186/188
  - SAB 8086/88
- 4 independent channels
- 16 megabyte addressing range
- 16 megabyte byte count
- Memory-based communication with CPU
- Transfer rates up to 8 megabyte/s
- Single-cycle and two-cycle transfer
- Automatic chaining of command blocks
- Automatic chaining of data blocks
- Local and remote (standalone) mode
- Support of 16-bit and 8-bit data buses
- Programmable synchronization modes
- Programmable control of channel priorities
- Direct and fast CPU/channel communication

2.2 DMA Operations

This section contains a survey of the SAB 82257 DMA operations.

2.2.1 Single-Cycle and Two-Cycle Transfer

Each of the four independent high-speed channels controls data transfer in two basic operating modes:

- single-cycle mode
- two-cycle mode

In **single-cycle mode**, bytes or words (16 bit) are transferred directly from the data source to the data destination in a single bus cycle per transfer. This mode enables a total data rate of up to 8 megabytes per second, and that as single channel data rate or as cumulative data rate of multiple channels (simultaneous operation of several channels in single-cycle mode). Thus the advantage in single-cycle mode lies in speed.

In **two-cycle mode**, source data is always stored within the SAB 82257 before being sent out to the destination. Although half as fast as single-cycle transfer, it has several compensating advantages. Since the data actually enters the controller, there is a possibility to act on the data.

A special feature of two-cycle transfer is **"automatic assembly and disassembly"** of data in bytes and words, meaning that the data can be read as one 16-bit word and written as two bytes, or vice versa. This is often desirable when using 8-bit-wide peripherals in a 16-bit system. In two-cycle transfer mode, data may also be transferred from one memory region to another which is impossible with single-cycle transfer.

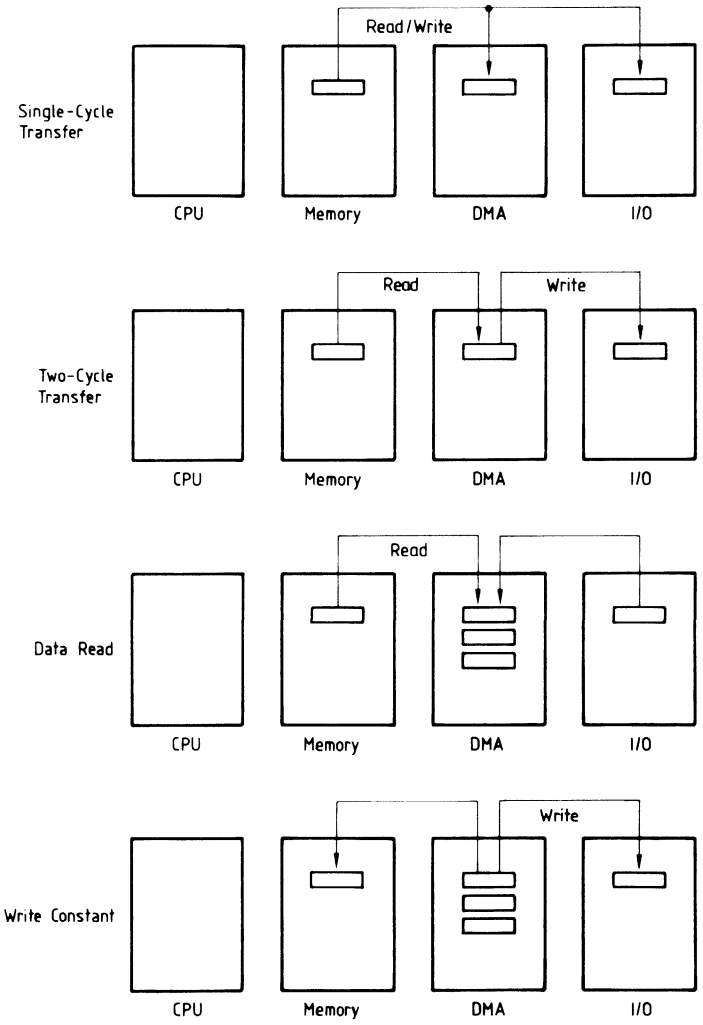
2.2.2 Data Read (DMA with no destination pointer)

A special modification of the two-cycle data transfer is the transfer without destination pointer. In this mode, only the source bus cycle of a two-cycle data transfer is executed. The data byte or word is read into the data assembly register (DAR) of the SAB 82257 but is not written out.

Other applications:

- Refresh dynamic memory by simply executing dummy read cycles
- Skip a peripheral data block in a sequential access memory
- Search a definite byte/word in a memory block.

Figure 3
Classes of SAB 82257's DMA Operations



2.2.3 Write Constant (DMA with no source pointer)

A special modification of the two-cycle data transfer is the transfer without source pointer. In this mode, only the destination bus cycle of a two cycle data transfer is executed. Thereby a byte or word specified within the command is used as source data. Thus, this DMA transfer operates like a literal or constant transfer. This is very useful for writing a command byte or word to a peripheral register.

Other applications:

- Fill a memory block with a constant
- Erase a peripheral data block
- Write a command to a register

2.2.4 Data Chaining

Often it is necessary to gather various source data blocks to one destination (source data chaining) or, conversely, to scatter data from one source to several destination block locations (destination data chaining). The SAB 82257 DMA controller offers source and destination chaining in two basic modes:

- list data chaining
- linked list data chaining.

In **list data chaining** mode, each data block of a data chain is specified by a list element of a chaining list. After each particular block transfer, the next list element is processed. Going to the next block only takes one microsecond (at 8 MHz system clock).

In **linked list data chaining** mode, each list element specifying a particular data block also holds a pointer (link pointer) to the next list element which should be processed. Thus, data blocks can be included, removed, or their sequence altered dynamically by manipulating the link pointers through the CPU. This feature, for example, can be used for serial data communications controllers, where different blocks represent different types of information, like header, address, tail, and so on. Linked list chaining is a little bit slower, but offers greater flexibility than list chaining.

2.2.5 Fast Channel Switching

For high-speed DMA controllers with multiple channels, it is very important that no time penalty (no latencies) be imposed on them when switching from one channel to another during data transfer. Even though the SAB 82257 is a multi-stage pipelined, microprogram-controlled machine, channel switching imposes no performance penalty on the SAB 82257. Depending on the channel requests and the priority scheme used, as for example.

- fixed priority or
- rotating priority

the data transfer can be switched from one channel to the other after each bus cycle. In the case of rotating priority of all four channels, there is no discrimination of any channel. Thus, a data transfer rate of 2 Mbytes/s can be processed per channel, if all the channels are active.

2.2.6 Synchronization of Data Transfers

For synchronization of DMA transfers the SAB 82257 distinguishes basically two modes:

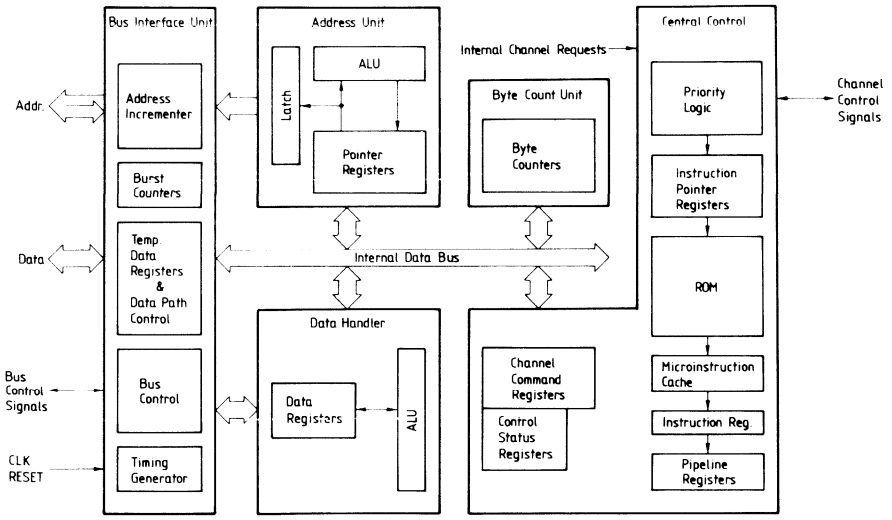
- external synchronization
- internal synchronization (free running).

The preferred synchronization mode is the **external synchronization** of DMA transfers. The external DMA request may be issued from the source (source synchronization) or destination (destination synchronization) device. It can initiate a single-cycle or a two-cycle DMA transfer. The external synchronization allows to control input/output operations at the data rate of peripheral device and to occupy the bus only if the peripheral is really able to receive or to transmit data.

The DMA transfer can also be performed in a **free-running** mode, i.e. without any external synchronization. This is advantageous for memory-to-memory transfers.

Free-running data transfer is also performed during a continuous DMA request or in the data block multiplex subchannel after channel start. Note that on the byte/word multiplex channel every DMA cycle has to be started with a request.

Figure 4
SAB 82257 Block Diagram



2.3 Block Diagram

The SAB 82257 is built as a multi-stage pipelined, microprogram-controlled machine with dedicated hardware for time-critical operations. This is because in most applications it will perform high-speed DMA operations on multiple channels where, for example, no time penalty should be imposed when switching from one channel to another.

The SAB 82257 is internally divided into the functional units depicted schematically in figure 4. The SAB 82257 block diagram also includes the major logic blocks and some of the accessible, user visible registers. The data interconnection paths are also shown. Not shown are the various control signals between the functional units. A detailed description of the logic blocks and their functions can be found in the following sections.

2.4 Reflections on System

2.4.1 Generalized Bus System

Three types of buses are generally referred to in this manual (see figure 5):

- local bus,
- system bus and
- resident bus.

The local bus is the bus comprised of signals coming directly from the microprocessor and/ or SAB 82257 components. The local bus is generally not buffered.

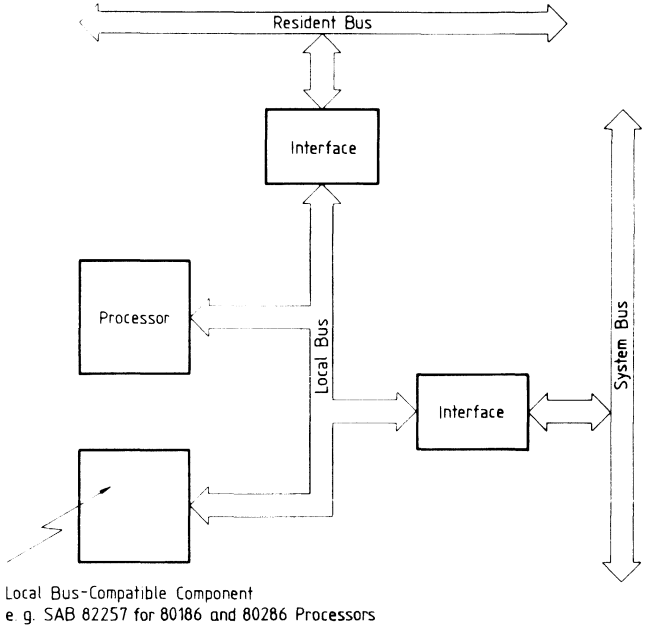
The local bus is interfaced with components like bus arbiter, bus controller, address latches, data drivers to generate a **system bus**. The system bus is a multiprocessor bus, i.e. it can be used by other processors, too. All the resources on the system bus are accessible for each bus master. Since there can be multiple bus masters competing for access to shared resources on the system bus, the system bus must be arbitrated ot only one master at any given time.

The local bus can also be interfaced by components like bus controller, address latches and data drivers to form a so-called **resident bus**. The resident bus is private to the local bus master, i.e. no arbitration is necessary for using this bus.

Note:
All these buses may be 8 bit or 16 bit wide. In general, a 16-bit bus leads to a higher throughput and an 8-bit bus requires less interface components.

The memory on a 16-bit bus must be (16 bit) word-organized, byte-addressable as for SAB 8086 systems.

Figure 5
Generalized Bus System



2.4.2 Address Spaces and Mapping

Since the SAB 82257 can access memory components located in two different address spaces – like the SAB 8086, SAB 80186 and SAB 80286 microprocessors – two types of address spaces are generally referred to in this manual:

- memory space and
- I/O space.

The **memory space**, which coincides with the CPU's memory space, may contain

- up to 16 Mbyte, if the SAB 82257 works tightly coupled with an SAB 80286 processor, or if it works loosely coupled with a CPU by means of the system bus (remote mode);
- up to 1 Mbyte, if the SAB 82257 works tightly coupled with SAB 80186/188/8086/88 processors.

The **I/O space**, which may either coincide with the CPU's I/O space (in local mode, see section 3.1.1) or be private to the SAB 82257 (in remote mode, see section 3.1.2), may contain

- up to 16 Mbyte, if the SAB 82257 works tightly coupled with an SAB 80286 processor, or if it works loosely coupled with a CPU by means of the system bus;
- up to 1 Mbyte, if the SAB 82257 works tightly coupled with SAB 80186/188/8086/88 processors.

If the SAB 82257 works in a stand-alone application using remote mode,

- the memory space is called "system space" and
- the I/O space is called "resident space"

throughout the rest of this manual.

The SAB 82257 does not distinguish between accessing a peripheral and accessing a memory. Thus, either peripheral or memory can belong to either of the two spaces. Each space can independently be 8 bits or 16 bits wide.

2.5 Upgradability

The SAB 82257 provides complete upward-compatibility with the advanced DMA controller SAB 82258. The SAB 82258 offers enhanced functions (i.e. data handling, improved flexibility of DMA channels, etc.) beyond the performance of the SAB 82257.
Each system or application can easily be upgraded, as the SAB 82257 provides compatibility on any level.

Interfaces

Both devices have identical bus interfaces for 286 mode, 186 mode and remote mode. Therefore both DMA controllers fit into all the processor systems described. The DMA interfaces are compatible as well. Upgrading a system hardware is no problem at all, because hardware redesigns in existing systems are not required due to compatibility.

Programming Technique

Portating system software to an SAB 82258 system is also no problem. The arrangement of the SAB 82257 control registers and even the bit arrangement within these registers can be found within the SAB 82258 as well. The locations of the control bits have been chosen such that upgrading is allowed directly. If the control registers and bits are used as described in this manual, the programmer can be sure that any software (channel programs including data chain lists, etc.) written for the SAB 82257 will run on the SAB 82258 without requiring changes. Changing the channel programs will be necessary, of course, when utilizing the additional functional features of the SAB 82258!

Summary

In addition to offering excellent performance, the SAB 82257 opens the door to more advanced DMA controllers. For the reasons pointed out in this section the SAB 82257 is the ideal component for getting into touch with the growing family of Siemens high-performance DMA controllers (please also refer to section 6.5 Reflections on Compatibility).

Operating Modes

3 Operating Modes

3.1 Basic Modes

Concerning the connection to a processor, the SAB 82257 has two basic modes of operation:

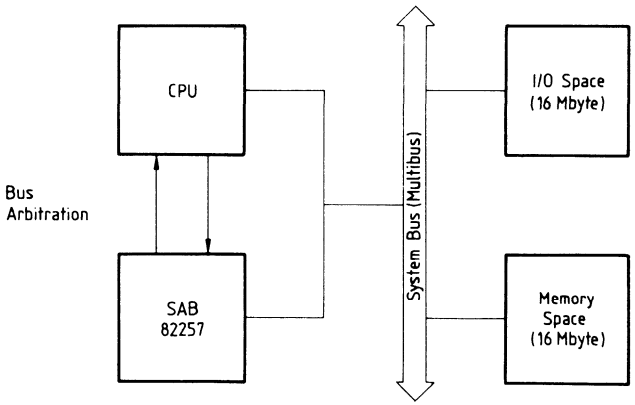
- local mode (SAB 82257 tightly coupled to a processor) and
- remote mode (SAB 82257 loosely coupled to a processor).

3.1.1 Local Mode

In local mode the SAB 82257 is directly coupled to the processor and uses the same local bus (see figure 26). The bus arbitration sequence (HOLD/HLDA and $\overline{RQ}/\overline{GT}$ protocol, respectively) ensures that only one of those devices occupies the bus at any time. All addresses issued by the SAB 82257 can be separately programmed for memory or I/O space. Both spaces are 16 Mbyte large for the 286 and 1 Mbyte large for the 186/188/8086/88 mode.

The local mode combination of processor and SAB 82257 working through a shared bus is very compact and efficient. This operating mode is possible with processors like SAB 8086/8088/80186/80188/80286.

Figure 6
Local Configuration



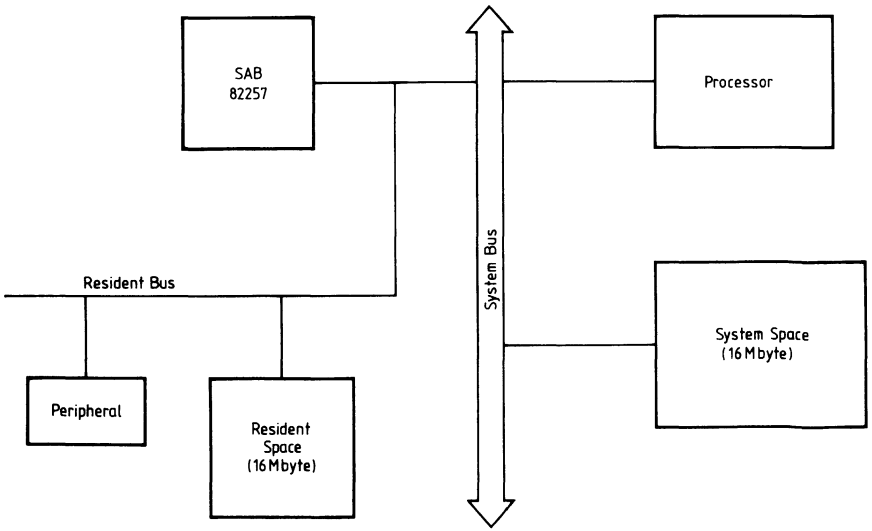
3.1.2 Remote Mode

Although the main application of the SAB 82257 will be to work tightly coupled with a companion CPU, the SAB 82257 is also designed to work together with a processor via the system bus. In this second basic operating mode, called remote mode, the SAB 82257 has its own set of bus interface circuits and can thus dispose of its own local bus (see figure 7).

This allows the DMA controller to work in parallel to the main processor, accessing resources on the resident bus most of the time, and accessing the system bus only for communication with the processor.
All addresses issued by the SAB 82257 can be programmed separately for system or for resident space (both 16 Mbyte). As a result of the private bus (resident bus) of the SAB 82257, a bus arbitration sequence is only necessary for system space accesses. However, although the SAB 82257 is master and arbiter of its local/resident bus, the SAB 82257 registers are always accessible for the processor via the system bus.

The remote configuration (see figure 7) is ideally suited to interface the SAB 82257 to any processor (including 68000, 32000 etc.) and for producing modular systems where every board has a specific function. For example, a subsystem dealing with Winchester and floppy disks could be put on a separate board holding the SAB 82257, sufficient resident memory and the necessary controllers.

Figure 7
Remote Configuration



3.2 Bus Interface Operating Modes

3.2.1 Local-Mode Bus Interface

In local mode the SAB 82257 bus interface has two basic timing modes:

- the 286 mode and
- the 186 mode.

In **286 mode**, the SAB 82257 is connected to the SAB 80286 processor bus (processor pins) thus sharing all the necessary bipolar support chips (latches, transceivers, controllers and arbiters). All status and bus timing information is identical with that of the SAB 80286 bus.

In **186 mode**, the SAB 82257 is connected to the SAB 80186 processor bus (processor pins). All status and bus timing signals meet the SAB 80186 bus timing signals.

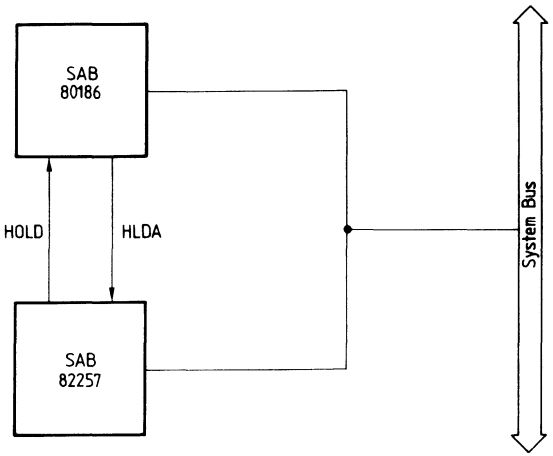
In addition there are three derivates of the 186 mode:

- the 188 mode,
- the 8086 mode and
- the 8088 mode.

In **188 mode**, the SAB 82257 operates with the SAB 80188. In this mode the signals and timings are identical to 186 mode, but the physical bus width (in general mode register) must be programmed to 8 bit.

In **8086 mode**, the SAB 82257 operates with the SAB 8086 using $\overline{RQ}/\overline{GT}$ protocol for bus arbitration instead of the HOLD/HLDA protocol in the other modes. All other signals and timings are identical to the 186 mode.

Figure 8
186 Mode Configuration



In 8088 mode, the SAB 82257 operates with the SAB 8088. In this mode the signals and timings are identical to 8086 mode, but the physical bus width (in general-mode register) must be programmed to 8 bit.

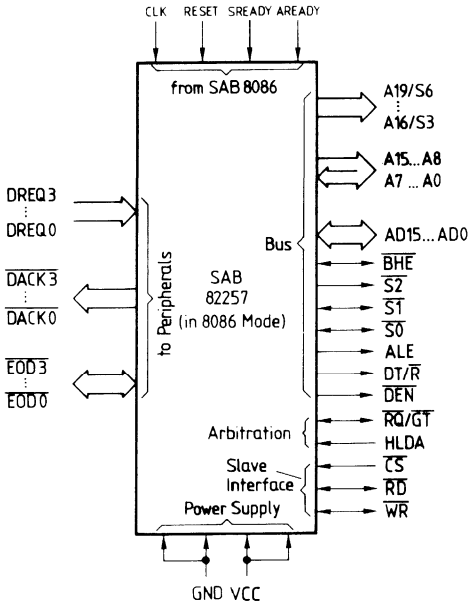
3.2.1.1 186 Mode

If 186 mode is selected via the AREADY and HLDA pins on RESET, all the signals and timings are compatible with the SAB 80186 processor. This means that the SAB 82257 has a multiplexed bus identical to that of the SAB 80186, and in 186 mode configuration (see figure 8) all signals with the same names (see figure 9) are directly interconnected with the ones of SAB 80186.

Only exception: CLK is connected to CLKOUT of SAB 80186.

A0 to A7 are bidirectional in the sense that they are latched in along with chip select ($\overline{CS}$) at a proper time (derived from status signals $\overline{S0}$ to $\overline{S1}$) to achieve a synchronous chip select.

Figure 9
Pinning in 186 Mode



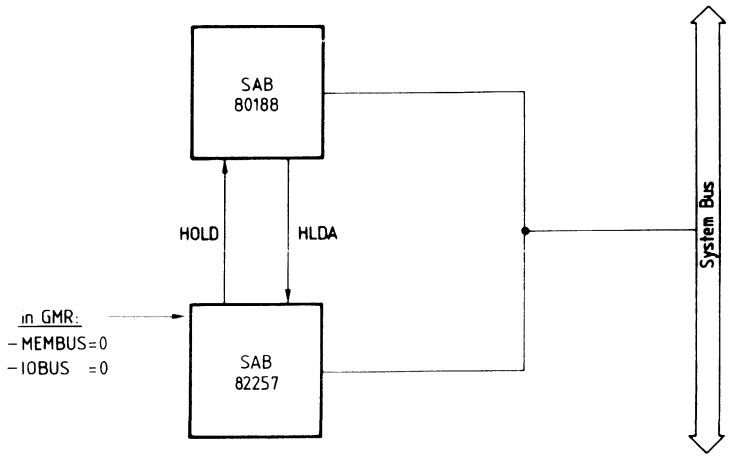
3.2.1.2 188 Mode

If 186 mode is selected via the AREADY and HLDA pins on RESET and the physical bus width is programmed to 8 bit in the general mode register (GMR), the SAB 82257 operates in 188 mode and its bus interface is compatible to the SAB 80188 processor.

Since there is no difference to operating in 186 mode – including signals, timings, pinning, etc. – the 188 mode is not described explicitly in the subsequent chapters.

Throughout the remainder of the manual, all descriptions and information concerning the SAB 82257 operating in 186 mode are valid also for operating in 188 mode.

Figure 10
188 Mode Configuration



3.2.1.3 8086 Mode

If 8086 mode is selected via the AREADY and HLDA pins on RESET all the signals and timings are compatible to the 8086 processor. This means that the SAB 82257 has a multiplexed bus identical to that of the SAB 8086, the arbitration between the different bus masters on the local bus is done using $\overline{RQ}/\overline{GT}$ lines, and in 8086 mode configuration (see figure 11) all signals with the same names (see figure 12) are directly interconnected with the ones of the SAB 8086.

Exceptions:
- CLK is connected to CLKOUT from SAB 8086.
- After entering the 8086 mode, pin HLDA has no function.

A0 to A7 are bidirectional in the sense that they are latched in along with chip select ($\overline{CS}$) at a proper time (derived from status signals $\overline{S0}$ to $\overline{S1}$) to achieve a synchronous chip select.

Figure 11
8086 Mode Configuration

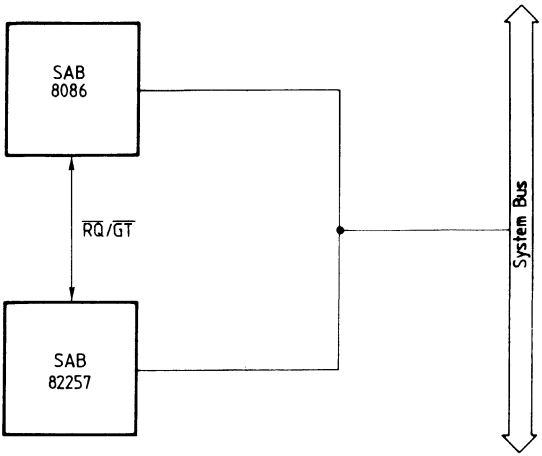
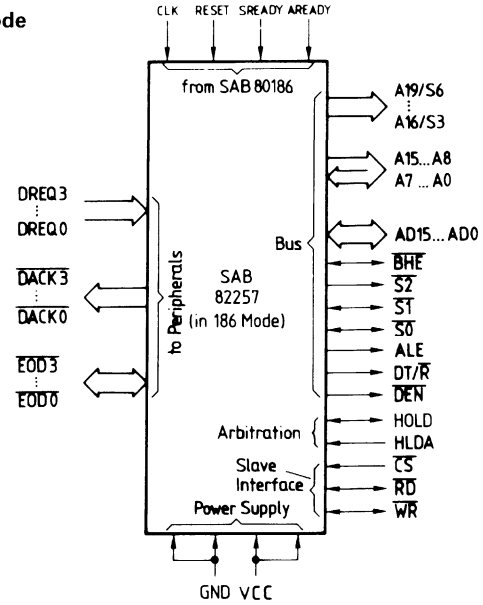


Figure 12
Pinning in 8086 Mode

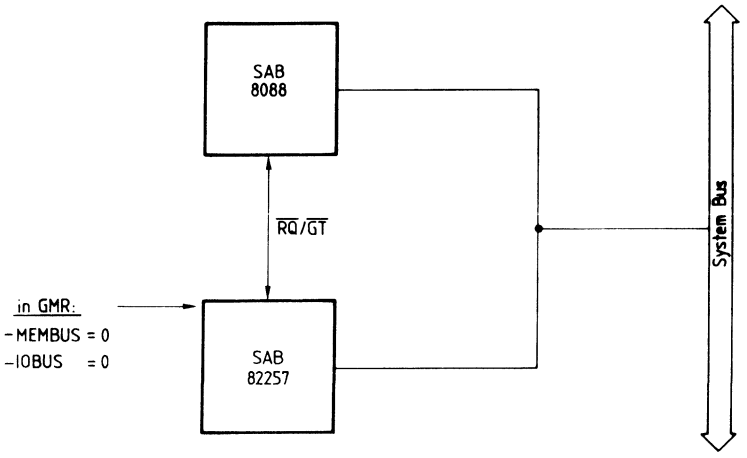


3.2.1.4 8088 Mode

If 8086 mode is selected via the AREADY and HLDA pins on RESET and the physical bus width is programmed to 8 bit in the general mode register (GMR), the SAB 82257 operates in 8088 mode and its bus interface is compatible with the SAB 8088 processor.

Since there is no difference to operating in 8086 mode – including signals, timings, pinning, etc. – the 8088 mode is not described explicitly in the subsequent chapters. Throughout the remainder of the manual, all descriptions and information concerning the SAB 82257 operation in 8086 mode are valid also for operating in 8088 mode.

Figure 13
8088 Mode Configuration



3.2.1.5 286 Mode

If 286 mode is selected via the A23 pin on RESET all the signals and timings are compatible with the SAB 80286 processor. This means that the SAB 82257 has a demultiplexed, pipelined bus identical with that of the SAB 80286, and in 286 mode configuration (see figure 14) all signals with the same names (see figure 15) are directly interconnected with the ones of the SAB 80286.

Exceptions: CLK, RESET and $\overline{\text{READY}}$ come from the SAB 82284.

Figure 14
286 Mode Configuration

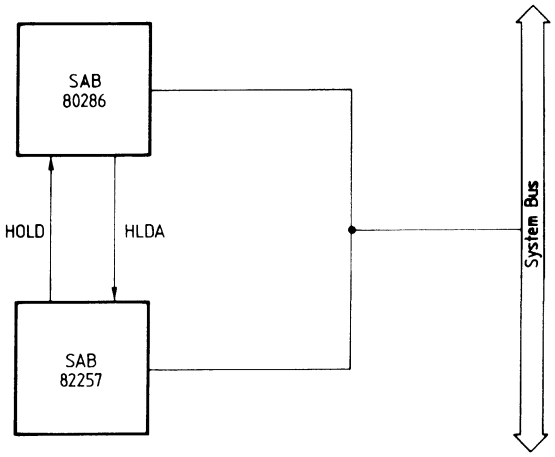
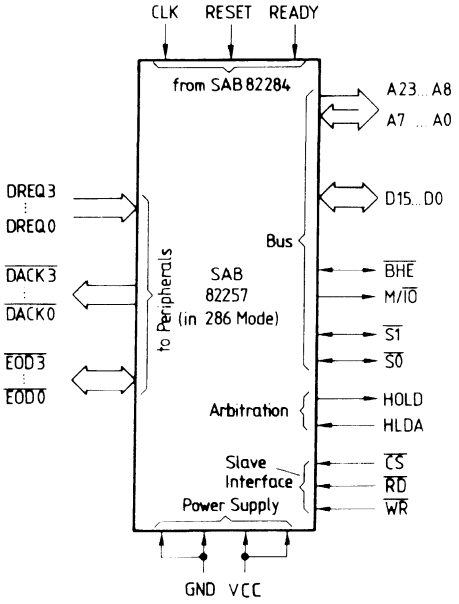


Figure 15
Pinning in 286 Mode



Note:

- For operation of the SAB 82257 together with the SAB 80286 processor in "protected virtual address mode" certain special considerations are necessary.
  The SAB 82257 itself does not support the protected virtual addressing of the SAB 80286 CPU, it only works with real addresses. But this does not mean a loss in security provided that the following conditions are fulfilled:
  1. The 80286 kernel software must check all the protection rules during the setup routine for the SAB 82257 (this is supported by the 80286 instructions).
  2. The 80286 kernel has to translate the logical addresses into the physical addresses and to perform the limit checks for all block transfers.
  3. All SAB 82257 registers should be memory-mapped and access to them should only be allowed for an 80286 kernel routine (task isolation).

  Normally, an I/O utility routine is provided by the operating system to service the SAB 82257.
  No direct user access should be allowed from lower privilege levels to the SAB 82257. The real addresses can be generated only by using the protection mechanism of the SAB 80286 and hence are checked against protection violation.
- A synchronous selection as slave through the SAB 80286 as in 186 mode is also provided here.

3.2.2 Remote-Mode Bus Interface

In remote mode the SAB 82257 bus interface is slightly different from that in 286 mode. As a result all bus timing information is identical to that of the SAB 80286 bus.

This means that most of the bus signals (see figure 17) have the same function as in 286 mode,

the exceptions are: BREL, $\overline{CS}$, HOLD and HLDA signal,

and in remote mode configurations (see figure 16) the same support logic, such as bus controller and bus arbiter, as for the SAB 80286 processor can be used.

Since in remote mode the SAB 82257 is connected to the system bus as a bus master without being directly coupled to a processor (the communication with the main CPU is done via the system bus; system bus arbitration via HOLD and HLDA signals) and disposes of its own local bus (see figure 16), the bus interface must support an arbitration of the local bus to enable accesses to the SAB 82257 registers by the main processor.

This "local bus arbitration" in remote mode is done via the $\overline{CS}$ and BREL lines.

Note
For remote mode throughout this manual the I/O space is called the "resident space" and the memory space is called the "system space".

Figure 16
Remote Mode Configuration

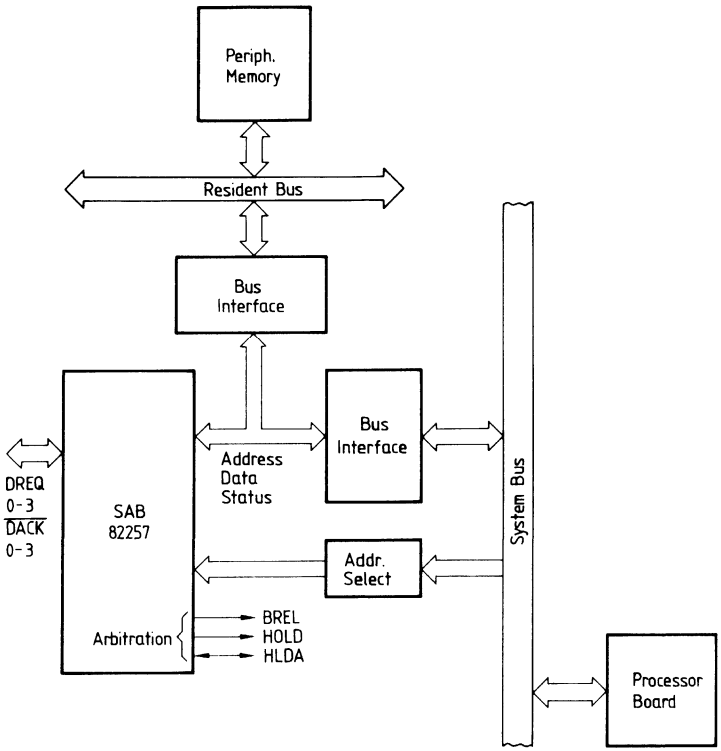
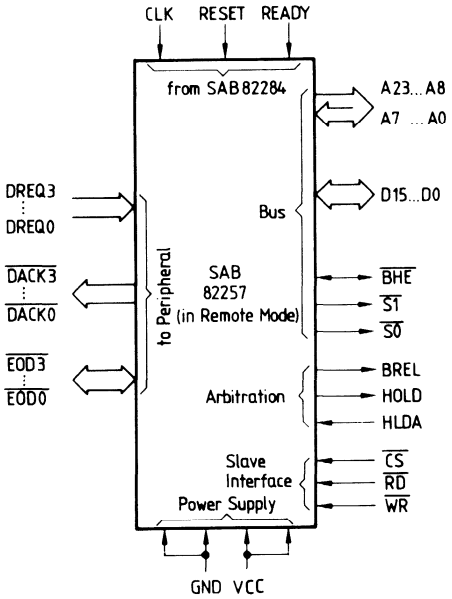


Figure 17
Pinning in Remote Mode



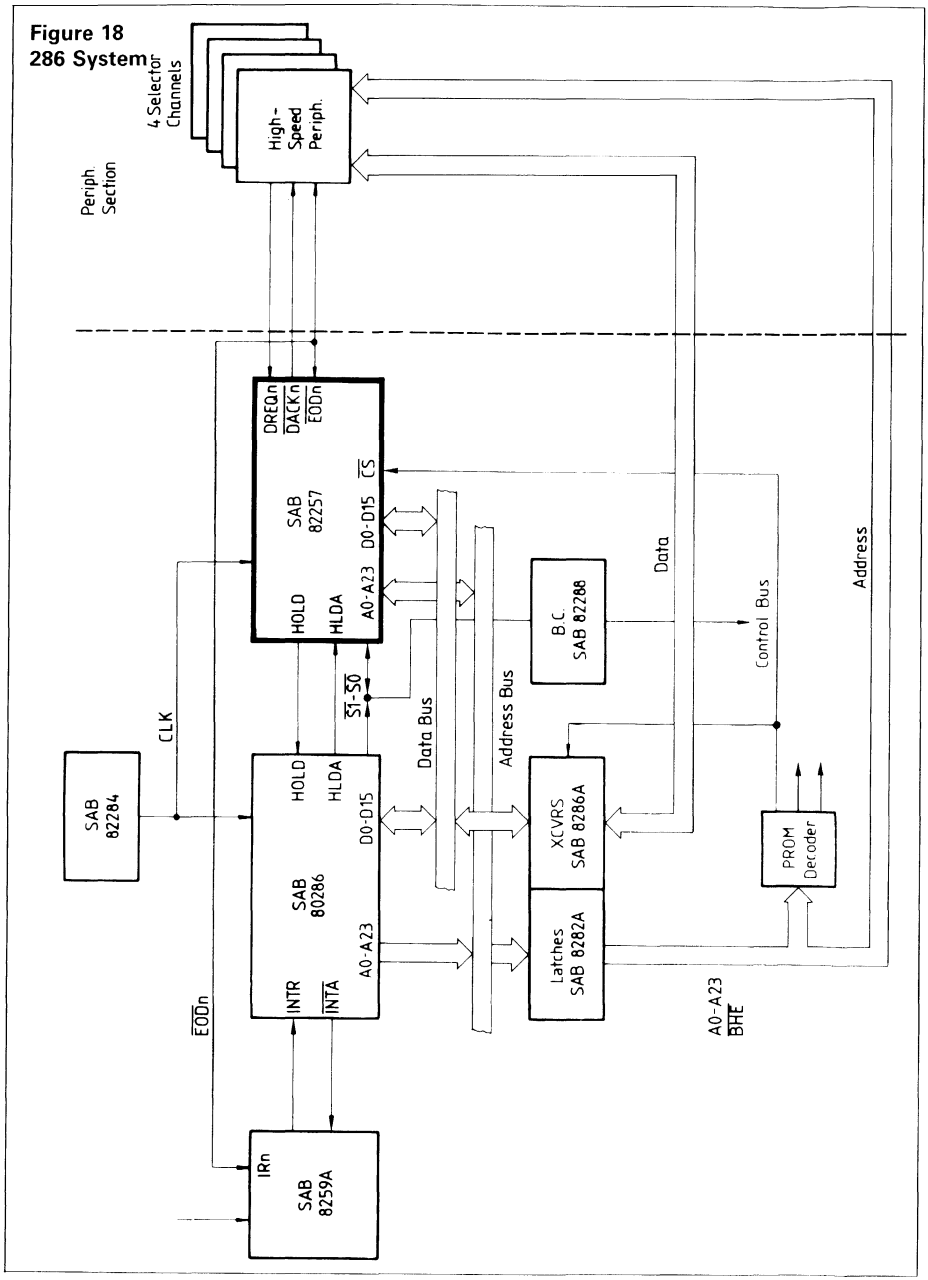
3.3 Typical System Configurations

3.3.1 286 System

The configuration in figure 18 shows a typical 286 system with the SAB 82257 where the DMA controller is tightly coupled with the SAB 80286 processor. The SAB 82257 resides on the local bus together with the processor thus being able to share all the support circuits, latches, transceivers, bus controller, bus arbiter, clock generator, etc. To ensure that only one of the two occupies the local bus, and hence the resident and system bus, at a time, the bus exchange signals HOLD and HLDA are used.
As illustrated in figure 18, a master interrupt controller is also needed (e.g. SAB 8259A) receiving the "End of DMA Signals" ($\overline{EOD}$) 0 to 3 as interrupts. Interrupts may occur in this configuration under any terminate condition, no matter whether internal or external, if programmed to do so.

The peripheral subsystem example consists of 3 high-speed peripherals (e.g. hard disk, CRT and communication lines).

Figure 18
286 System



3.3.2 186 System

The configuration in figure 19 depicts an SAB 82257 used in a "186 system" (SAB 82257 tightly coupled with an SAB 80186 processor) which also contains an SAB 8087 math processor. The SAB 82257 resides on the local bus together with the processor thus being able to share all the support components. In this system, the support components are the SAB 8086 bipolar components, latches, transceivers and bus controller.

An arbiter , e.g. SAB 82188 glue chip, is needed to arbitrate the SAB 8087 math processor and the SAB 82257 controller, since the SAB 80186 processor has only one set of bus exchange signals (HOLD and HLDA) to ensure that only one of them occupies the local bus, and hence the resident and system bus, at a time.

Some interrupt control logic is also needed (e.g. SAB 8259A) to process the four $\overline{EOD}$ signals. Interrupts may occur in this configuration under any terminate condition, no matter whether internal or external, if programmed to do so.

In 186 mode the SAB 82257 directly supports the SAB 80186 processor bus with 16 address bits A0 to A15 internally multiplexed onto the data lines, address pins A16 to A19 are multiplexed with status signals S6 to S3 and address pins A22 to A20 are used to generate the control signals ALE, $\overline{DEN}$ and DT/$\overline{R}$. The A23 pin serves as additional (asynchronous) ready input AREADY. As a master the SAB 82257 in 186 mode offers address pins A15 to A0 as latched output and shares all the SAB 80186 support components with the processor.

The peripheral subsystem consists of up to 4 high-speed peripherals.

3.3.3 8086 System

The configuration in figure 20 shows a typical "8086 system" with the SAB 82257 where the DMA controller is tightly coupled with an SAB 8086 processor. The SAB 82257 resides on the local bus together with the processor thus being able to share all the support components. In this system, the support components are the SAB 8086 bipolar components, latches, transceivers, and bus controller.

The local bus arbitration is done via the bus exchange signals $\overline{RQ}/\overline{GT}$. Because there are two $\overline{RQ}/\overline{GT}$ lines no local bus arbiter is necessary (in contrast to SAB 80186 systems) if an SAB 8087 processor is added to the system as shown in figure 19. Some interrupt control logic is also needed (e.g. SAB 8259A) to process the four $\overline{EOD}$ signals. Interrupts may occur in this configuration under any terminate condition, no matter whether internal or external, if programmed to do so.

In 8086 mode the SAB 82257 directly supports the SAB 8086 processor bus with 16 address bits A0 to A15 internally multiplexed onto the data lines, address pins A16 to A19 are multiplexed with status signals S6 to S3 and address pins A22 to A20 are used to generate the control signals ALE, $\overline{DEN}$ and DT/$\overline{R}$. The A23 pin serves as additional (asynchronous) ready input AREADY. As a master the SAB 82257 in 8086 mode offers address pins A15 to A0 as latched outputs and shares all the SAB 8086 support components with the processor.
The peripheral subsystem consists of up to 4 high-speed peripherals.
Note that the system configuration for the SAB 8086/8088 in maximum mode is very similar to the SAB 80186 configuration.

Figure 19
186 System

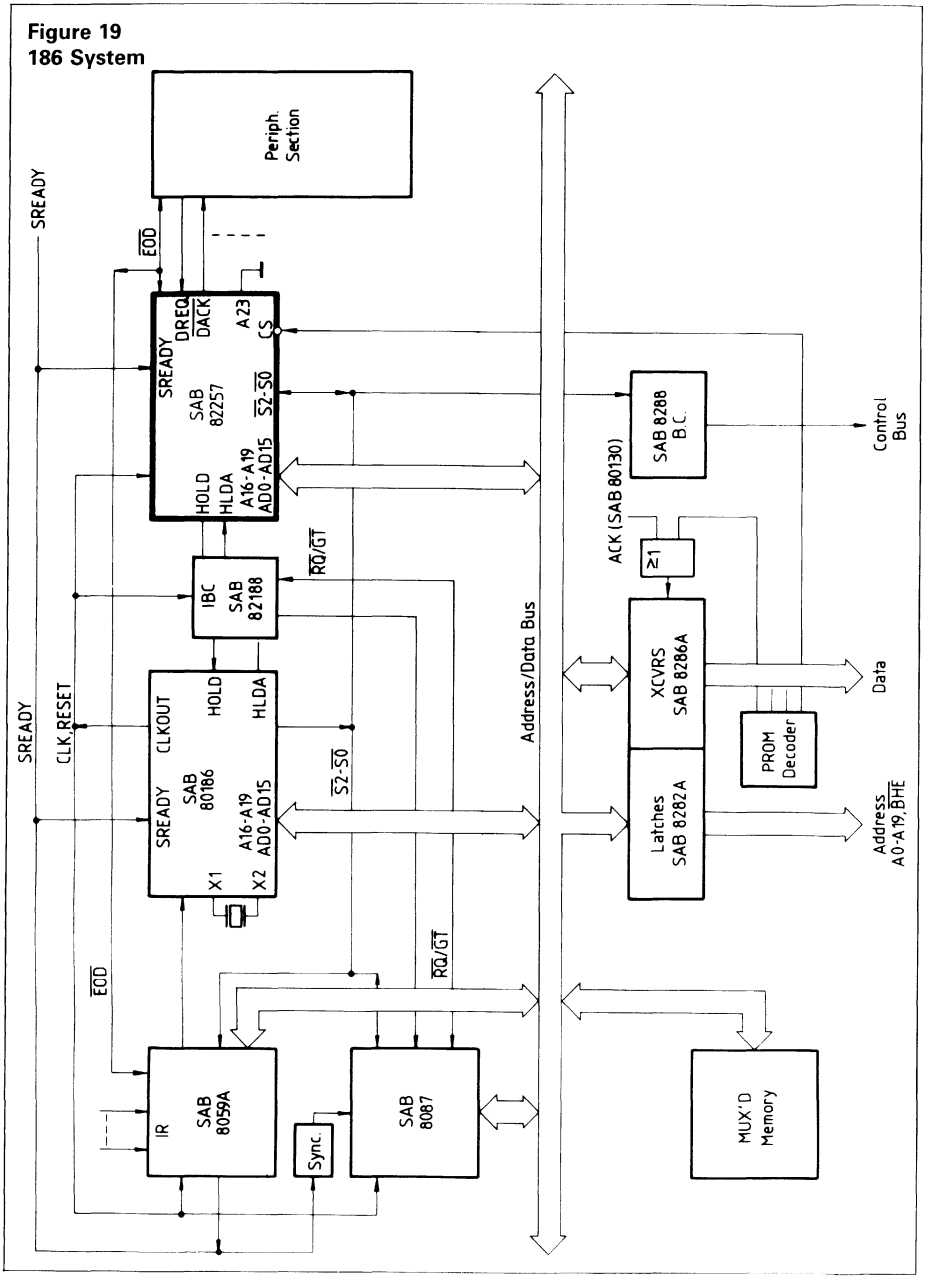
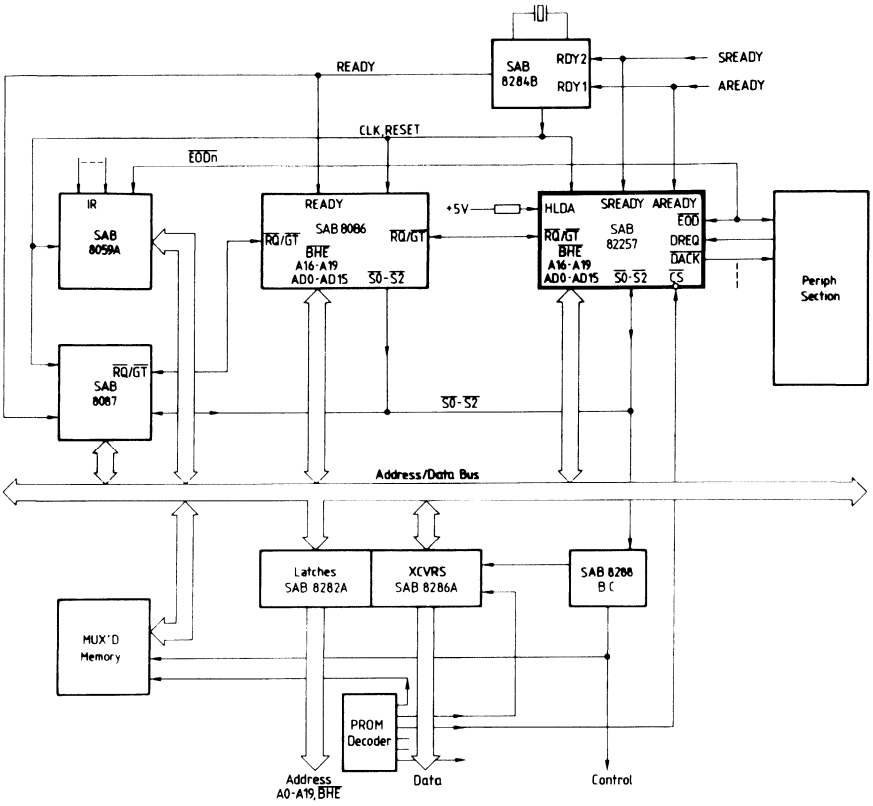


Figure 20
8086 System



3.3.4 Autonomous SAB 82257 Subsystem

The configuration in figure 21 depicts an "autonomous SAB 82257 subsystem" where the DMA controller is the sole local bus master interfaced to a CPU only through the system bus. In this SAB 82257 operating mode, called remote mode, the SAB 82257 can work in parallel to the processor since it has its own resident bus with private resources on it. The SAB 82257 requires essentially the **same** support components such as latches, transceivers, bus controllers, bus arbiters, clock generators, etc., as an SAB 80286 processor with two buses and **in addition** a standard slave select logic, so it can be addressed as a slave and read/written via the system bus (e.g. Multibus) by a CPU.

As shown in figure 21, a CPU request to the SAB 82257 is indicated by the signal $\overline{CS}$ and $\overline{RD}$ or $\overline{WR}$. On being selected via the system bus, the SAB 82257 waits till it can get off the local bus and then activates the BREL (bus released) signal. As a result, the register address reaches the DMA controller and system bus access to SAB 82257 is achieved. The transfer acknowledge signal ($\overline{XACK}$) is generated to terminate the access. The BREL signal uses the M/$\overline{IO}$ pin since in remote mode the M/$\overline{IO}$ signal is not needed.

Since the SAB 82257 is the only master of the local/resident bus, it can start a local bus cycle immediately, this means without any bus arbitration by a HOLD/HLDA sequence.
For system bus accesses, a special arbitration control of the local bus is necessary to prevent a situation where

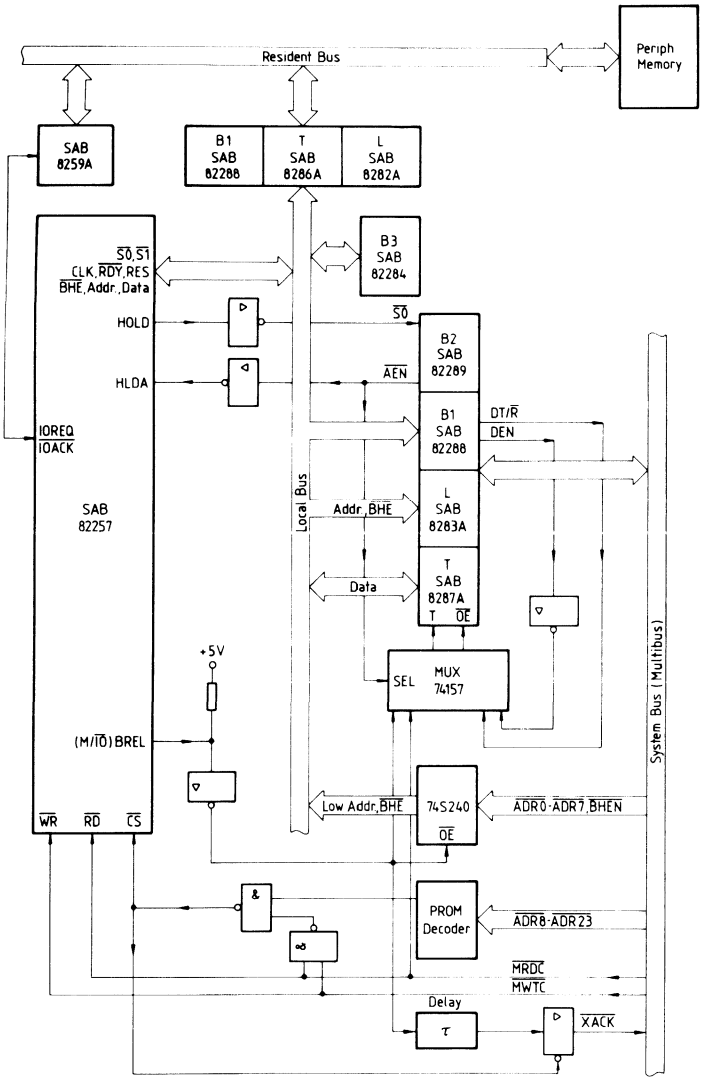
- the SAB 82257 waits to access the system bus thereby occupying the local bus and at the same time
- the CPU waits to access the SAB 82257 (via SAB 82257 local bus) thereby occupying the system bus.

This problem can only be solved if, in the case of system bus accesses, the SAB 82257 does not occupy the local bus until it has gained the system bus. Therefore, the SAB 82257 in remote mode initiates all system bus accesses (and only those) with a HOLD/HLDA sequence. Thus the HOLD signal can also be used for distinction between the two address spaces (local space, gobal space).

The peripheral subsystem consists of up to 4 high-speed peripherals.

Note
The bus interface and the slave select logic shown in figure 21 are standard, and are required for any master/slave system using the system bus.

Figure 21
Autonomous SAB 82257 Subsystem (Remote Mode)



Bus Operation

4 Bus Operation

4.1 Local-Mode Bus Operations

4.1.1 286 Mode

4.1.1.1 Bus Cycles

The internal hardware of the SAB 82257 communicates with the CPU and the external hardware via the bus using bus cycles.

Cycle Definitions

Three types of cycles are used to describe SAB 82257 operation:

- CLK cycle
- processor cycle and T-state
- bus cycle

CLK Cycle

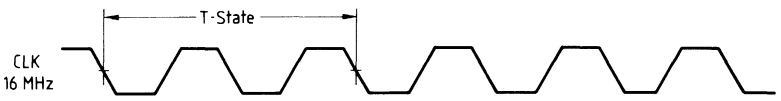
A CLK cycle is one period of the clock signal at the SAB 82257 CLK pin. In standard systems, the frequency of CLK may be 16 MHz.

Processor Cycle and T-State

The internal SAB 82257 logic is driven form an internal clocking system. One period of this internal clock is called a T-state. In standard systems, the frequency of the clocking system is 8 MHz.

In 286 mode, the clocking system uses the CLK frequency divided by two. Therefore, only the falling edge of CLK is important for the SAB 82257 timing and a T-state equals two CLK periods. In 286 local mode, the SAB 82257 must share the support components with the CPU. For correct operation in this case, the processor cycles of CPU and the T-states of SAB 82257 must be synchronized. This synchronization is done with the leading edge of the reset signal (see section 4.1.1.3) or with the leading edge of the bus status signals (see section "Bus Cycle Status").

Figure 22
T-State Definition



Bus Cycle

A bus cycle consists of an integer number of T-states. In 286 mode, a bus cycle has a minimum length of 3 T-states but, due to the pipelining of bus cycles, these bus cycles can come at a rate of one bus cycle for every two T-states. Therefore, the maximum bus rate is 4 million bus cycles or 8 megabytes per second. The bus cycles are, of course, compatible with the ones of the SAB 80286 microprocessor.

A bus cycle can be lengthened beyond the minimum by a delayed activation of the ready signal ($\overline{\text{READY}}$).

Bus Cycle Types

There are two basic types of bus cycles:

- active bus cycles
- passive bus cycles

Active Bus Cycles

For data transfers between the SAB 82257 and memory or peripherals, the SAB 82257 generates active bus cycles by activating the status signals $\overline{\text{S0}}$, $\overline{\text{S1}}$ and M/$\overline{\text{IO}}$. The SAB 82257 must be master of the local bus and control such a bus cycle. Such a bus cycle can be a read or a write from or to memory space or I/O space.

A normal DMA transfer needs at least two bus cycles:

- the read from source and
- the write to destination.

These two operations can be combined in a single bus cycle, for higher transfer rates (single-cycle transfer). Source or destination is accessed by the SAB 82257 address and the command, like a normal bus cycle, whereas the peripheral is selected by the $\overline{\text{DACKn}}$ signal. The data does not flow through the SAB 82257 but directly from source to destination via the data bus.

Therefore even for a write cycle data pins must float like for a read cycle. Because the SAB 82257 must control the bus cycle, the other bus pins function like during a normal bus cycle.

Note
For correct operation on a buffered data bus the direction of the data transceiver must be controlled by the $\overline{\text{DACKn}}$ signal.

Passive Bus Cycles

If the SAB 82257 is not the master of the local bus, then the CPU can generate a bus cycle which accesses one of the internal registers of the SAB 82257. The SAB 82257 monitors all bus cycles and, when it is selected via $\overline{\text{CS}}$, it executes the data transfer from the local bus into the addressed register or vice versa, as requested.

When the SAB 82257 and a processor share the local bus, then a passive bus cycle is normally a **synchronous** access to an SAB 82257 register, i.e. the processor activates the status line $\overline{\text{S0}}$ or $\overline{\text{S1}}$. The SAB 82257 monitors the status lines of the local bus and, when selected via $\overline{\text{CS}}$, it generates internal read and write signals which have the same timing as the signals generated by the support chips. The end of this passive bus cycle is defined by a $\overline{\text{READY}}$ activation in 286 mode. During execution of a passive cycle, the address signals A7 to A0, $\overline{\text{BHE}}$ signal and $\overline{\text{CS}}$ signal are latched. The execution of a synchronous passive cycle is fast enough to eliminate the need for wait states.

If no activation of the status lines $\overline{S0}$ or $\overline{S1}$ is recognized, the SAB 82257 monitors the $\overline{RD}$ and $\overline{WR}$ signals. By activation of one of these signals the SAB 82257 latches address A7 to A0, $\overline{BHE}$ signal and $\overline{CS}$ signal. After the synchronization of the asynchronous signals $\overline{RD}$ and $\overline{WR}$, the SAB 82257 begins an **asynchronous** passive bus cycle. The end of this bus cycle is defined by the trailing edge of a $\overline{RD}$ or $\overline{WR}$ signal.

Note
An asynchronous passive bus cycle requires more time than a synchronous one, because the synchronization needs additional time.

When a passive bus cycle is lengthened beyond the minimal time, a read from an SAB 82257 register is made once and the result is latched to the output. Therefore, a change of register contents during a passive bus cycle will not change the data read. By lengthening a passive write bus cycle, the transfer of the data on the pins to the selected SAB 82257 register is repeated until the end of the bus cycle. Therefore, only the last data on the data pins is stored in the register.

When the SAB 82257 is not in possession of the bus all output signals except the following are tristated:

- HOLD
- $\overline{DACK0}$ to $\overline{DACK3}$
- $\overline{EOD0}$ to $\overline{EOD3}$

Bus Cycle Status

In 286 mode, there are three signals to indicate the type of a bus cycle:

- $\overline{S0}$ signal
- $\overline{S1}$ signal
- M/$\overline{IO}$ signal

The beginning of a bus cycle is indicated by $\overline{S0}$ or $\overline{S1}$ or both going active. The termination of a bus cycle is indicated by the bus ready signal ($\overline{READY}$) going active.

When the SAB 82257 **is** the master of the local bus, it can generate the following bus cycles by activating $\overline{S0}$, $\overline{S1}$ and M/$\overline{IO}$:

M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Active Bus Cycle Type
0	0	0	Not Valid
0	0	1	Read from I/O Space
0	1	0	Write into I/O Space
0	1	1	Not a Bus Cycle
1	0	0	Does Not Occur
1	0	1	Read from Memory Space
1	1	0	Write into Memory Space
1	1	1	Not a Bus Cycle

Note
In 286 mode, the M/$\overline{IO}$ signal is valid with the address on the address lines.

When the SAB 82257 **is not** the master of the local bus, the status signals are used as inputs for detection of synchronous accesses to the SAB 82257.

The following table shows the bus status and $\overline{CS}$ signals and their interpretation by the SAB 82257:

$\overline{CS}$	$\overline{S1}$	$\overline{S0}$	Description
1	X	X	SAB 82257 Is Not Selected, No Action
0	0	0	No SAB 82257 Access, No Action
0	0	1	Read from an SAB 82257 Register
0	1	0	Write into an SAB 82257 Register
0	1	1	Not a Bus Cycle

Note
The SAB 82257 is selected but no synchronous access is activated. In such a case the SAB 82257 monitors $\overline{RD}$ and $\overline{WR}$ signals for detection of an asynchronous access.

In 286 mode, the leading edge of the status is also used for the synchronization of the internal SAB 82257 clock with the processor cycle. If the leading edge of status appears in the first half of a T-state, the SAB 82257 will lengthen this T-state by one CLK period. Due to this synchronization the first bus cycle of the processor after reset must not access SAB 82257 registers.

Bus High Enable ($\overline{BHE}$)

The SAB 82257 works on a 16-bit bus as well as on an 8-bit bus. To control transfers of bytes or words on such buses, there are two signals:

- bus high enable ($\overline{BHE}$) and
- the least significant address bit (A0).

A low on the $\overline{BHE}$ signal means that the data on the higher part of the bus (D15 to D8) is valid. A low on the A0 signal means that the data on the lower part of the bus (D7 to D0) is valid. Therefore, an even-addressed byte is transferred on the lower data bus byte and an odd-addressed byte is transferred on the higher data bus byte.
A full word can be transferred by using both data bus bytes (D15 to D0) and addressing the lower (even) byte of the word in conjunction with an active $\overline{BHE}$ signal.

For transfers on an 8-bit physical bus both kinds of bytes, the even-addressed and the odd-addressed byte, must be transferred over the lower data bus part. The transfer of an even-addressed byte uses the same coding as a byte transfer on a 16-bit bus. The transfer of the odd-addressed byte on the lower data bus half is indicated by $\overline{BHE}$ being high and A0 being high.

The following table summarizes this description.

$\overline{BHE}$	A0	Valid Data Bus Pins	Used for 16-Bit Bus	Used for 8-Bit Bus	Notes
0	0	D15 to D0	Yes	No	Word Transfer
0	1	D15 to D8	Yes	No	Odd-Addressed Byte on 16-Bit Bus
1	0	D7 to D0	Yes	Yes	Even-Addressed Byte
1	1	D7 to D0	No	Yes	Odd-Addressed Byte on 8-Bit Bus

The physical bus width is indicated by the MEMBUS bit and the IOBUS bit in the general mode register (GMR) for SAB 82257 bus cycles.
When an access to an SAB 82257 register is made (passive bus cycle) the SAB 82257 handles the data in the correct way by sensing the $\overline{BHE}$ and A0 signal and acting according to the above table.

Bus Ready

When a bus cycle must be lengthened beyond the minimal length, for supporting slower memories or peripherals, this must be done with the bus ready signal.
If the bus ready signal is not active at the sampling point, the bus cycle is lengthened by one T-state. The bus ready signal is also sampled one T-state later for additional lengthening of the bus cycle. Therefore, the bus cycle length can be extended indefinitely by an integer number of T-states.

In 286 mode the signal for indicating the ready status of the bus is:

- $\overline{READY}$,
  which is a synchronous input with the demand of meeting setup and hold times specified in the data sheet.

Any activation of the $\overline{READY}$ signal which does not meet the setup and hold times may cause erroneous SAB 82257 operation. Therefore, the SAB 82284 clock generator should be used guaranteeing correct activation of the $\overline{READY}$ signal.
An active bus ready signal at the sampling time terminates the bus cycle.

4.1.1.2 Bus Arbitration

For the arbitration of the bus there are two signals:

- HOLD signal
- HLDA signal (HOLD acknowledge signal)

In local configurations, these lines are directly connected to the processor. Whenever the SAB 82257 wants to perform one or more bus cycles it activates the HOLD signal. The processor then surrenders the local bus as soon as possible. The processor tristates its bus and acknowledges with a high on the HLDA signal. After sensing an active HLDA, the SAB 82257 gets onto the bus after a minimum of 3 CLK periods to perform bus cycles. If no further bus cycles are necessary after completing a bus cycle the SAB 82257 will surrender the local bus by floating the bus and switching the HOLD signal low 1 CLK after the last bus cycle. The processor acknowledges this by a low on the HLDA signal and takes the bus.

The SAB 82257 can be forced off the bus if HLDA is driven inactive. When "losing HLDA" it surrenders the bus

- after the currently running bus cycle, or
- after the bus cycle(s) following the running bus cycle.

The second case is for supporting inseparable bus cycle, which are:

- word transfer on odd addresses, which is realized by two bus cycles where each transfer is a byte transfer,
- fetch of 24-bit address pointers out of memory.

Figure 23
HOLD-HLDA Sequence, 286 Mode

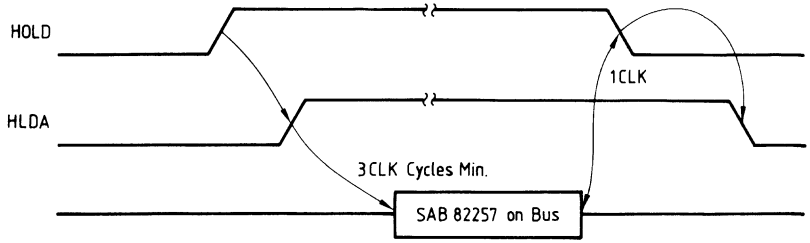
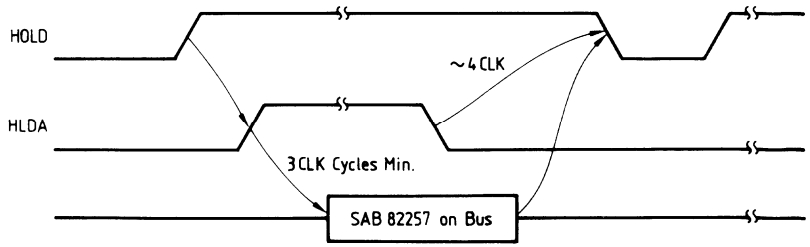


Figure 24
Losing HLDA, 286 Mode



The SAB 82257 signals the surrendering of the bus by floating the bus and removing the HOLD signal.
If requests for bus cycles are present the HOLD signal will go active after a delay of two T-states.

Note
After HLDA is made inactive, the SAB 82257 releases the bus within an average of 4 CLK cycles.

4.1.1.3 Reset Signal

A reset signal on pin RESET (high level), forces the SAB 82257 into a well defined initial state. This state is characterized as follows:

1. Upon activation of the RESET signal
   - all channels are stopped
   - all bus activities are stopped
   - all tristate signals are in tristate and others in passive state.
2. After activation of the RESET signal
   - above state is maintained, and in addition:
   - some registers have defined values or defined control bits.
     The contents of the registers are:

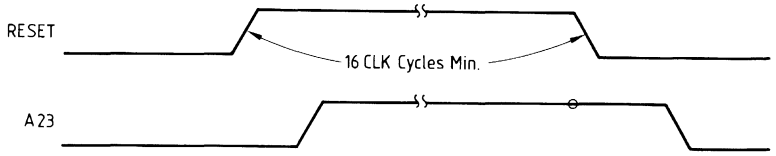
GMR:	all bits are zero
GCR:	undefined
GBR:	all bits are zero
GDR:	all bits are zero
GSR:	– DMST bits for all channels: 0 (MSB) X (LSB) – INT bit for all channels: 0 – S/R bit for all channels: 0
CPRn:	undefined
SPRn:	undefined
DPRn:	undefined
LPRn:	undefined
BCRn:	undefined
CCRn:	undefined
CSRn:	all bits are zero
DARn:	undefined

The leading edge of the RESET signal is used to synchronize the internal clock generator in the same manner as the SAB 80286 microprocessor. Therefore the processor cycles of SAB 82257 and processor are identical.

To continue operating the SAB 82257 in 286 mode after a reset sequence it is necessary that the A23 pin is high at the falling edge of RESET and that the **RM bit** in the general mode register (GMR) (written first after RESET) **is low**.

This operating mode is valid until the next activation of RESET. The RESET signal must be activated for at least 16 CLK cycles.

Figure 25
286 Mode after RESET



4.1.2 186/8086 Mode

4.1.2.1 Bus Cycles

The internal hardware of the SAB 82257 communicates with the CPU and the external hardware via the bus using bus cycles.

Cycle Definitions

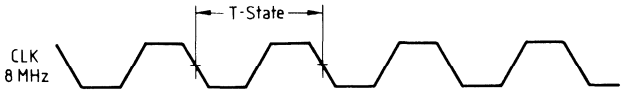
Three types of cycles are used to describe the SAB 82257 operation:

- CLK cycle
- processor cycle and T-state
- bus cycle

CLK Cycle
A CLK cycle is one period of the clock signal at the SAB 82257 CLK pin. In standard systems, the frequency of CLK may be 8 MHz.

Processor Cycle and T-State
The internal SAB 82257 logic is driven from an internal clocking system. One period of this internal clock is called a T-state. In standard systems, the frequency of the clocking system is 8 MHz.
In 186/8086 mode, the clocking system uses the CLK signal directly. Therefore in 186/8086 mode, CLK and T-states are identical.

Figure 26
T-State Definition, 186/8086 Mode



Bus Cycle

A bus cycle consists of an integer number of T-states. In 186/8086 mode, a bus cycle has a minimum length of 4 T-states. Thus, a transfer rate of up to 2 million bus cycles or 4 megabytes per second is possible. The bus cycles are, of course, compatible with the SAB 80186 or the SAB 8086 microprocessors. A bus cycle can be lengthened beyond the minimum by a delayed activation of the AREADY signal or the SREADY signal.

Bus Cycle Types

There are two basic types of bus cycles:

- active bus cycles
- passive bus cycles

Active Bus Cycles

For data transfers between the SAB 82257 and memory or peripherals the SAB 82257 generates active bus cycles by activating the status signals $\overline{S0}$, $\overline{S1}$ and $\overline{S2}$. The SAB 82257 must be master of the local bus and control such a bus cycle. Such a bus cycle can be a read from or a write to memory space or I/O space. A normal DMA transfer needs at least two bus cycles:

- the read from source and
- the write to destination.

These two operations can be combined in a single bus cycle, for higher transfer rates (single-cycles transfer). Source or destination is accessed by the SAB 82257 address and the command, like a normal bus cycle, whereas the peripheral is selected by the $\overline{DACKn}$ signal. The data does not flow through the SAB 82257 but directly from source to destination via the data bus.

Therefore, even for a write cycle data pins must float like for a read cycle. Because the SAB 82257 must control the bus cycle, the other bus pins function like during a normal bus cycle.

Note
For correct operation on a buffered data bus, the direction of the data transceiver must be controlled by the $\overline{DACKn}$ signal.

Passive Bus Cycles

If the SAB 82257 is not the master of the local bus, then the CPU can generate a bus cycle which accesses one of the internal registers of the SAB 82257. The SAB 82257 monitors all bus cycles and, when it is selected via $\overline{CS}$, it executes the data transfer from the local bus into the addressed SAB 82257 register or vice versa, as requested.

When the SAB 82257 and a processor share the local bus, then a passive bus cycle is normally a synchronous access to an SAB 82257 register, i.e. the processor activates status line $\overline{S0}$ or $\overline{S1}$. The SAB 82257 monitors the status lines of the local bus and, when selected via $\overline{CS}$, it generates internal read and write signals having the same timing as the signals generated by the support chips. The end of this passive bus cycle is defined by the trailing edge of the status signals in 186/8086 mode. During execution of a passive cycle the address signals AD7 to AD0, $\overline{BHE}$ signal and $\overline{CS}$ signal are latched. The execution of a synchronous passive cycle is fast enough, so no wait states are necessary.

If no activation of status lines $\overline{S0}$ or $\overline{S1}$ is detected, the SAB 82257 monitors the $\overline{RD}$ and $\overline{WR}$ signals. On activation of one of these signals, the SAB 82257 latches the address AD7 to AD0, $\overline{BHE}$ signal and $\overline{CS}$ signal. After the synchronization of the asynchronous signals $\overline{RD}$ and $\overline{WR}$, the SAB 82257 begins an asynchronous passive bus cycle. The end of this bus cycle is defined by the trailing edge of the $\overline{RD}$ or $\overline{WR}$ signal.

Note
An asynchronous passive bus cycle requires more time than a synchronous one, because the synchronization needs additional time.

When a passive bus cycle is lengthened beyond the minimum time, a read from an SAB 82257 register is made once and the result is latched to the output. Therefore, a change of register contents during a passive bus cycle will not change the data read. By lengthening a passive write bus cycle, the transfer of the data on the pins to the selected SAB 82257 register is repeated until the end of the bus cycle. Therefore, only the last data on the data pins is stored in the register.

When the SAB 82257 is not in possession of the bus, all output signals except the following are tristated:

186 Mode	8086 Mode
HOLD $\overline{DACK0}$ to $\overline{DACK3}$ $\overline{EOD0}$ to $\overline{EOD3}$ ALE	$\overline{DACK0}$ to $\overline{DACK3}$ $\overline{EOD0}$ to $\overline{EOD3}$ ALE

Bus Cycle Status

In 186/8086 mode, there are three signals to indicate the bus cycle type:

- $\overline{S0}$ signal
- $\overline{S1}$ signal
- $\overline{S2}$ signal

The beginning of a bus cycle is indicated by $\overline{S0}$ or $\overline{S1}$ or both going active. The termination of a bus cycle is indicated by all status signals going inactive.

When the SAB 82257 **is** the master of the local bus, it can generate the following bus cycles by activating $\overline{S0}$, $\overline{S1}$ and $\overline{S2}$:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Active Bus Cycle Type
0	0	0	Not Valid
0	0	1	Read from I/O Space
0	1	0	Write into I/O Space
0	1	1	Does Not Occur
1	0	0	Does Not Occur
1	0	1	Read from Memory Space
1	1	0	Write into Memory Space
1	1	1	Not a Bus Cycle

Note
In 186/8086 mode, the $\overline{S2}$ signal can be active only if at least either $\overline{S0}$ or $\overline{S1}$ is active.

When the SAB 82257 **is not** the master of the local bus, the status signals are used as inputs for detection of synchronous accesses to the SAB 82257.

The following table shows the bus status and $\overline{CS}$ signals and their interpretation by the SAB 82257:

$\overline{CS}$	$\overline{S1}$	$\overline{S0}$	Description
1	X	X	SAB 82257 is Not Selected, No Action
0	0	0	No SAB 82257 Access, No Action
0	0	1	Read from an SAB 82257 Register
0	1	0	Write into an SAB 82257 Register
0	1	1	Not a Bus Cycle

Note
The SAB 82257 is selected but no synchronous access is activated. In such a case the SAB 82257 monitors the $\overline{RD}$ and $\overline{WR}$ signals for detection of an asynchronous access. In 186/8086 mode, the higher address part is multiplexed with additional status information (A16/S3 to A19/S6).

When the SAB 82257 **is** the master of the local bus the following table shows the additional status information:

S6	S5	S4	S3	Channel Number for the Running Bus Cycle
1	0	0	0	0
1	0	0	1	1
1	0	1	0	2
1	0	1	1	3

Note
During an active bus cycle, S6 is always "high" and S5 is always "low". They can be used to distinguish processor bus cycles from SAB 82257 bus cycles.

When the SAB 82257 **is not** the master of the local bus the following table shows the additional status information:

S4	S3	Channel Number for the Running Bus Cycle
0	0	0
0	1	1
1	0	2
1	1	3

Bus High Enable ($\overline{BHE}$)

The SAB 82257 works on a 16-bit bus as well as on an 8-bit bus. To control transfers of bytes or words on such buses, there are two signals:

- bus high enable ($\overline{BHE}$) and
- the least significant address bit (A0).

A low on the $\overline{BHE}$ signal means that the data on the higher part of the bus (AD15 to AD8) is valid.
A low on the A0 signal means that the data on the lower part of the bus (AD7 to AD0) is valid. Therefore an even-addressed byte is transferred on the lower data bus byte and an odd-addressed byte is transferred on the higher data bus byte. A full word can be transferred by using both data bus bytes (AD15 to AD0) and addressing the lower (even) byte of the word in conjunction with an active $\overline{BHE}$ signal.

For transfers on an 8-bit physical bus both kinds of bytes, the even-addressed and the odd-addressed byte, must be transferred over the lower data bus part. The transfer of an even addressed byte uses the same coding as a byte transfer on a 16-bit bus. The transfer on the odd-addressed byte on the lower data bus half is indicated by $\overline{BHE}$ being "high" and A0 being "high".

A summary of this description is shown in the following table:

$\overline{BHE}$	A0	Valid Data Bus Pins	Used for 16-Bit Bus	Used for 8-Bit Bus	Notes
0	0	AD15 to AD0	Yes	No	Word Transfer
0	1	AD15 to AD8	Yes	No	Odd-Addressed Byte on 16-Bit Bus
1	0	AD7 to AD0	Yes	Yes	Even-Addressed Byte
1	1	AD7 to AD0	No	Yes	Odd-Addressed Byte on 8-Bit Bus

The physical bus width is indicated by the **MEMBUS bit** and the **IOBUS bit** in the general mode register (GMR) for SAB 82257 bus cycles. When an access to SAB 82257 register is made (passive bus cycle), the SAB 82257 handles the data the correct way by sensing the $\overline{BHE}$ and A0 signal and acting according to the above table.

Bus Ready

When a bus cycle must be lengthened beyond the minium length, e.g. for supporting slower memories or peripherals, this must be done with the bus ready signals. If the bus ready signal is not active at the sampling time, the bus cycle is lengthened by one T-state. The bus ready signal is also sampled one T-state later for additional lengthening of the bus cycle. Therefore the bus cycle length can be extended indefinitely by an integer number of T-states. In 186/8086 mode, there are two signals for indicating the ready status of the bus:

- **SREADY**, which is a synchronous input with the demand of meeting setup and hold times, and
- **AREADY**, which is an asynchronous input and the synchronization is done internally (see data sheet).

Only one of the bus ready signals needs to be activated for indicating the rady status of the bus. When the SAB 82257 detects ready active it terminates the bus cycle by deactivating the status signals.
All other action necessary for termination of a bus cycle is started from the trailing edges of the status signals.

4.1.2.2 Bus Arbitration

186 Mode

For the arbitration of the bus there are two signals:

- HOLD signal
- HLDA signal (HOLD acknowledge signal)

In local configurations these lines are directly connected to the processor. Whenever the SAB 82257 wants to perform one or more bus cycles it activates the HOLD signal. The processor receives this and surrenders the local bus as soon as possible. The processor tristates its bus and acknowledges the request with a high on the HLDA signal. After sensing an active HLDA the SAB 82257 gets onto the bus after a minimum of 2 CLK cycles to perform the bus cycles. After completion of the last bus cycle the SAB 82257 will surrender the local bus by floating the bus and switching the HOLD signal low. The processor acknowledges this by a low on the HLDA signal and takes the bus.

The SAB 82257 can be forced off the bus if HLDA is made inactive. After "losing HLDA" the SAB 82257 surrenders the bus

- after the currently running bus cycle, or
- after the bus cycle(s) following the running bus cycle.

The second case is for supporting inseparable bus cycle, which are:

- word transfer on odd addresses, which is realized by two bus cycles where each transfer is a byte transfer,
- fetch of 24-bit address pointers out of memory.

The SAB 82257 signals the surrendering of the bus by floating the bus and removing the HOLD signal.

If requests for bus cycles are present the HOLD signal will go active after a delay of two T-states.

Figure 27
HOLD-HLDA Sequence, 186 Mode

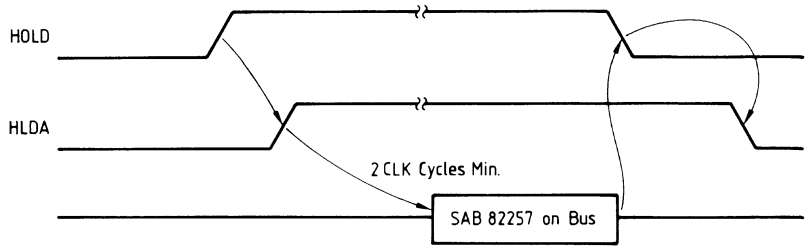
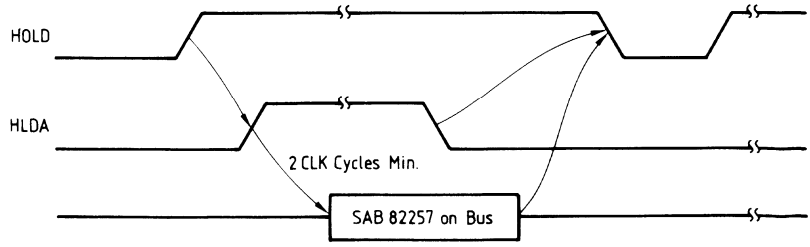


Figure 28
Losing HLDA, 186 Mode



8086 Mode

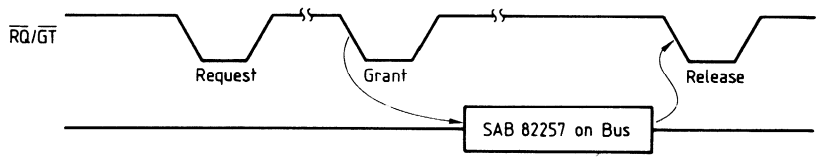
In 8086 mode, the bus arbitration is done via the

- $\overline{RQ}/\overline{GT}$ protocol.

The REQUEST/GRANT protocol implements a one-line communication dialog between the SAB 82257 and the processor. Whenever the SAB 82257 wants to perform one or more bus cycles it sends a request pulse lasting one CLK period via the $\overline{RQ}/\overline{GT}$ signal to the processor. The processor receives this and it will acknowledge this request also with a pulse on this line. After sensing that grant pulse, the SAB 82257 controls the bus and uses it to perform bus cycles. If no further bus cycles are necessary after bus cycle completion, the SAB 82257 will surrender the local bus by sending a release pulse on the $\overline{RQ}/\overline{GT}$ line. The processor receives that signal and takes the bus.

Note
In 8086 mode, the HLDA pin has no function.

Figure 29
Request-Grant Sequence, 8086 Mode



4.1.2.3 Reset Sequence

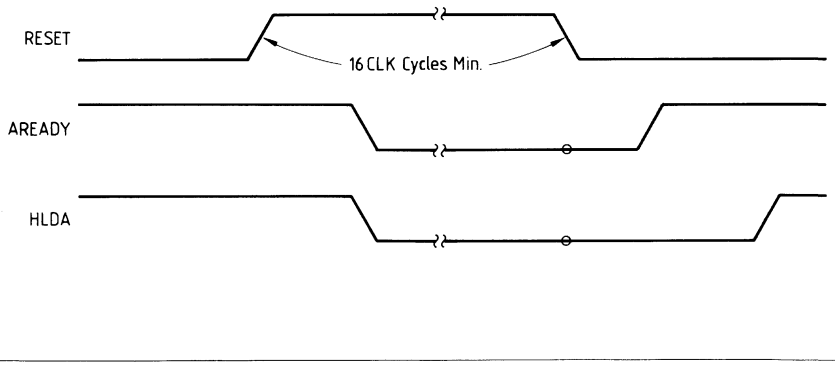
A reset signal on pin RESET (high level), forces the SAB 82257 into a well defined initial state. This state is characterized as follows:

1. **Upon activation** of the RESET signal
   - all channels are stopped
   - all bus activities are stopped
   - all tristate signals are in tristate and others in passive state.
2. **After activation** of the RESET signal
   - above state is maintained, and in addition:
   - some registers have defined values or defined control bits.
     The contents of the registers are:
     GMR: all bits are zero
     GCR: undefined
     GBR: all bits are zero
     GDR: all bits are zero
     GSR: – DMST bits for all channels: 0 (MSB) × (LSB)
     – INT bit for all channels: 0
     – S/R bit for all channels: 0

CPRn:	undefined
SPRn:	undefined
DPRn:	undefined
LPRn:	undefined
BCRn:	undefined
CCRn:	undefined
CSRn:	all bits are zero
DARn:	undefined

186 Mode
To continue operating the SAB 82257 in 186 mode after a reset sequence, it is necessary that the AREADY pin and HLDA pin are low at the falling edge of RESET.

Figure 30
186 Mode after RESET

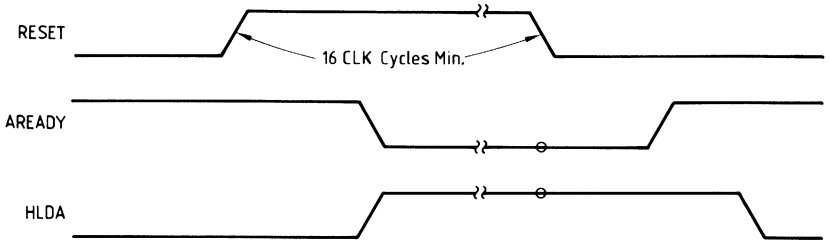


This operating mode is valid until the next activation of RESET. The RESET signal must be activated for at least 16 CLK cycles.

8086 Mode
To continue operating the SAB 82257 in 8086 mode after a reset sequence, it is necessary that the AREADY pin is low and HLDA pin is high at the falling edge of RESET.

This SAB 82257 operating mode is valid until the next activation of RESET. The RESET signal must be activated for at least 16 CLK cycles.

Figure 31
8086 Mode after RESET



4.2 Remote-Mode Bus Operations

4.2.1 Bus Cycles

The internal hardware of the SAB 82257 communicates with the CPU and the external hardware via the bus using bus cycles.

Cycle Definitions

Three types of cycles are used to describe the SAB 82257 operation:

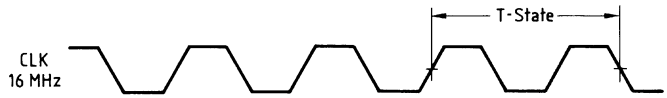
- CLK cycle
- processor cycle and T-state
- bus cycle

CLK Cycle
A CLK cycle is one period of the clock signal at the SAB 82257 CLK pin. In standard systems, the frequency of CLK may be 16 MHz.

Processor Cycle and T-State
The internal SAB 82257 logic is driven from an internal clocking system. One period of this internal clock is called a T-state. In standard systems, the frequency of the clocking system is 8 MHz.

In remote mode the clocking system uses the CLK frequency divided by two. Therefore only the falling edge of CLK is important for the SAB 82257 timing and a T-state equals two CLK periods.

Figure 32
T-State Definition, Remote Mode



Bus Cycle
A bus cycle consists of an integer number of T-states. In remote mode a bus cycle has a minimum length of 3 T-states but due to the pipelining of bus cycles, these bus cycles, may come at a rate of one bus cycle for every two T-states. Therefore the maximum bus rate is 4 million bus cycles or 8 megabytes per second.

A bus cycle can be lengthened beyond the minimum by a delayed activation of the ready signal ($\overline{\text{READY}}$).

Bus Cycle Types

There are two basic types of bus cycles:

- active bus cycles
- passive bus cycles

Active Bus Cycles
For data transfer between the SAB 82257 and memory or peripherals the SAB 82257 generates active bus cycles by activating the status signals $\overline{S0}$ and $\overline{S1}$. In remote mode the SAB 82257 is the only master of local/resident bus. Therefore the SAB 82257 can start such active bus cycles immediately, this means without any bus arbitration. For system bus accesses a special arbitration control of the local bus is necessary to prevent a deadlock situation (see section 4.2.2 Bus Arbitration).

An active bus cycle can be a read from or a write to system space or resident space.

A normal DMA transfer needs at least two bus cycles:

- the read from source and
- the write to destination.

These two operations can be combined in a single bus cycle, for higher transfer rates (single-cycle transfer). Source or destination is accessed by the SAB 82257 address and the command, like a normal bus cycle, whereas the peripheral is selected by the $\overline{\text{DACKn}}$ signal. The data does not flow through the SAB 82257 but directly from source to destination via the data bus.

Therefore even for a write cycle data pins must float like for a read cycle. Because the SAB 82257 must control the bus cycle, the other bus pins function like during a normal bus cycle.

Note
For correct operation on a buffered data bus the direction of the data transceiver must be controlled by the $\overline{\text{DACKn}}$ signal.

Passive Bus Cycles
In remote mode, a passive bus cycle is an **asynchronous** access to an SAB 82257 register, i.e. the processor drives signals $\overline{\text{CS}}$ and $\overline{\text{WR}}$ or $\overline{\text{RD}}$. After receiving this access request the SAB 82257 releases its local bus as soon as possible and signals this by activating the BREL line. Now it monitors the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals. Upon activation of one of these signals the SAB 82257 latches the address signals A7 to A0, $\overline{\text{BHE}}$ signal and $\overline{\text{CS}}$ signal. After synchronization of the asynchronous signals $\overline{\text{RD}}$ and $\overline{\text{WR}}$, the SAB 82257 begins an asynchronous passive bus cycle. The end of this bus cycle is defined by the trailing edge of the $\overline{\text{RD}}$ or $\overline{\text{WR}}$ signal.

When a passive bus cycle is lengthened beyond the minimum time, a read from an SAB 82257 register is made once and the result is latched to the output. Therefore, a change of register contents during a passive bus cycle will not change the data read. By lengthening a passive write bus cycle the transfer of the data on the pins to the selected SAB 82257 register is repeated until the end of the bus cycle. Therefore only the last data on the data pins is stored in the register.

When the SAB 82257 is not in possession of the bus, all output signals except the following are tristated:

- HOLD
- $\overline{\text{DACK0}}$ to $\overline{\text{DACK3}}$
- $\overline{\text{EOD0}}$ to $\overline{\text{EOD3}}$
- BREL

Bus Cycle Status

In remote mode, there are basically two signals to indicate the bus cycle type:

- $\overline{\text{S0}}$ signal and
- $\overline{\text{S1}}$ signal

The beginning of a bus cycle in indicated by $\overline{\text{S0}}$ or $\overline{\text{S1}}$ or both going active. The termination of a bus cycle is indicated by the bus ready signal ($\overline{\text{READY}}$) going active.

Since in remote the SAB 82257 initiates all system bus accesses (and only these) with a HOLD/HLDA sequence (see section 4.2.2 Bus Arbitration), the

- HOLD signal

can be used for distinction between the two address spaces (resident space and system space).

Therefore the SAB 82257 can generate the following bus cycles by activating status signals $\overline{\text{S0}}$ and $\overline{\text{S1}}$:

HOLD	$\overline{S1}$	$\overline{S0}$	Active Bus Cycle Type
0	0	0	Not Valid
0	0	1	Read from Resident Space
0	1	0	Write into Resident Space
0	1	1	Not a Bus Cycle
1	0	0	Does Not Occur
1	0	1	Read from System Space
1	1	0	Write into System Space
1	1	1	Not a Bus Cycle

Bus High Enable ($\overline{BHE}$)

The SAB 82257 works on a 16-bit bus as well as on an 8-bit bus. To control transfers of bytes or words on such buses, there are two signals:

- bus high enable ($\overline{BHE}$) and
- the least significant address bit (A0).

A low on the $\overline{BHE}$ signal means that the data on the higher part of the bus (D15 to D8) is valid. A low on the A0 signal means that the data on the lower part of the bus (D7 to D0) is valid. Therefore an even-addressed byte is transferred on the lower data bus byte and an odd-addressed byte is transferred on the higher data bus byte. A full word can be transferred by using both data bus bytes (D15 to D0) and addressing the lower (even) byte of the word in conjunction with an active $\overline{BHE}$ signal.

For transfers on an 8-bit physical bus both kinds of bytes, the even-addressed and the odd-addressed byte, must be transferred over the lower data bus part. The transfer of an even-addressed byte uses the same coding as a byte transfer on a 16-bit bus. The transfer on the odd-addressed byte on the lower data bus half is indicated by $\overline{BHE}$ being high and A0 being high.

A summary of this description is shown in the following table:

$\overline{BHE}$	A0	Valid Data Bus Pins	Used for 16-Bit Bus	Used for 8-Bit Bus	Notes
0	0	D15 to D0	Yes	No	Word Transfer
0	1	D15 to D8	Yes	No	Odd-Addressed Byte on 16-Bit Bus
1	0	D7 to D0	Yes	Yes	Even-Addressed Byte
1	1	D7 to D0	No	Yes	Odd-Addressed Byte on 8-Bit Bus

The physical bus width is indicated by the **SYSBUS bit** and the **RESBUS bit** in the general mode register (GMR) for SAB 82257 bus cycles. When an access to a register is made (passive bus cycle) the SAB 82257 handles the data in the correct way by sensing the $\overline{BHE}$ and A0 signal and acting according to the above table.

Bus Ready

When a bus cycle must be lengthened beyond the minimum length for supporting slower memories or peripherals, this must be done with the bus ready signal. If the bus ready signal is not active at the sampling time, the bus cycle is lengthened by one T-state. The bus ready signal is also sampled one T-state later for additional lengthening of the bus cycle. Therefore the bus cycle length can be extended indefinitely by an integer number of T-states.

In remote mode the signal for indicating the ready status of the bus is:

- $\overline{\text{READY}}$

  which is a synchronous input with the demand of meeting setup and hold times specified in the data sheet.

Any activation of the $\overline{\text{READY}}$ signal which does not meet the setup and hold times may cause erroneous operation. Therefore, the SAB 82284 clock generator should be used guaranteeing the correct activation of $\overline{\text{READY}}$ signal.

An active bus ready signal occurring at the sampling time terminates the bus cycle.

4.2.2 Bus Arbitration

In remote mode the SAB 82257 is the only master of the local/resident bus. Therefore it can start the local bus cycles immediately, this means without any bus arbitration by the HOLD/HLDA sequence. For system bus accesses a special arbitration control of the local bus is necessary to prevent a deadlock situation. A deadlock would arise if at the same time

- the SAB 82257 waits to access the system bus thereby occupying the local bus, and
- the CPU waits to access the SAB 82257 (via the SAB 82257's local bus) thereby occupying the system bus.

This problem can only be solved whenever, in the case of system bus accesses, the SAB 82257 does not occupy the local bus until it has control of the system bus. Therefore the SAB 82257 in remote mode initiates all system bus accesses (and only these) with a HOLD/HLDA sequence.

HOLD/HLDA Sequence

For arbitration of the system bus there are two signals:

- HOLD signal
- HLDA signal (HOLD acknowledge signal)

In autonomous configurations, these lines are connected to the bus arbiter.

When the SAB 82257 is in remote mode, the HOLD/HLDA sequence is defined as follows:

1. For access to the resident bus, the SAB 82257 does not generate HOLD and starts the access without receiving HLDA.
2. For all accesses to the system bus, the SAB 82257 generates HOLD before getting onto the local bus. Only when it gets a HLDA it starts the bus cycle and occupies its local bus. This ensures a deadlock-free arbitration of the local bus.

The CPU is only allowed to occupy the local bus if this bus is not accessed by the SAB 82257. Therefore an arbitration of the local bus is necessary.

$\overline{CS}$/BREL Sequence

For the arbitration of the local bus there are two signals:

- $\overline{CS}$ signal
- BREL signal (bus release signal).

Whenever the processor needs access to the local bus of the SAB 82257, it forces the $\overline{CS}$ signal low. After receiving $\overline{CS}$, the SAB 82257 surrenders the local bus

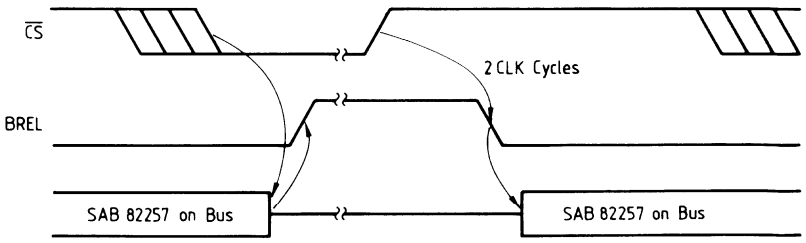
- immediately if no bus cycle is running, or
- after the currently running bus cycle, or
- after the bus cycle(s) following the running bus cycle.

The third case is for supporting inseparable bus cycles which are:

- word transfers on odd addresses, which are realized by two bus cycles where each transfer is a byte,
- the fetch of 24-bit address pointers out of memory.

The SAB 82257 signals the surrendering of the bus by floating the bus and activating the BREL signal. Now an access to the SAB 82257 registers or to the SAB 82257's resident bus can be made. After that the $\overline{CS}$ signal must be set high. The SAB 82257 responds to this by deactivating the BREL signal. Now the local bus is free for bus cycles of the SAB 82257.

Figure 33
CS-BREL Sequence, Remote Mode



4.2.3 Reset Sequence

A reset signal on pin RESET (high level) forces the SAB 82257 into a well defined initial state.

This state is characterized as follows:

1. **Upon activation** of the RESET signal
   - all channels are stopped
   - all bus activities are stopped
   - all tristate signals are in tristate and others in passive state.
2. **After activation** of the RESET signal
   - above state is maintained, and in addition:
   - some registers have defined values or defined control bits.

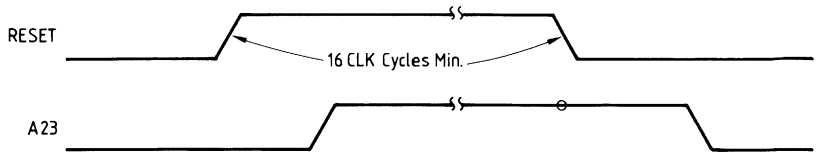
   The contents of the registers are:

GMR:	all bits are zero
GCR:	undefined
GBR:	all bits are zero
GDR:	all bits are zero
GSR:	– DMST bits for all channels: 0 (MSB) × (LSB) – INT bit for all channels: 0 – S/R bit for all channels: 0
CPRn:	undefined
SPRn:	undefined
DPRn:	undefined
LPRn:	undefined
BCRn:	undefined
CCRn:	undefined
CSRn:	all bits are zero
DARn:	undefined

To continue operating the SAB 82257 in remote mode after a reset sequence, it is necessary that the A23 pin is high at the falling edge of RESET and that the **RM bit** in the general mode register (written first after RESET) **is high.**

Note
After RESET, the SAB 82257 is programmed to local mode (after RESET all bits of the GMR are zero, i.e. the RM bit is low). An access to the SAB 82257 general mode register is possible because the BREL pin is in tristate after RESET. Connecting a pullup resistor, this signal is active for the external circuit and thus allows access to the SAB 82257. By this access the SAB 82257 is programmed for remote mode, i.e. RM = 1, via the general mode register (GMR).

Figure 34
Remote Mode after RESET



This SAB 82257 operating mode is valid until the next activation of RESET. The RESET signal must be activated for at least 16 CLK cycles.

Communications Mechanism

5 Communications Mechanism

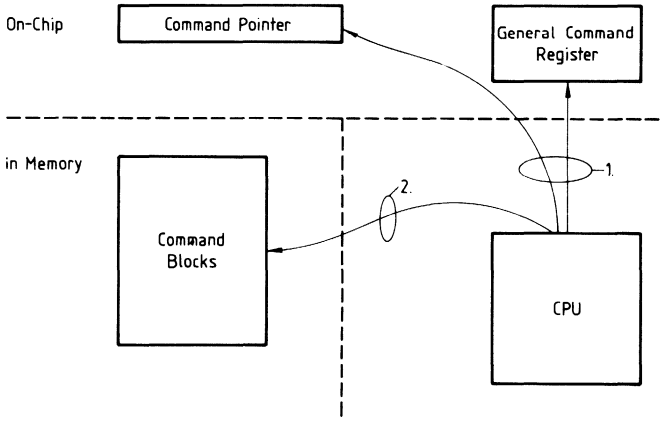
5.1 CPU/SAB 82257 Communication

The CPU communicates with the SAB 82257 by depositing data in memory and into on-chip registers. The CPU can access the SAB 82257's general registers and status registers and can start up a channel by writing the proper command into the general command register. The SAB 82257 will then read data from memory command blocks and set up itself accordingly.

Thus the communication between CPU and SAB 82257 is two-folded:

- communication by direct register access via the slave interface of the SAB 82257 and
- communication via the control space in memory.

Figure 35
CPU/SAB 82257 Communication



1. Communication by direct register access
2. Communication via the control space in memory

5.1.1 Communication via Control Space in Memory

The normal communication between the CPU and the SAB 82257 is the data transfer via the control space in memory (**memory-based communication**). This means that all necessary information for a transfer with all its modifications, as for example, addresses of data source and data destination, block length (byte count) or a list pointer for data chaining (instead of source or destination pointer) is contained in a command block in memory accessible to the CPU and the SAB 82257. The control space consists of all the organizational blocks like command blocks, chain list etc.

The control space can lie

- **for local mode**: in the memory space as well as in the I/O space, and
- **for remote mode**: either in the system space or in the resident space.

The particular control space can be dynamically changed with every start channel command.

5.1.2 Communication via Slave Interface

Although nearly all of the necessary communication between CPU and SAB 82257 is done via memory-based data blocks, some direct accesses to SAB 82257 registers are necessary. For example, during the initialization phase the general mode register must be written, or to start a channel the command pointer register and the general command register must be loaded. Also during the debugging phase it is of great benefit to have direct access to all of the SAB 82257's internal registers. This direct access to registers (read/write registers) is provided by the **slave interface** of the SAB 82257.

The slave interface consists of the following lines:

- $\overline{S0}$, $\overline{S1}$: status lines (inputs)
- $\overline{RD}$, $\overline{WR}$: control lines (inputs)
- A0 to A7: register addresses (inputs)
- D0 to D15: data lines (inputs/outputs)

And for synchronous access in 186 mode:

- AD0 to AD15: address/data lines (inputs/outputs).

The slave interface can only be accessed by the CPU, if the SAB 82257 does not occupy the local bus. Therefore a local bus arbitration is necessary. This arbitration is

- **in local mode**: performed by the CPU (HOLD/HLDA sequence) and
- **in remote mode**: done by the SAB 82257 itself ($\overline{CS}$/BREL sequence).

The slave interface can be synchronous or asynchronous to the processor. The following table presents a survey.

Access Type	Address Information Expected at Lines	Possible in 286 Mode	Possible in Remote Mode	Possible in 186/8086 Mode
Synchronous	A0 to A7	Yes	No	–
Synchronous	AD0 to AD7	–	No	Yes
Asynchronous	A0 to A7	Yes	Yes	Yes

Note

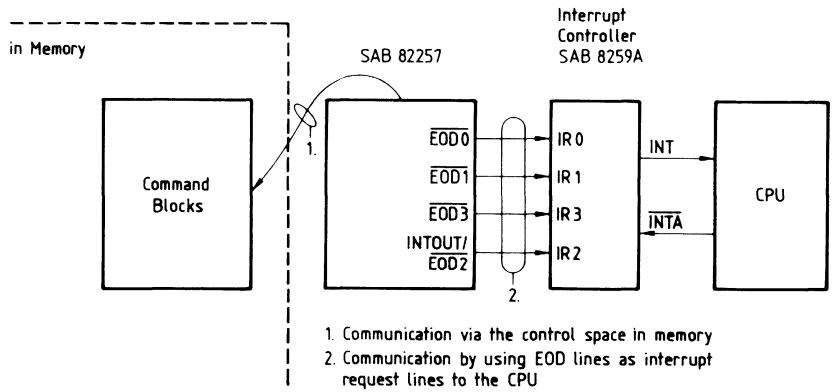
- For synchronous access, processor and SAB 82257 must be directly coupled and use the same clock.
- In an autonomous SAB 82257 subsystem (remote mode), only the asynchronous access is possible because SAB 82257 has to release its local bus first to enable the register access. On receiving an access request (activation of $\overline{CS}$ input), the SAB 82257 releases his local bus as soon as possible and signals this by activating the BREL line. Now the CPU can accomplish its access. (A detailed description of this arbitration is given in the subsequent sections).
- All the lines the slave interface consists of are outputs if the SAB 82257 is an active bus master.

5.2 SAB 82257/CPU Communication

The SAB 82257 communicates with the CPU by depositing data at the end of each DMA transfer in the control space in memory and by using $\overline{EOD}$ lines as interrupt request lines to the CPU. Thus the communication between SAB 82257 and CPU can also use two ways:

- Communication by using $\overline{EOD}$ lines as interrupt request lines to the CPU (hardware-based communication)
- communication via the control space in memory (memory-based communication).

Figure 36
SAB 82257/CPU Communication



5.2.1 Memory-Based Communication

As described in section 5.1.1, control-space based communication means that all information necessary for DMA transfers in contained in command blocks in memory accessible to the SAB 82257 and the CPU.

Saving the status on termination the SAB 82257 writes the contents of the appropriate channel status register (on-chip-register) into these command blocks at the end of each DMA transfer examination by the CPU.

5.2.2 Hardware-Based Communication

The SAB 82257 has 4 $\overline{\text{EOD}}$ pins (one for each channel) for CPU interrupt and for communication with the system environment. Since the $\overline{\text{EOD}}$ pins are multiple function pins, their application can be programmed. Thereby input and output functions have to be distinguished.

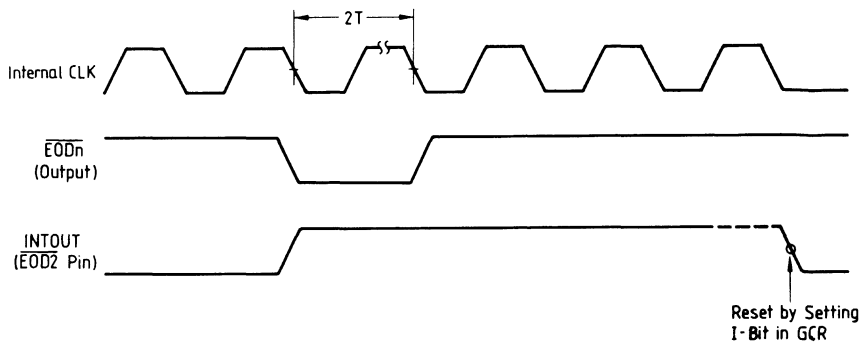
In this section, only the output functions of the $\overline{\text{EOD}}$ pins are discussed. Two basic functions have to be distinguished:

- $\overline{\text{EOD}}$ function ("end of DMA")
- INTOUT function ("interrupt output").

$\overline{\text{EOD}}$ is a channel-specific active-low pulse signal lasting 2 T-states. It is always enabled by software.

INTOUT can be hardware-generated (error detection) or enabled by software. The channel which is generating INTOUT is indicated in the general status register (GSR) by the channel's INT (Interrupt) bit. INTOUT remains active until all INT bits in the general status register are reset by the CPU with general commands ("clear interrupt").

Figure 37
$\overline{\text{EOD}}$/INTOUT Signals



With INTOUT or $\overline{EOD}$, the CPU (or other components of systems environment) is synchronized to channel-specific events, as for example

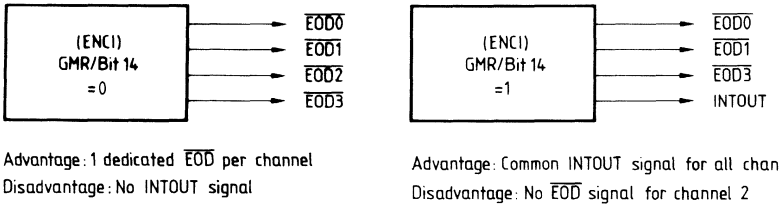
- the channel is stopped because of a fatal error condition (INTOUT)
- a block transfer is terminated by exceeded byte count ($\overline{EOD}$)
- a certain point in the channel program execution has been reached ($\overline{EOD}$ or INTOUT)
- channel program execution is finished and channel is stopped ($\overline{EOD}$ or INTOUT).

The SAB 82257 provides two $\overline{EOD}$ and INTOUT signal options:

- The interrupts of channels are issued **channel-specific** as $\overline{EOD}$ signals.
- The interrupts of channels are **common for all channels** as INTOUT signals on the $\overline{EOD2}$ pin ($\overline{EOD}$ pin of channel 2), the other three $\overline{EOD}$ pins may be used as $\overline{EOD}$ output described above.

Both options have advantages and disadvantages. System constraints decide which option is the better one.

Figure 38
$\overline{EOD}$/INTOUT Signal Options



Advantage: 1 dedicated $\overline{EOD}$ per channel
Disadvantage: No INTOUT signal

Advantage: Common INTOUT signal for all channels
Disadvantage: No $\overline{EOD}$ signal for channel 2

5.3 SAB 82257/Peripheral Communication

The SAB 82257/peripheral communication uses the **DMA interface,** which consists of the following lines:

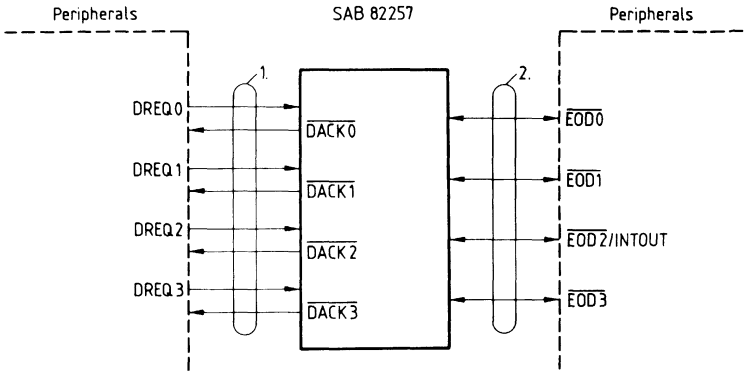
- DREQn ("DMA request") lines,
- $\overline{DACKn}$ ("DMA acknowledge") lines and
- bidirectional $\overline{EODn}$ ("end of DMA") signals.

5.3.1 Communication via DREQn/$\overline{DACKn}$ Signals

These lines work as request and acknowledge lines to control synchronized DMA transfers as known from conventional DMA controllers.

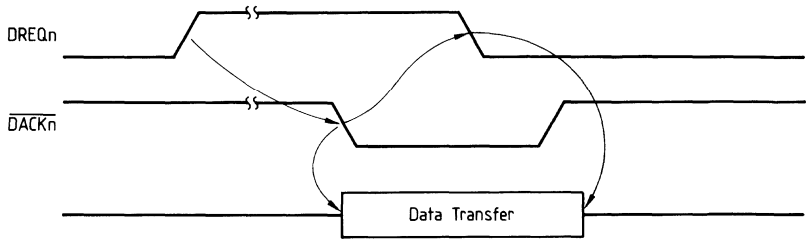
Before accessing a synchronizing device, the SAB 82257 waits for an activation of the DREQn signal. After sensing an active DREQn, the SAB 82257 starts the data transfer as soon as possible and acknowledges the request by activating the $\overline{DACKn}$ signal. A burst of data is transferred in case of a continuous DMA request, as long as the DREQn signal is active.

Figure 39
SAB 82257/Peripheral Communication



1. Communication via DREQn/$\overline{\text{DACKn}}$ signals
2. Communication via bidirectional $\overline{\text{EODn}}$ signals

Figure 40
DREQ/$\overline{\text{DACK}}$ Sequence



DREQ/$\overline{\text{DACK}}$ Sequence

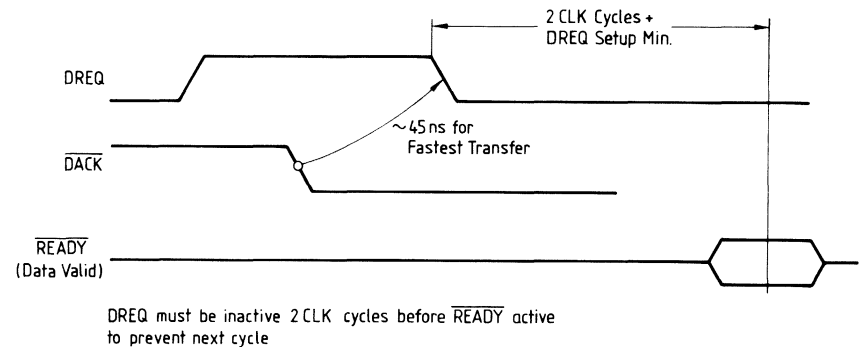
A DMA request initiates the execution of one or, in the case of continuous request, several DMA cycles. To synchronize the data transfers there are two pins for each channel:

- DREQn
- $\overline{\text{DACK}}$n (n = number of the corresponding channel)

Before accessing a synchronizing device, the SAB 82257 waits for an activation of the DREQn signal. During this wait phase the SAB 82257 can execute transfers on other channels or release the bus for CPU actions. If the SAB 82257 receives an active signal on DREQn line, it starts the first bus cycle for the data transfer as soon as possible. During this bus cycle the SAB 82257 acknowledges the request by activating the $\overline{\text{DACK}}$n signal. For a single data transfer the DREQn signal must go inactive before the SAB 82257 can start a second bus cycle to access the synchronizing device again. This means that the request must disappear at the latest one T-state before the T-state in which the SAB 82257 begins to issue a new status (TS) for the synchronizing device.

If the request is reset at least one and a half T-states before the last abort point (mentioned above), the SAB 82257 can prepare a following bus cycle within this time and therefore can perform the optional pipelining of bus cycles. Otherwise the SAB 82257 must delay the following bus cycle by one or two T-states, which reduces the bus transfer rate.

Figure 41
DREQ/$\overline{\text{DACK}}$ Signals



Summary

- For a single DMA transfer the DREQ signal should be released about 2 CLK cycles before the end of the bus cycle, otherwise the next bus cycle will follow.
- For a burst-mode transfer, DREQ must be kept active during the burst and made inactive 2 CLK cycles before the end of the last bus cycle.

5.3.2 Communication via Bidirectional $\overline{\text{EOD}}$ Lines

A special feature of SAB 82257 are the bidirectional $\overline{\text{EOD}}$ lines.

First they can be used as **outputs** (see section 5.2.2) to send out a pulse which interrupts the CPU and/or signals to the peripheral a specific status as for example transfer aborted or end of a block or send/receive next block, etc. In addition, the $\overline{\text{EOD}}$ output of channel 2 can be used as a collective interrupt output (INTOUT) for all DMA channels while the other three retain their normal function.

Secondly, as an **input,** the $\overline{\text{EOD}}$ lines can be used to receive an asynchronous external signal to terminate a running DMA transfer. For this purpose the $\overline{\text{EOD}}$ lines are forced low by an external circuitry.

An external termination is enabled by the SAB 82257 during the channel status "DMA in progress" as indicated in the general status register (GSR).

Additionally, an external termination is processed only, if it is enabled with the EXT bit in the type 1 channel command. An external termination is indicated in the channel status register and can be sampled by conditional type 2 commands.

With these lines various configurations are possible depending on system resources and requirements. Figures 42, 43 and 44 show some possible $\overline{\text{EOD}}$/INTOUT configurations.

The configuration in figure 42 requires four interrupt request inputs, one for each channel. None of the $\overline{\text{EOD}}$ pins is available as input then (see also figure 36).

The configuration in figure 43 requires only one interrupt request input and three $\overline{\text{EOD}}$ pins are a available for peripheral interface.

The configuration in figure 44 is essentially a combination of the configurations in figure 42 and figure 43.

Figure 42
Channel-Specific $\overline{\text{EOD}}$ Outputs

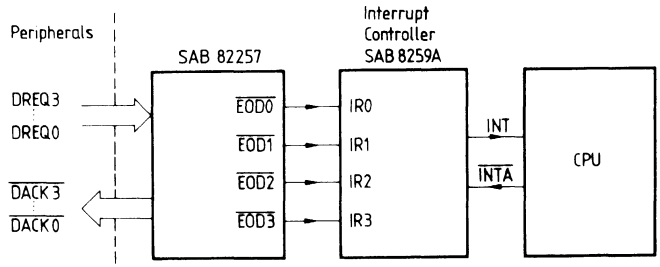


Figure 43
Common INTOUT

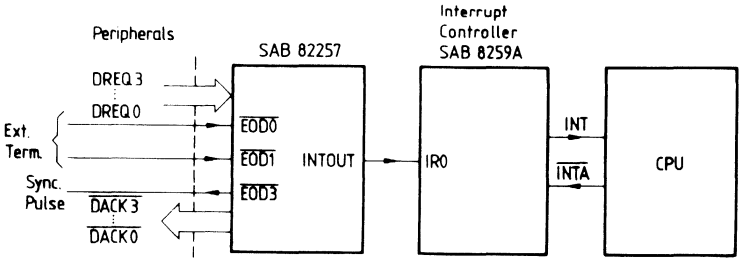
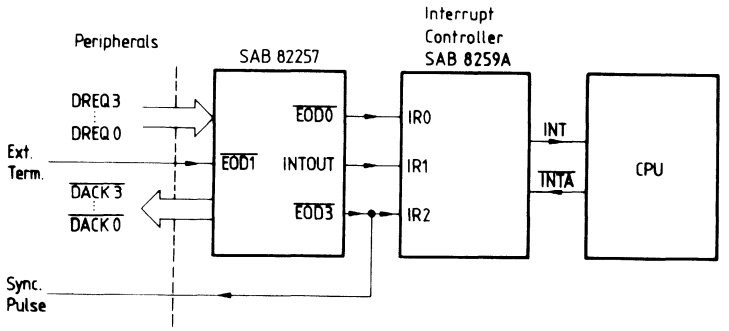


Figure 44
Mixed $\overline{\text{EOD}}$ Configuration



Programming and Control

6 Programming and Control

6.1 Register Model

The SAB 82257 employs a large number of programmable, user-accessible and logically ordered registers to control its operation and maintain address pointers, status information, etc.

These registers are classified as:

- **General registers**:
  - A set of five registers used for all 4 channels.
- **Channel registers**:
  - A set of 32 registers
  - For each channel there is a separate, independent set of 8 channel registers.

All these user-visible registers can be read or loaded by the CPU. Some of the general registers are loaded during initialization after a reset sequence (see section on Bus Arbitration) and others during the invocation of a channel. Also some of the channel registers are programmed or read by the CPU but most of them are loaded by the SAB 82257 itself during the setup routine after a channel start. Therefore most of the SAB 82257's registers are accessed by the CPU only for test purposes. All registers can be accessed bytewise or wordwise by the CPU.

Accessibility and Register Addresses

Although most of the registers are loaded and saved from memory by the SAB 82257 automatically, it is of great benefit to have access to all of the internal registers, e.g. during the debugging phase. This access, reading or writing, is done via the slave interface of the SAB 82257. The slave interface can only be accessed by the CPU, if the SAB 82257 does not occupy the local bus. Therefore, a local bus arbitration is necessary:

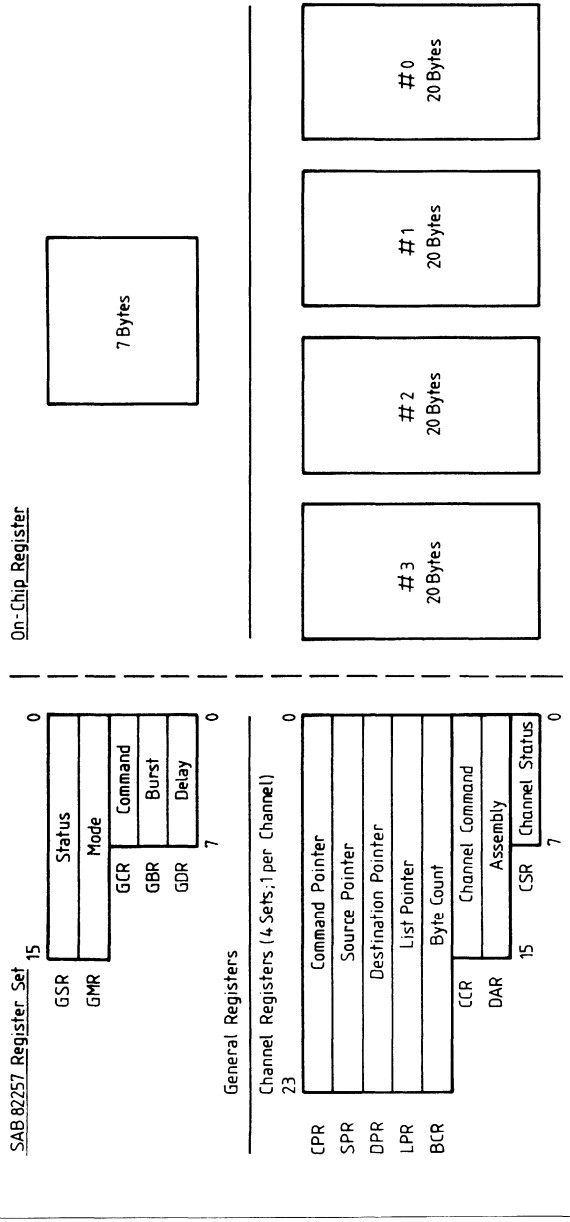
In **local mode** this arbitration is performed by the CPU (HOLD/HLDA sequence).

In **remote mode** this arbitration is done by the SAB 82257 itself ($\overline{CS}$/BREL sequence).

The slave interface can be synchronous or asynchronous to the processor.

In the **asynchronous** case ($\overline{RD}$, $\overline{WR}$ inputs) the internal register address (8-bit) is taken from the low-order address pins (A7 to A0).

Figure 45
Register Set



In case of **synchronous** slave interface (status signals inputs, not possible in remote mode), the internal register address (8-bit) is taken

- in 286 mode from the low-order address pins (A7 to A0), and
- in 186/8086 mode from the low-order address/data pins (AD7 to AD0).

The following table gives a summary:

Modes of Operation	Slave Interface	
	Asynchronous to the Processor. Address information is expected at line	**Synchronous** to the Processor. Address information is expected at line
Remote Mode	A7 to A0	–
286 Mode	A7 to A0	A7 to A0
186/8086 Mode	A7 to A0	AD7 to AD0

The next table gives a list of the accessible, user-visible registers and their internal addresses. All the addresses are even (word) addresses, but the registers can also be accessed bytewise.

Note that the 24-bit registers (CPR, SPR, DPR, LPR, BCR and CCR) have **two** addresses, one for the low word and one for the high byte. The allocation of addresses is done on a functional and on a channel number basis, e.g. bits 6 and 7 determine the number of the channel.

Note

- Not defined locations are unused and reserved. They should, however, not be addressed to prevent occurring of undefined effects.
- For accesses to SAB 82257 registers by the SAB 80286 CPU in "protected virtual address mode", refer to the note in section 3.2.1.5.

Register Address Arrangement

Register	Size	Register Address (Bits 7 to 0)				
		Address Bits 7, 6				Address Bits 5 to 0
		0 0 (n = 0)	0 1 (n = 1)	1 0 (n = 2)	1 1 (n = 3)	
General Register						
GSR	16	GSR	–	–	–	000100
GMR	16	GMR	–	–	–	001000
GCR	8	GCR	–	–	–	000000
GBR	8	GBR	–	–	–	001010
GDR	8	GDR	–	–	–	001100
Channel Register						
CPR_n	24	CPR_0L	CPR_1L	CPR_2L	CPR_3L	100000
		CPR_0H	CPR_1H	CPR_2H	CPR_3H	100010
SPR_n	24	SPR_0L	SPR_1L	SPR_2L	SPR_3L	100100
		SPR_0H	SPR_1H	SPR_2H	SPR_3H	100110
DPR_n	24	DPR_0L	DPR_1L	DPR_2L	DPR_3L	101000
		DPR_0H	DPR_1H	DPR_2H	DPR_3H	101010
LPR_n	24	LPR_0L	LPR_1L	LPR_2L	LPR_3L	110000
		LPR_0H	LPR_1H	LPR_2H	LPR_3H	110010
BCR_n	24	BCR_0L	BCR_1L	BCR_2L	BCR_3L	111010
		BCR_0H	BCR_1H	BCR_2H	BCR_3H	111010
CCR_n	16	CCR_0	CCR_1	CCR_2	CCR_3	111100
DAR_n	16	DAR_0	DAR_1	DAR_2	DAR_3	010010
CSR_n	8	CSR_0	CSR_1	CSR_2	CSR_3	010000

General Registers

GSR	= General Status Register
GMR	= General Mode Register
GCR	= General Command Register
GBR	= General Burst Register
GDR	= General Delay Register

Channel Registers

CPR	= Command Pointer Register
SPR	= Source Pointer Register
DPR	= Destination Pointer Register
LPR	= List Pointer Register
BCR	= Byte Count Register
DAR	= Data Assembly Register
CSR	= Channel Status Register
L	= Low Word
H	= High Byte
n	= Channel Number

Note
The register locations which are not specified are used for several internal working registers. Therefore these locations should never be accessed and - what is even more important- should never be written into.

6.2 General Control

6.2.1 Mode Selection

The SAB 82257 has been defined to work with all Siemens 16-bit family processors, i. e. SAB 80286, SAB 80186/188 and SAB 8086/88, without additional support and interface logic. As the local buses of the above processors are different concerning signals, functions and timings, the SAB 82257 has an adaptive bus interface to meet the different requirements of these local buses.

As a result of this, a bus compatibility with identical timing is ensured for processors SAB 80286, SAB 80186/188 and SAB 8086/88. The compatibility with the 8-bit bus version of the processors SAB 8088 and SAB 80188 is guaranteed by defining the physical bus width of the SAB 82257 (per software) to 8 bits.

Note
In addition, the SAB 82257 can work in remote mode or standalone mode where it is not coupled directly to a processor. In remote mode all bus timings are compatible with 286 mode.

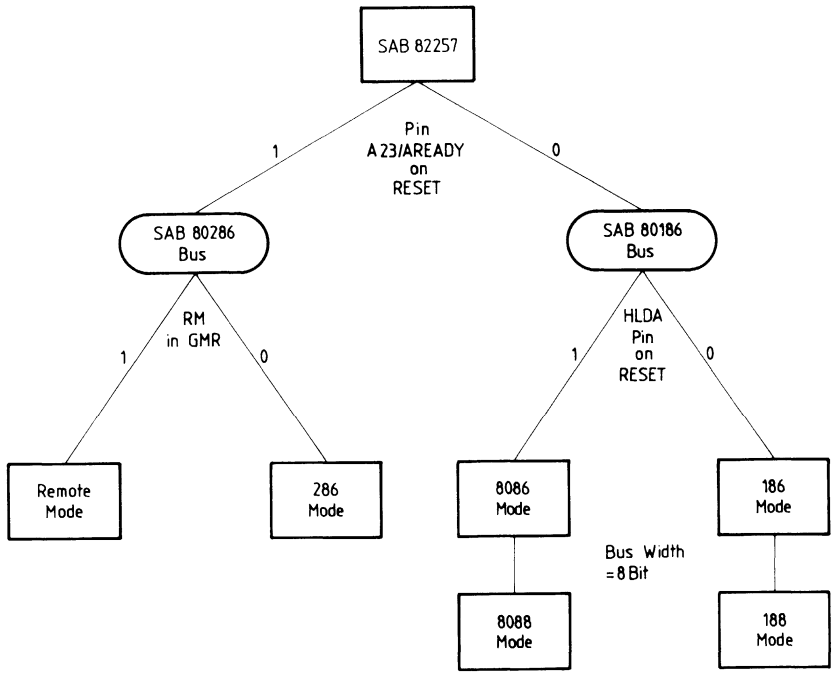
Bus Timing Mode Selection
For the bus timing mode selection the SAB 82257 uses two pins:

- A23/AREADY and
- HLDA

On the trailing edge of the RESET signal the logic levels at these pins determine the type of bus timing for SAB 82257 operation.

Pin A21 must be high during reset for proper operation! (For details see section "Bus Operation")

Figure 46
Mode Selection (Survey)



6.2.2 General Commands

To control the channel execution the following general commands are used:

1. **START channel(s)** with control space in system/memory space
   This general command defines the location of the control space (here system/memory space) and initiates the setup routine (see section 6.4.3). The execution of the command is prioritized.

2. **START channel(s)** with control space in resident / I/O space

   This general command defines the location of the control space (here resident / I/O space) and initiates the setup routine (see section 6.4.3). The execution of the command is prioritized.

3. **CONTINUE channel(s) operation**

   The CONTINUE command works directly with the internally stored register parameters and it continues a previously stopped channel operation. The execution of the command is prioritized.

4. **STOP channel(s)**

   The STOP command forces the channel(s) into the status "stopped", indicated by the channel's DMST bits in general status register (GSR), without any additional routine. The command is executed immediately.

5. **HALT/single-step channel(s)**

   The HALT command is a multiple function command:

   - It forces the channel into the single-step and halt mode, indicated by the SSH bit in the channel status register (CSR).
   - If the channel is running, it will be halted after completion of the current command block execution (either a type 1 or a type 2). The halted state is shown by the H-bit of the CSR. Note, that the DMST bits in general status register (GSR) are not changed.
   - If the channel is halted (or stopped), the HALT/single-step command starts the channel, and the channel will again be halted after completion of the next command block execution (type 1 or type 2).

   The single-step and halt mode is finished by a START or CONTINUE command.

6.2.3 General Control Registers

The SAB 82257 contains the following general registers:

- General Mode Register (GMR)
- General Command Register (GCR)
- General Burst Register (GBR)
- General Delay Register (GDR)
- General Status Register (GSR)

These general registers are used by the CPU for all the channels.

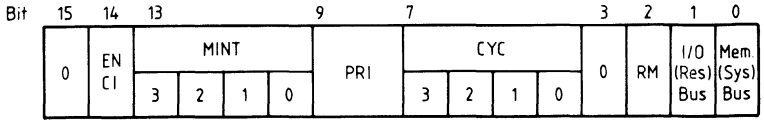
General Mode Register (GMR), 16-Bit Register

In the general mode register (GMR), the system wide parameters are specified:

- physical bus widths of system (memory) bus and resident (I/O) bus
  Note:
  System/resident corresponds with remote mode, memory / I/O corresponds with local mode.
- remote or local (normal) mode;
- which channels operate in single or two-cycle mode;
- the relative priority of channels;
- enable/disable of channel interrupts;
- the function of $\overline{EOD2}$ pin.

This register should be the first to be programmed by the CPU after reset. If it is loaded bytewise, the low byte should be programmed first.

Figure 47
General Mode Register Fields



- **MEMBUS/SYSBUS** (bit 0)
  Physical bus widths of memory (system) bus will be selected by bit 0:

MEMBUS SYSBUS	Local Mode	Remote Mode
0	The physical data bus width of memory bus is 8 bit.	The physical data bus width of system bus is 8 bit.
1	The physical data bus width of memory bus is 16 bit.	The physical data bus width of system bus is 16 bit.

- **IOBUS/RESBUS** (bit 1)
  Physical bus widths of I/O (resident) bus will be selected by bit 1:

IOBUS RESBUS	Local Mode	Remote Mode
0	The physical data bus width of I/O bus is 8 bit.	The physical data bus width of resident bus is 8 bit.
1	The physical data bus width of I/O bus is 16 bit.	The physical data bus width of resident bus is 16 bit.

- **RM Mode Select** (bit 2)
  The SAB 82257 has two basic modes of operation selected by bit 2:
  RM = 0: Local mode, SAB 82257 is locally coupled with CPU.
  RM = 1: Remote mode, SAB 82257 is autonomous and coupled with CPU via the system bus.
- **CYCn; n = 0, 1, 2, 3** (bits 4, 5, 6, 7)
  One bit for each channel
  Bits 4 to 7 define which channels operate in single or two-cycle mode:
  CYCn = 0: Two-cycle DMA transfer mode of channel n.
  CYCn = 1: Single-cycle transfer mode of channel n.
- **PRI** (bits 8, 9)
  Bits 8, 9 are used to describe the priority of the channels:

PRI		Channel Priority
Bit 9	Bit 8	
0	0	All channels have fixed priority (channel 0: highest priority, channel 3: lowest priority).
1	1	All channels have rotating priority.

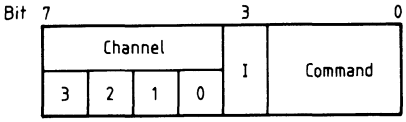
- **MINTn; n = 0, 1, 2, 3** (bits 10, 11, 12, 13)
  One bit for each channel
  Bits 10 to 13 are used to mask the interrupts from channels:
  MINTn = 0: Enable channel n interrupt.
  MINTn = 1: Disable (mask) channel n interrupt.

  Note
  For dynamic change of MINT bits, only the upper byte of the general mode register GMR can be addressed.
- **ENCI** (bit 14)
  Bit 14 defines, if channel interrupts (as masked with MINT bits) should be issued as "end of DMA" ($\overline{EOD}$) signals of the corresponding $\overline{EOD}$ line or as a common INTOUT signal on the $\overline{EOD2}$ line:
  ENCI = 0: $\overline{EOD2}$ pin = $\overline{EOD2}$ (common interrupt not enabled)
  ENCI = 1: $\overline{EOD2}$ pin = INTOUT (common interrupt enabled)

General Command Register (GCR), 8-Bit Register

The general command register (GCR) is essentially used to start, stop and continue the operations of any of the four channels of the SAB 82257 by commands. The START command additonally defines control space assignment. The pending interrupt from any channel is also cleared through the GCR. It is possible to address any combination of channels simultaneously. A HALT/single-step command enables execution of commands in a single-step mode, e.g. for debugging. The general command register (GCR) is directly loaded by the CPU.

Figure 48
General Command Register Fields



- **COMMAND** (bits 0, 1, 2)
  These three bits select the general commands for the channels of the SAB 82257:

COMMAND Bit 2	Bit 1	Bit 0	Channel Command
0	0	0	NOP (no operation).
0	0	1	CONTINUE Channel(s) Operation after it has been stopped by the STOP command (with existing register parameters)
0	1	0	START Channel(s) – Command block at system/memory space.
0	1	1	START Channel(s) – Command block at resident/I/O space.
1	0	0	STOP Channel(s).
1	0	1	Not Valid
1	1	0	Not Valid
1	1	1	HALT/Single-Step Channel(s): Start execution and stop after next command block has been loaded.

Note
- A STOP command immediately stops the activity of the addressed channel(s).
- The HALT command stops activity after termination of the running channel command. An additional HALT command leads to the execution of the next channel command (type 1 or type 2) of the channel program (single step).

- **I INTERRUPT** (bit 3)
  I = 0: NOP (no operation)
  I = 1: CLEAR pending interrupt(s) of channel(s).

- **Channel n; n = 0, 1, 2, 3** (bits 4, 5, 6, 7)
  This field determines to which channel(s) the given command refers:

Bit 7	Bit 6	Bit 5	Bit 4	Explanation
0	0	0	1	General Command for Channel 0
0	0	1	0	General Command for Channel 1
0	1	0	0	General Command for Channel 2
1	0	0	0	General Command for Channel 3

Note
- A START command to a channel which is running is regarded as NOP (no operation).
- It is possible to start/stop/continue more than one channel simultaneously, or to clear interrupts of more than one channel at the same time.

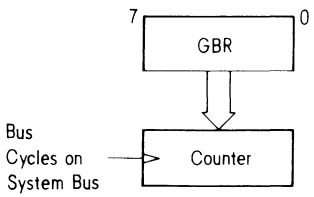
General Burst Register (GBR), 8-Bit Register

The value in the general burst register (GBR) determines the maximum number of contiguous bus cycles that can be requested by the SAB 82257. It must be loaded directly by the CPU (not possible via command blocks).

Attention: If the GBR is programmed to zero, contiguous bus cycles are not limited (no burst count).

The contents of the general burst register is loaded into a general burst counter (GBC) for the actual counting.

Figure 49



For the operation of the burst counter the following rules apply:

Local Mode	Remote Mode
– Whenever the SAB 82257 controls a bus cycle the burst counter is decremented by one, but not beyond zero.	– The General Burst Counter is decremented only during system bus accesses. *Note* As resident bus accesses in remote mode do not burden the system bus, "bursts" are only relevant for system bus cycles.
– If the burst counter reaches zero, the SAB 82257 releases the bus.	– If the burst counter reaches zero, the SAB 82257 releases the bus.

Note
The execution of unseparable (locked) bus cycles prevents the release of the bus even if the burst counter has reached zero!
Refer to the note for general delay register (GDR).

General Delay Register (GDR), 8-Bit Register

The contents of the general delay register (GDR) determines the minimum number of clock cycles between burst accesses. It must be loaded directly by the CPU (not possible via command blocks). If the GDR is programmed to zero, there is no minimum delay between HOLD requests.

The contents of the general delay register is loaded into a general delay counter (GDC) for the actual counting.

Figure 50

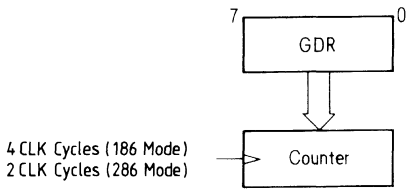
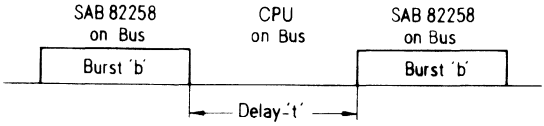


Figure 51
Bus Loading

a) **Bus Loading**

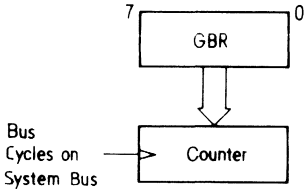


$$\text{Bus Load Due to SAB 82258} = \frac{b}{b+t}$$

b) **General Burst Register (GBR)** - to Program 'b'

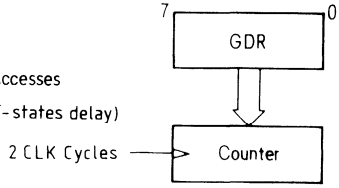
Determines Max. Number of Contiguous Bus Cycles from SAB 82258

If GBR=0, No Limit



c) **General Delay Register (GDR)** - to Program 't'

Determines Min. Number of Clock Cycles Between Burst Accesses (default after reset=0, i.e 4 T-states delay)



For the operation of the delay counter the following rules apply:

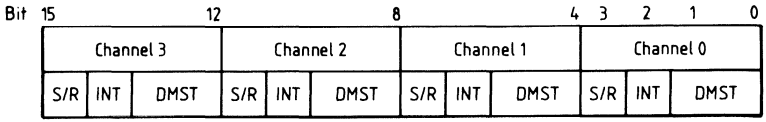
Local Mode	Remote Mode
– The general delay counter is decremented by one, if the SAB 82257 does not perform bus accesses. – In **286 mode,** it is decremented every second T-state and in **186 mode** every fourth T-state	– The general delay counter is decremented if the SAB 82257 does not perform system bus accesses. – It is also decremented during resident bus accesses. *Note* As in remote mode resident bus accesses do not burden the system bus, "delay" is only relevant for system bus cycles.
– Whenever the delay counter is counted to zero, both the burst counter **and** the delay counter are reloaded from their registers.	– Whenever the delay counter reaches zero, both the burst counter **and** the delay counter are reloaded from their registers.

Note
- Refer to the note for the general burst register (GBR).
- By programming the burst and delay register it is possible to restrict the bus load generated by the SAB 82257 on the CPU bus. The bus load is defined by the formula given in figure 51. The factor b (burst) is programmed in the GBR, t (delay time) in the GDR (see figure 51).
  Since the SAB 82257 can also execute locked bus cycles, the maximum burst length consists of b + 3 (8-bit bus) or b + 2 (16-bit bus) bus cycles.

General Status Register (GSR), 16-Bit Register

The general status register (GSR) provides the current status information for all four channels. It shows whether the channels are running or stopped, which channels have interrupts and where the control space for each channel lies.

Figure 52
General Status Register Fields



- **DMST** DMA status
  (bits 0, 1; bits 4, 5; bits 8, 9; bits 12, 13)
  – two bits per channel
  These bits show the current status of the appropriate channel:

DMST Bits 1, 5, 9, 13	DMST Bits 0, 4, 8, 12	Indicated Status **Four** Channels (DMST0... DMST3)
0	0	Channel inactive (stopped), no DMA request pending.
0	1	Channel inactive (stopped), DMA request pending.
1	0	Channel in organizational processing (setup, chaining, termination).
1	1	DMA transfer in progress.

Note
The status "DMA transfer in progress" (i.e. DMST = 11) is indicated from beginning of block transfer (after setup) until termination of transfer.

- **INT**
  (bit 2; bit 6; bit 10; bit 14)
  – one bit for each channel
  This bit indicates a pending interrupt from the appropriate channel.
  INT = 0: No interrupt from this channel.
  INT = 1: Interrupt pending from this channel (indicated on channel specific $\overline{\text{EOD}}$ pin, or on $\overline{\text{EOD2}}$ pin as INTOUT signal, if common interrupt is enabled in general mode register GMR).

- **S/R** SYSBUS/RESBUS (in remote mode)
  MEMBUS/IOBUS (in local mode)
  (bit 3; bit 7; bit 11; bit 15)
  – one bit per channel
  This bit indicates where the control space for the appropriate channel lies:

S/R	Local Mode	Remote Mode
0	Control space is on I/O bus.	Control space is on resident bus.
1	Control space is on memory bus.	Control space is on system bus.

6.3 Channel Control

All channels of the SAB 82257 can be operated with synchronized and nonsynchronized transfers. Features like command chaining and data chaining are supported.

Data transfer, with all its modifications, is controlled with the aid of channel command blocks. These contain the channel command word and all the initial parameters for data transfer execution. The channel commands, that the SAB 82257 is to execute-are edited by the CPU and stored in common memory (system memory). The CPU loads the start address of the channel command block into the command pointer register of the respective channel in the SAB 82257, in the same manner as if the CPU were updating a memory cell. In the same way the SAB 82257 receives a channel start command (loading of the GCR). Initiation of a channel operation in the SAB 82257 is thus very simple, as the CPU need not perform anything else. The start command for a channel causes the SAB 82257 to read the channel command block with all its parameters from memory, and to load them into the internal channel registers.

The channel registers that can be loaded via the command blocks are:

- Source Pointer Register (SPR)
- Destination Pointer Register (DPR)
- List Pointer Register (LPR)
- Byte Count Register (BCR)
- Channel Command Register (CCR)

After examining the channel command for programming errors, the data block transfer is executed if no errors are detected. After termination of the transfer, the reason for termination is indicated within a word in the channel command block (channel status).

While the channel is active (not stopped), the CPU should not access the channel's control space to prevent errors during operation.

6.3.1 Channel Commands

Survey

There are two basic types of channel commands:

– type 1 channel commands for data transfers and
– type 2 channel commands for command chaining control.

The **type 1 channel commands** specify the actual data transfer operation, as for example the data block transfer from a peripheral to memory. A type 1 channel command and its parameters such as source pointer, destination pointer, byte count, constitute a type 1 channel command block (CCB). Type 1 commands are also called transfer channel commands.

The **type 2 channel commands** specify the control operations of the channel, as for example conditional jump, programmable interrupt or channel stop operations. A type 2 channel command and its parameters such as signed 16-bit displacement or an absolute address for a JUMP operation constitute a type 2 channel command block (CCB). Type 2 commands are also called organizational channel commands. Comparatively complex operations are possible using the following chaining features of the SAB 82257:

- Data Chaining
  Data Chaining is performed through the linking of different source blocks to form one destination block, or through the scattering of one source block into different destination blocks.

- Command Chaining
  Command chaining is performed through the linking of several channel command blocks. Normally command blocks are executed sequentially. Branching is possible using type 2 commands.

- Channel Programs
  Command chaining is performed automatically after completion of a channel command (except a stop command). This means that a series of channel commands will execute sequentially without CPU activity. Such a series of channel commands is called a channel program. The type 2 channel commands allow the construction of complex powerful channel programs apt for an intelligent response to a request.

A complete channel program consists of at least two channel command blocks (CCBs), one with a type 1 command and one with a type 2 command (see figure 56). All channel programs are stored in memory and prepared by the CPU.

Type 1 Channel Commands and Command Blocks

As shown in figure 54, a type 1 channel command block consists of the following elements in order to accomplish a data transfer:

- a 16-bit type 1 channel command,
- a 24-bit source pointer,
- a 24-bit destination pointer,
- a 24-bit byte count (block length) and
- a 16-bit status word updated by the SAB 82257 after DMA operation.

All pointers represent 24-bit real physical addresses; not selector:offset!

Figure 53
Simplest Channel Program

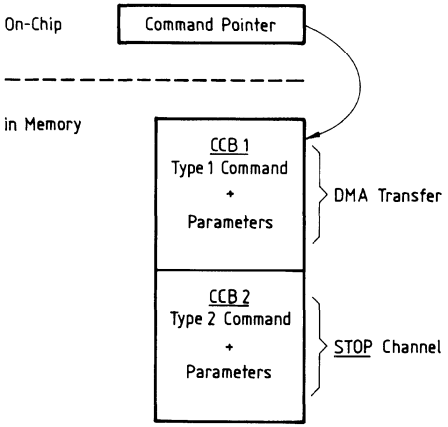
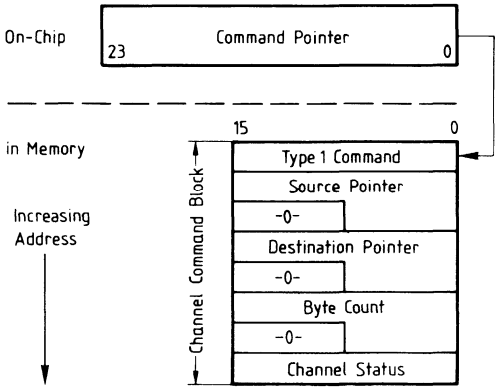


Figure 54
Type 1 Channel Command Block



The type 1 channel command defines the task to be performed by the channel:

- For both source and destination:
  - the logical bus width,
  - memory space/I/O space in local mode or global space/local space in remote mode,
  - how the pointers should be changed during transfer.
- If data chaining is to be performed, if so then what type.
- If $\overline{EOD}$ signal is to be generated at the end of transfer.
- If $\overline{EOD}$ pin is to be used as an external terminate input.
- Is the DMA transfer free-running or synchronized, if so then what type.

This standard command block is 16 bytes long and is needed for all DMA transfers.
The type 1 channel command is detailed in section 6.3.2 under Channel Command Register.

Note
Single-cycle transfers only use the source pointer field of the command block.

Type 2 Channel Commands and Command Blocks

As shown in figure 56, a type 2 channel command block contains

- a 16-bit type 2 channel command and
- a 32-bit parameter field.

The type 2 channel commands provide means to either branch to another command block or stop the respective channel.

The type 2 jump command (branching) either requires a 16-bit displacement (relative jump) referring to the first byte of the currently executed command block or a 24-bit pointer (absolute jump) to the desired next command block.

The type 2 stop command immediately stops the activity of a channel. The parameter field is not used and normally contains zero.

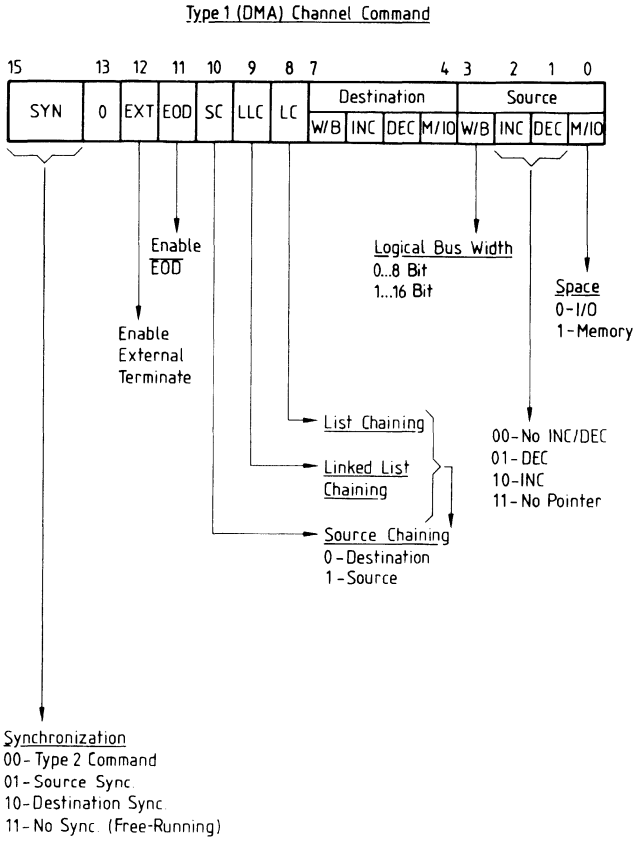
Conditional Execution

As shown in figure 56, the type 2 channel command contains a 2-bit condition code providing 2 termination conditions:

- Bit 0: BC byte count exceeded
- Bit 1: ET external termination of block transfer

Setting the respective bit to one enables execution of the type 2 command if the indicated condition occurs. Several conditions are ORed.
A condition is recognized as true if the associated termination status bit (within the CSR) is active. The flag I (see type 2 command) allows to invert the status bits (CSR) before they are evaluated.

Figure 55
Type 1 Command Fields



Unconditional Execution

If both condition bits are set to one, the command is executed anyway. If both condition bits are set to zero, the command is never executed, which means a NOP command.

With the new command pointer the next channel command is fetched (command chaining). If the indicated condition code for conditional commands is not fulfilled, the command pointer is incremented by 6 bytes and the next channel command (type 1 or type 2) is fetched. Thus a NOP is executed.

The type 2 channel command is detailed in section 6.3.2 under Type 2 Channel Command Register Fields.

The type 2 commands also allow activation of a program-controlled interrupt (INTOUT or/ and $\overline{\text{EOD}}$ signal) during execution of the command.
This is controlled by two flags in the command, the IT flag (bit 10) and the ED flag (bit 11). In contrast to the type 1 command $\overline{\text{EOD}}$ (synchronous with last data transfer), the type 2 command $\overline{\text{EOD}}$ is an asynchronous $\overline{\text{EOD}}$. If the ED or IT flag is set, signal generation is always unconditional, independent of the condition code.

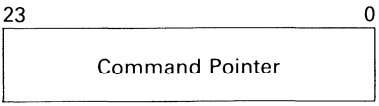
6.3.2 Channel Registers

Each of the four SAB 82257 channels has the following channel registers:

- Command Pointer Register (CPR)
- Source Pointer Register (SPR)
- Destination Pointer Register (DPR)
- List Pointer Register (LPR)
- Byte Count Register (BCR)
- Channel Command Register (CCR)
- Channel Status Register (CSR)
- Data Assembly Register (DAR)

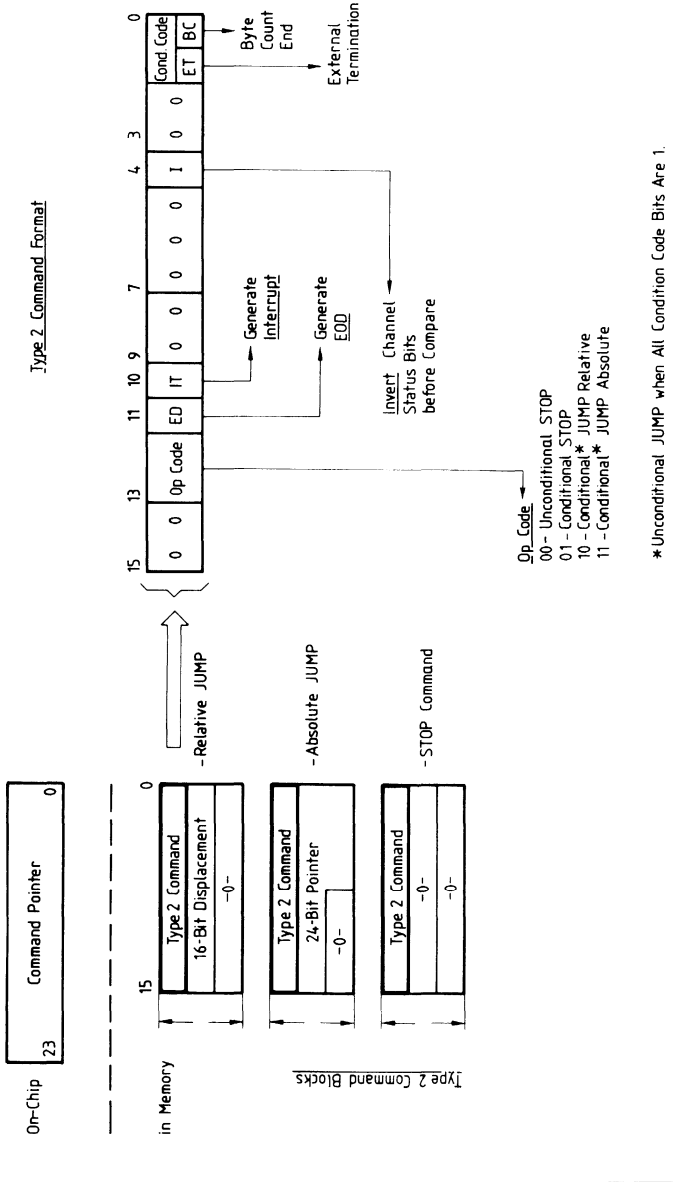
All these channel registers, except the command pointer register (CPR) and the channel status register (CSR), are loaded by the SAB 82257 from the channel command blocks located in memory.

Command Pointer Register (CPR), 24-Bit Register

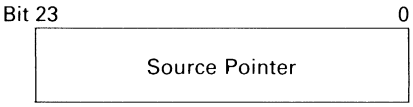


The command pointer register (CPR) contains the physical address of the command block in memory. It must be loaded by the CPU before starting the channel.

Figure 56
Structure of Type 2 Channel Commands



Source Pointer Register (SPR), 24-Bit Register

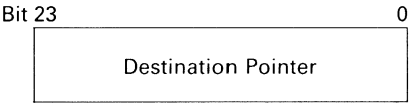


The source pointer register (SPR) contains the physical address of the source

- in local mode: memory space or I/O space
- in remote mode: system space or resident space

in a DMA transfer. In single-cycle transfer mode (i.e. CYCn = 1 in the general mode register), the source pointer register (SPR) contains the sole address pointer (which can also be the address of destination).

Destination Pointer Register (DPR), 24-Bit Register

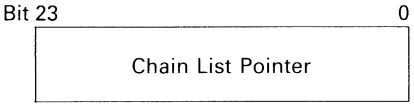


The destination pointer register (DPR) contains the physical address of the destination

- in local mode: memory space or I/O space
- in remote mode: system space or resident space

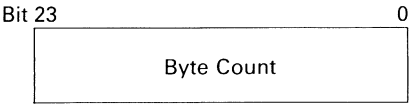
in a DMA transfer. In single-cycle transfer mode the DPR is not used.

List Pointer Register (LPR), 24-Bit Register



The list pointer register (LPR) is used for data chaining operation. It points to the actual list element.

Byte Count Register (BCR), 24-Bit Register



The byte count register (BCR) contains the byte count for the DMA transfer.

Channel Command Register (CCR), 16-Bit Register

The channel command register (CCR) specifies

- the type of DMA transfer when it is loaded from the type 1 channel command field of a type 1 channel command block or
- the type of internal operation when it is loaded from the type 2 channel command field of a type 2 channel command block.

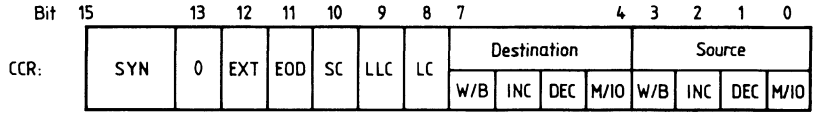
Channel Command Register (CCR) for Type 1 Channel Commands

In the channel command register for type 1 channel commands, the type of DMA transfer is specified:

- For both source and destination:
  - the logical bus width,
  - memory space / I/O space in local mode or global space/local space in remote mode,
  - how the pointers should be changed during transfer.
- If data chaining is to be performed, and of which type.
- If $\overline{\text{EOD}}$ signal is to be generated at the end of transfer.
- If $\overline{\text{EOD}}$ pin is to be used as an external terminate input.
- Is the DMA transfer free-running or synchronized, if so then what type.

Type 1 Channel Command Register Fields

Figure 57



The explanation for M/IO bit, DEC bit, INC bit and W/B bit is valid for source (bit 0 to bit 3) and destination (bit 4 to bit 7).

- **M/IO** (bit 0, bit 4)
  Distinction between memory/system and I/O / resident space addresses:

M/IO	Local Mode	Remote Mode
0	The source/destination pointer resides in the I/O space.	The pointer resides in the local space (resident bus).
1	The source/destination pointer resides in the memory space.	The pointer resides in the global space (system bus).

- **DEC** (bit 1, bit 5)
  INC (bit 2, bit 6)
  These bits define how the pointers should be changed during DMA transfer:

INC Bit 2 Bit 6	DEC Bit 1 Bit 5	Explanation
0	0	Do not increment or decrement source/destination pointer after each transfer.
0	1	Decrement source/destination pointer by one or two (depends on W/B bit) after each transfer.
1	0	Increment source/destination pointer by one or two (depends on W/B bit) after each transfer.
1	1	No source/destination pointer needed at all.

Note
In the case of no destination the SAB 82257 itself (assembly register) is the destination.
In the case of no source pointer the source is a byte or word constant and located in the command block within the source pointer field.

- **W/B Logical Bus Width** (bit 3, bit 7)
  Logical bus width of source/destination will be selected by bit 3 and bit 7:
  W/B = 0: Source/destination data transferred in bytes
  W/B = 1: Source/destination data transferred in words

  Note
  If the physical bus width is 8 bit, W/B = 1 is ignored.

- **LC List Chaining** (bit 8)
  Channel command register bit 8 enables list chaining:
  LC = 0: No list chaining
  LC = 1: List chaining enabled

  Note
  The source pointer (if source chaining is indicated by SC bit = 1 in the CCR) or the destination pointer (if destination chaining is indicated by SC bit = 0 in the CCR) points at the chain list.

- **LLC Linked List Chaining** (bit 9)
  Channel command register bit 9 enables linked list chaining:
  LLC = 0: No linked list chaining
  LLC = 1: Linked list chaining enabled

Note
Only one of the bits LC and LLC must be set, i.e.

LLC Bit 9	LC Bit 8	Explanation
0	0	No Chaining
0	1	List Chaining
1	0	Linked List Chaining
1	1	Not Allowed

- **SC** (bit 10)
  Bit 10 indicates whether destination or source data chaining is to be performed.

  SC = 0: Destination data chaining if LC or LLC is set.
  SC = 1: Source data chaining if LC or LLC is set.

  A summary of the description of LC bit, LLC bit and SC bit is shown in the following table:

SC Bit 10	LLC Bit 9	LC Bit 8	Explanation
0	0	0	No Chaining
0	0	1	List Chaining of the Destination Data
0	1	0	Linked List Chaining of the Destination Data
0	1	1	Not Allowed
1	0	0	No Chaining
1	0	1	List Chaining of the Source Data
1	1	0	Linked List Chaining of the Source Data
1	1	1	Not Allowed

- **EOD End of DMA Transfer** (bit 11)
  Channel command register bit 11 indicates if the $\overline{\text{EOD}}$ signal is to be generated at the end of a DMA transfer:
  EOD = 0: End of DMA signal from SAB 82257 to peripheral device (or CPU) disabled
  EOD = 1: End of DMA signal from SAB 82257 to peripheral device (or CPU) enabled
  Note
  The channel generates a 2-clock $\overline{\text{EOD}}$ strobe signal at byte count termination of block transfer. Transmission is synchronous with last data transfer to synchronizing device or – in case of no synchronization – with last destination cycle. In case of data chaining $\overline{\text{EOD}}$ is generated by each byte-count-exceeded condition. $\overline{\text{EOD}}$ signals initiated by type 1 commands are called synchronous $\overline{\text{EOD}}$ signals.

- **EXT External Termination** (bit 12)
  Bit 12 indicates if the $\overline{EOD}$ pin is used as an external terminate input:
  EXT = 0: External termination not enabled
  EXT = 1: External termination enabled
  Note
  The peripheral device uses the channel's $\overline{EOD}$ line (strobe pulse) for external termination signal if bit 12 is set.

- **SYN** (bits 14, 15)

SYN Bit 15	SYN Bit 14	Explanation
0	0	Type 2 Channel Command
0	1	Source Synchronization of Data Transfer
1	0	Destination Synchronization of Data Transfer
1	1	Free-Running of DMA Transfer, i.e. No Synchronization

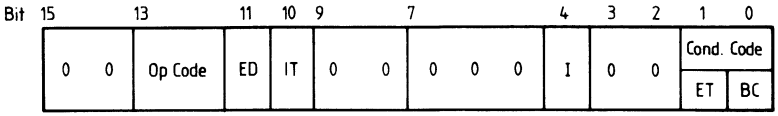
Channel Command Register (CCR) for Type 2 Channel Commands

In the channel command register for type 2 channel commands the type of internal operation such as
- conditional jump operation
- channel stop operation
- programmable interrupt

is specified. The register is loaded from the type 2 command field of a command block.

Type 2 Channel Command Register Fields

Figure 58



- **Condition Code CC** (bits 0, 1)
  These two bits define the DMA termination condition code for a conditional operation.

  BC (bit 0)

  BC = 0: No function
  BC = 1: Byte count exceeded

  ET (bit 1)

  ET = 0: No function
  ET = 1: External termination (not programmed $\overline{EOD}$)

  Note
  - The bits BC and ET correspond to the bits in the channel status register (CSR).
  - If the CC field equals 0 0 (and I bit = 0) for a jump command, it is a NOP.
  - If more than one of these bits are set, then the condition is an OR operation of those bits.
  - If the CC field equals 1 1 (and I bit = 0), the command becomes an unconditional command.
  - An explicit unconditional stop command ignores the CC field.

- **I Invert Channel Status Bits before Compare** (bit 4)

  I = 0: No invert
  I = 1: Status register bits are first complemented and then compared with the CC bits.

- **IT Interrupt** (bit 10)
  Bit 10 indicates whether an INTOUT signal on the $\overline{EOD2}$ pin or a channel specific $\overline{EOD}$ signal should be activated with the execution of the command:

  IT = 0: No interrupt
  IT = 1: Send interrupt; this is
  - a static signal (INTOUT) on $\overline{EOD2}$ pin if
    1. not masked with MINT bit in GMR and
    2. ENCI bit in GMR is equal 1, else
       - a strobe pulse on channel-specific $\overline{EOD}$ pin.

  Note
  - In all cases the interrupt is indicated in the general status register (GSR).
  - In all cases the interrupt has to be cleared by the CPU with a general command.

- **ED End of DMA** (bit 11)
  Bit 11 indicates if a channel-specific $\overline{EOD}$ signal should be activated with the execution of the command:

  ED = 0: No $\overline{EOD}$
  ED = 1: Send $\overline{EOD}$ of channel n (strobe signal)

  Note
  This is an asynchronous $\overline{EOD}$ signal in contrast to the $\overline{EOD}$ signal generated during the last data transfer of a type 1 transfer command.

- **Operation Code** (bits 12, 13)
  These two bits select the command for the channel operations:

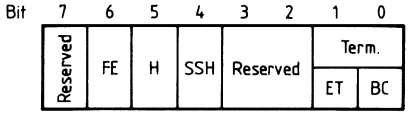
Op Code Bit 13	Op Code Bit 12	Command
0	0	Unconditional Stop Channel Operation.
0	1	Conditional Stop Channel Operation.
1	0	Conditional Jump Relative.
1	1	Conditional Jump Direct.

Note
- A NOP command is realized with the condition code all zero for a conditional command.
- An unconditional command is realized with the condition code all ones.
- Refer to note for bits BC and ET.

Channel Status Register (CSR), 8-Bit Register

The channel status register (CSR) reflects the status of the appropriate channel. The least significant bits (0 and 1) contain the termination condition (can be used for conditional execution of type 2 commands), and the most significant half byte (bit 4 to bit 7) is for error, busy status and halted state. After execution of a type 1 channel command block the CSR is written back into the command block for CPU access. The channel status register (CSR) can also be read directly by the CPU.

Figure 59
Channel Status Register Fields



- **DMA Termination** (bits 0, 1)
  These two bits of the channel status register (CSR) reflect the way in which the DMA transfer was terminated:

  BC Byte Count (bit 0)
  BC = 0: No function
  BC = 1: Byte count exceeded

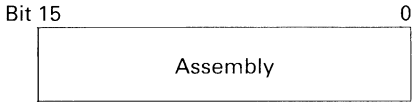
  ET External Termination (bit 1)
  ET = 0: No function
  ET = 1: External termination (not programmed $\overline{EOD}$)

- **SSH Single-Step Halt Mode** (bit 4)
  The single-step halt mode bit indicates the operations of the channel in single-step mode, where after the execution of each CCB the channel comes to a halt:
  SSH = 0: No function
  SSH = 1: Channel operating in SSH mode

- **H Halted** (bit 5)
  Bit 5 indicates whether the channel is in halted state:
  H = 0: No function
  H = 1: Channel in halted state

- **FE Fatal Error** (bit 6)
  FE bit indicates the occurrence of a fatal error during the execution of the current CCB:
  FE = 0: No function
  FE = 1: Fatal error has occurred

Data Assembly Register (DAR), 16-Bit Register



The data assembly register (DAR) is used as a buffer for data assembly and disassembly operations during data transfer with unequal bus widths.

6.3.3 Command Chaining

The term "command chaining" describes the automatic linking of several channel command blocks to a command chain called a channel program (see section 6).

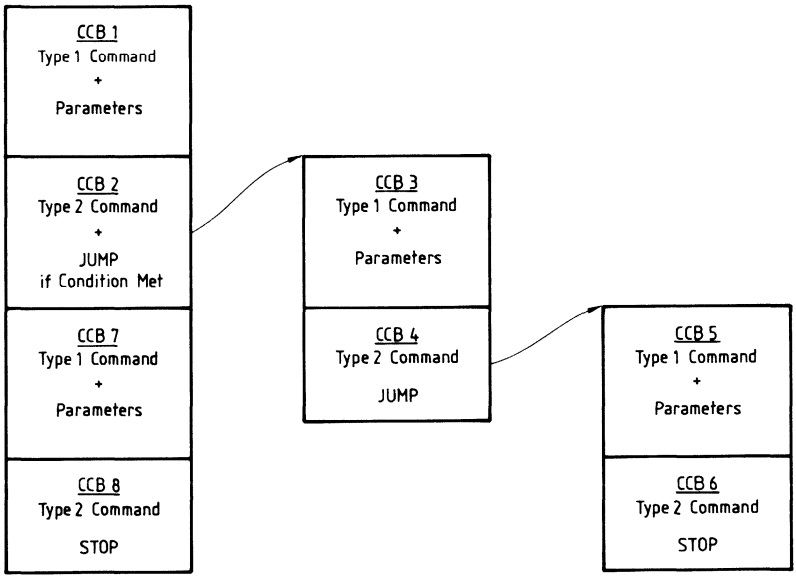
Normally command blocks are executed **sequentially** like a program. After completion of one channel command block, the SAB 82257 starts – without CPU intervention – processing the next block automatically by incrementing the command pointer by the length of the command block. It is thus possible to execute a sequence of different types of DMAs autonomously. Chaining will be performed until a STOP command is detected (type 2 command) or the CPU stops the channel directly.
Using type 2 (jump) commands, **conditional branching** is possible. This allows continuation of channel programs at arbitrary locations.

The shortest and simplest DMA implementation therefore consists of a type 1 command block followed by a "STOP" (type 2) command block.

A simple auto-reload DMA can be implemented with a DMA command block followed by an unconditional JUMP to the top of the DMA command block.

More complex structures are also easily implemented (see figures 60 and 61).

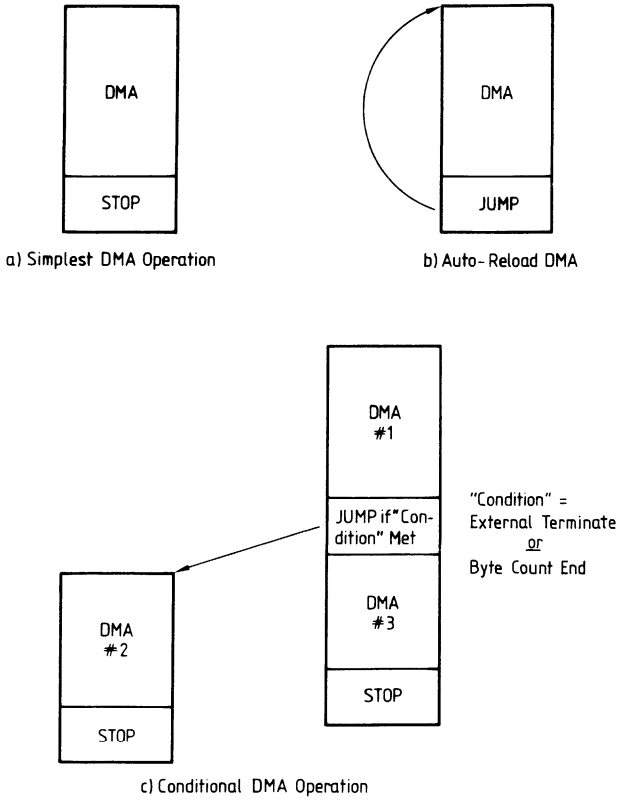
Figure 60
Command Chaining and Branching



The initiation of the command chaining is done by the termination processing belonging to the last data transfer execution (except of asynchronous external termination all termination processing belongs to the last data transfer execution). Thus command chaining and this means channel program execution is directly influenced by the last data transfer.

The channel command in process is pointed to by the command pointer in CPR. Therefore, if for any reason the channel is stopped or waiting, the CPR points to the last executed channel command. Only if termination processing is executed, the pointer is incremented (by the length of the current command block) and the next channel command is fetched and loaded into CCR. This can be a type 1 channel command or a type 2 channel command.

Figure 61
Complexities in Command Chaining



Type 1 Channel Command Chaining

A type 1 channel command is indicated by the two highest order bits of the fetched command word (SYN bits in CCR).

After erasing the channel status register CSRn the processing of type 1 command chaining depends on the mode of synchronization (SYN bits in CCR):

- In case of internal synchronization the channel starts data transfer immediately after the setup routine.
- In case of external synchronization the channel is waiting for requests after completion of the setup routine.

Type 2 Channel Command Chaining

A type 2 channel command is fetched during command chaining, if the two highest order bits of fetched word (SYN bits in CCR) are both zeros. In that case, the whole command is immediately fetched and executed and – if channel is not stopped – the next channel command (type 1 or 2) is chained.

Possible type 2 commands:

- relative jump
- absolute jump

  These jumps are conditional commands. Therefore jump commands have two ways of execution:

  - Condition is true:
    The next channel command is fetched with the new command pointer.
  - Condition is not fulfilled:
    The command pointer is incremented by 6 and the next channel command is fetched

  In both cases the following channel command may again be a conditional command. This implies that multiple evaluation of termination status is possible.

- unconditional stop
- conditional stop with valid condition (channel is only stopped if condition is true, otherwise CPR is incremented by 6 and next channel command is fetched).
- conditional stop without condition (= NOP).

If the channel is stopped, the command pointer is *not* incremented thus pointing to the last executed channel command. Only the channel's DMST bits in GSR are changed from "organizational processing" to "channel inactive" (with or without request pending). Channel program execution is finished.

All type 2 channel commands have an ED bit and an IT bit. If ED = 1, an $\overline{\text{EOD}}$ pulse lasting 2 internal clock cycles (T-states) is issued on the channel's $\overline{\text{EOD}}$ line after command execution. If IT = 1 and if the channel's interrupt is enabled in GMR, the INTOUT function is activated and the channel's interrupt pending bit INT in GSR is set.

6.4 Initialization

6.4.1 Initial State

To bring the SAB 82257 to a defined state the RESET signal has to be activated.

Upon activation of the RESET signal:

1. All channels are disabled by clearing the DMST bits in GSR.
2. All activities on the bus are stopped.
3. All tristate signals are tristated and other output signals are passive.

After the RESET signal becomes inactive, the SAB 82257 continues to be in the state defined above. Additionally the SAB 82257 is in a defined state characterized by:

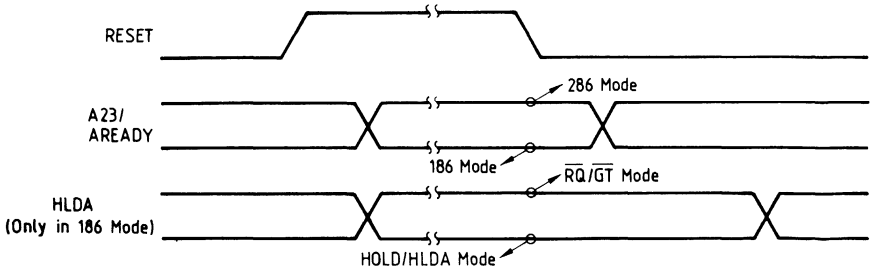
4a It is in 186 mode, if the A23 pin is low at the falling edge of RESET, otherwise it is in 286 mode.

4b When in 186 mode, it is in request/grant mode if the HLDA pin is high at the falling edge of RESET, otherwise it is in HOLD/HLDA mode.

5. Some registers have defined values or defined control bits.

Registers:	GBR:	zero value
	GDR:	zero value
Bits:	GMR:	all bits zero, i.e. even RM bit = 0
	GSR:	all bits zero
	CSRn:	all bits zero

Figure 62
Operation Mode after RESET



All other registers and bits are undefined.
Before any data transfer can be performed by a channel after the RESET signal has occurred

- the SAB 82257 first has to be initialized (programmed by the CPU) and
- after a START command the SAB 82257 has to set up the new channel state (setup routine).

6.4.2 Initialization and Channel Invoking

Figure 63 illustrates how the CPU should initialize the SAB 82257 to ensure correct operation. The first register which must be loaded after RESET is the general mode register (GMR). The main configuration information required by the SAB 82257 for overall processing and remaining unchanged during an installation (e.g. the physical bus width) is indicated in the lower byte of the GMR. Thus the main system configuration information can be loaded first, regardless of the fact whether the CPU has an 8-bit or 16-bit data path or the physical bus width is 8 or 16 bit.

The physical bus width of CPU/SAB 82257 communication is indicated by the SYSBUS/MEMBUS bit of GMR's configuration byte (both in local mode and in remote mode). Thus all register write and read operations (SAB 82257 is slave) are executed

- bytewise, if SYSBUS = 0, on lower half of data bus (D7 to D0) and
- wordwise, if SYSBUS = 1, on D15 to D0.

Note
If SYSBUS = 1, byte transfers are also possible. The bytes are then transferred on that half of data bus which is addressed by the least significant bit of the register address.

Internally the SAB 82257 only uses the bus signals $\overline{BHE}$ (bus high enable) and A0 to detect the effective transfer width (byte or word) of CPU/SAB 82257 communication (see chapter "Bus Operation").

After the general mode register (GMR), the

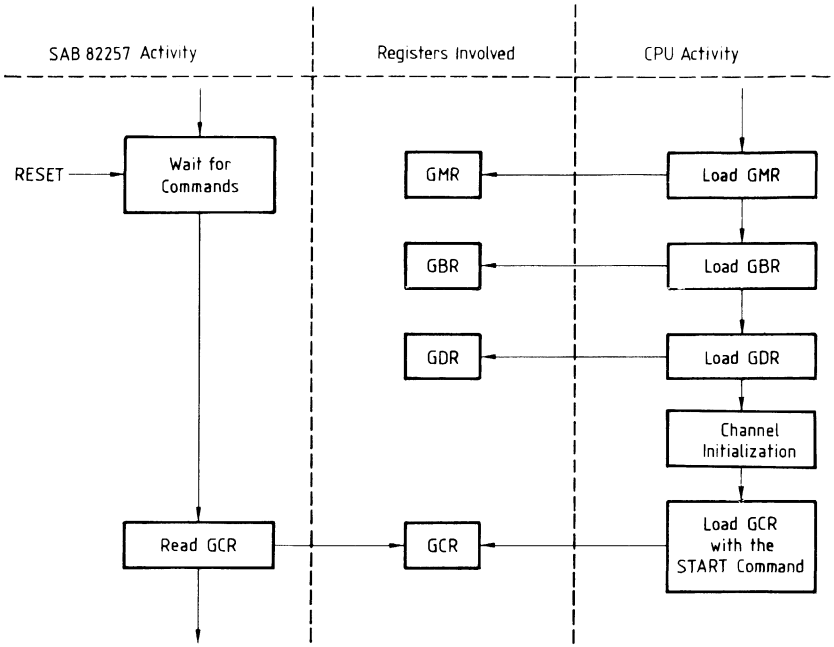
- general burst register (GBR) or the
- general delay register (GDR)

should be programmed by the CPU.

Since the initial state of these registers is zero, the GBR and GDR need only be programmed if the zero entries should be changed.

Before a channel will be invoked, additionally the control space in memory and channel registers within the SAB 82257 have to be prepared and programmed, as described in the following section.

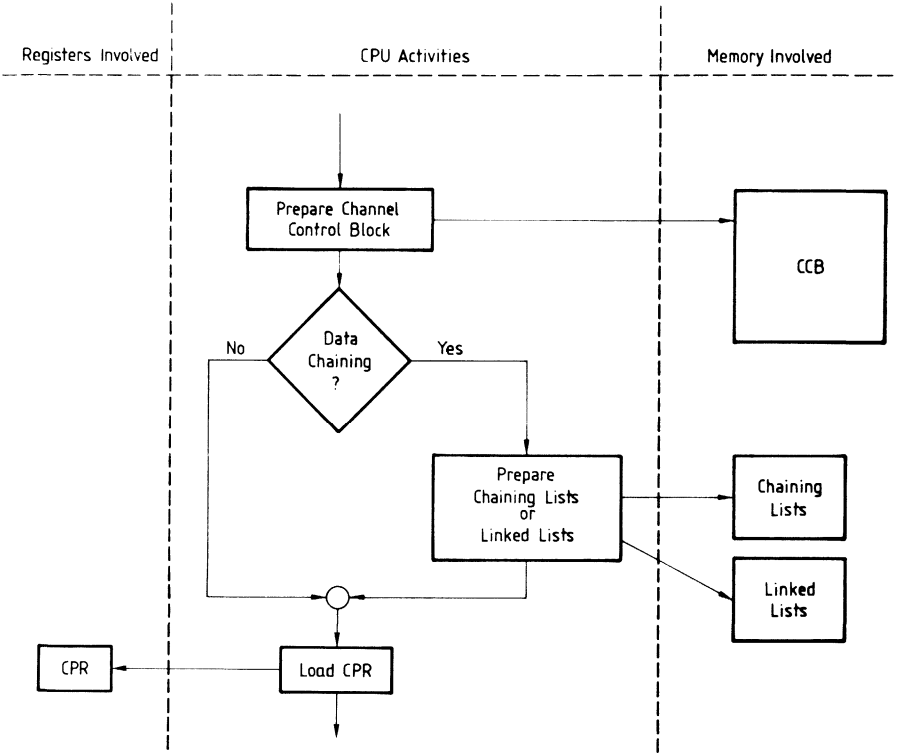
Figure 63
Initialization Sequence



Preparation for Channel Start

- If data chaining enabled: chaining list or linked lists in control space
- Load CPR with start address of channel program

Figure 64
Channel Initialization



As shown in figure 63, in all cases of channel start the last CPU operation is to write the general command into GCR. Then the start will be processed by the SAB 82257 according to the requested channel's priority. If more than one channel should be started at once, at first the one with highest priority is processed. If the addressed channel is already active, the start command is ignored. If I = 1 in the general command, the pending interrupt bit(s) of the indicated channel(s) will be cleared in GSR.

START Command Execution

The START command for a channel initiates the setup routine – executed as described in section 6.4.3 – and, in addition, it defines the location of the control space (memory or I/O space in local mode, system or resident space in remote mode).

6.4.3 Setup Routine

The setup routine causes the SAB 82257 to read the channel command block with all its parameters from memory into the internal channel registers. It includes the following operations:

1. Reset channel status register (CSR).
2. Load channel command into CCR with address out of CPR. Bus width is physical SYSBUS/MEMBUS width (system or memory control space), or RESBUS/IOBUS width in case of resident or I/O control space.
3. If channel command is a type 2 command: continue with type 2 command execution and command chaining.
4. If channel command is a type 1 command: check on fatal command error; if no error:
5. Read source pointer, 24 bit, and load it into SPR: concurrently, the low-order 16-bit word of source pointer is also loaded into DAR (used as literal or constant instead of source data, if source INC/DEC is 1 1 in CCR);
   Additionally the source pointer is loaded as list pointer into LPR, if source data chaining is enabled (SC = 1 and LC or LLC is set).
6. Read destination pointer, 24 bit, and load it into DPR;
   Additionally the destination pointer is loaded as list pointer into LPR, if destination data chaining is enabled (SC = 0 and LC or LLC is set; not allowed for single-cycle transfer).
7. If no data chaining: read byte count, 24 bit, and load it into BCR.
   If data chaining is enabled:
   - Read byte count, 16 bit, with address out of LPR and load it into BCR; clear high byte of BCR.
   - Read data pointer, 24 bit, out of chaining list and load it into SPR in case of source data chaining (SC = 1) or into DPR in case of destination data chaining (SC = 0).
   - If list chaining enabled (LC flag): increment LPR by 6; if linked list chaining is enabled (LLC flag): read new list pointer, 24 bit, and store it in LPR.
8. Change general status of channel (DMST bits in GSR) from "organizational processing" into "DMA transfer in progress".

Now the channel setup is finished and all transfer parameters are accessible in internal registers.

The setup routine is executed after receiving the general start command.

Note
After receiving the general commands
- STOP channel(s)
- CONTINUE channel(s) operation
- HALT/single-step channel(s)

only a change of general and specific channel status is performed, i.e. no channel setup routine is executed.

6.5 Reflections on Compatibility

When designing software to operate the SAB 82257 in a system environment, you might want to provide compatibility with future DMA controllers. This can be achieved by observing a few principles while writing your own software.

Special attention should be paid to the 24-bit quantities (pointer, byte count) used with the SAB 82257. 24-bit quantities within command blocks are placed into two 16-bit words with the most significant byte (MSbyte) left unused. 24-bit registers will normally be accessed as 16-bit word and consecutive byte.

For reasons of compatibility with future DMA controllers it is recommended to treat the MSbytes as zeros.

This means that within command blocks these MSbytes should be set to zero or discarded (e.g. status block), respectively. While writing or reading SAB 82257 registers, only word accesses should be used. Hereby the MSbytes again should be zero (write) or discarded (read), respectively.

By observing these principles you will be able to run user software written for the SAB 82257 also on future DMA controllers (e.g. 32-bit controllers) without any changes. This applies to control programs executed by the CPU as well as to channel programs executed by the SAB 82257.

Note

- Software designed uniquely for the SAB 82257, of course, is not subject to the above rules. No compatibility, however, will be provided in this case.
  The SAB 82257 itself will not bother if the MSbytes contain something else than zeros. Information contained in the MSbytes, however, will be lost or invalid.
- For compatibility with the advanced DMA controller SAB 82258 please refer to section 2.5 "Upgradability".

DMA Transfer

7 DMA Transfer

7.1 General

After initialization and execution of the setup routine all transfer parameters are accessible in internal registers. The SAB 82257 is now prepared for executing the data transfer.

Data is transferred from a source to a destination and can be controlled concurrently by four DMA channels. The source and destination may be any location in memory space or in I/O space. The SAB 82257 operations apply to both, memory components and I/O devices.

Individual transfer cycles (e.g. the movement of a byte or a word) may be synchronized by a signal (DMA request) from the source or from the destination. In synchronized mode, the channel waits for the synchronizing signal before starting the next transfer cycle. The transfer may also be unsynchronized (free-running), in which case the channel begins the next transfer cycle immediately upon completion of the previous one.

A transfer can be terminated on several programmer-specified conditions. The channel can stop the transfer when a specified number (up to 16 Mbytes without chaining) of bytes has been transferred. An external device may stop a transfer by signalling on the channel's terminate pin ($\overline{EOD}$ signal).

Basically the following **2 DMA transfer types** have to be distinguished:

- two-cycle transfer on DMA channel,
- single-cycle transfer on DMA channel,

Two-Cycle Transfer on DMA Channel
The two-cycle DMA transfer is the preferred transfer mode of the SAB 82257. The two cycles consist of a fetch cycle from source location and a store cycle to destination location. The DMA channel can transfer up to 4 Mbytes/s in this mode and has a lot of flexible options. Since the SAB 82257 operations apply to both, memory components and I/O devices, data can be transferred from memory to memory, memory to I/O device, I/O device to memory and I/O device to I/O device, whereby memory as well as I/O space are addressed by 24-bit pointers.

In remote mode the pointers can be either system space or resident space pointers. The pointer to each space can be auto-incremented, auto-decremented or remain unchanged during transfer.
Transfer can be synchronized by source or destination or not synchronized at all (free-running). There is an automatic assembly/disassembly of data, to support transfer between an 8-bit space and a 16-bit space.
The data chaining feature allows to gather two or more source blocks and transfer them to one destination or to scatter one source block to several destination locations. The maximum block size for normal transfers (unchained) is 16 Mbyte (BCR is a 24-bit register).

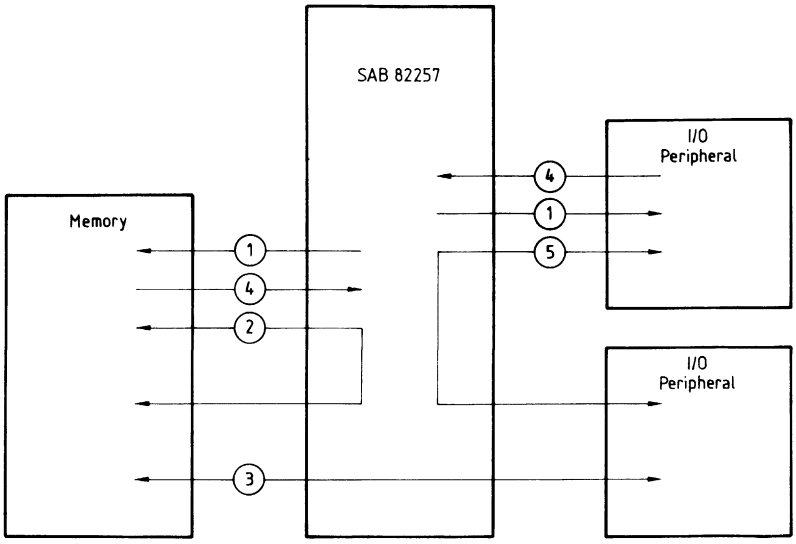
During data chaining blocks of up to 64 Kbyte maximum can be chained together. Termination of block transfer can be controlled internally (byte count expired) or externally ($\overline{EOD}$ signal).

Special modifications of two-cycle data transfers are:

- **DMA transfer with no destination pointer** (read operation), i.e. data byte/word is read into the DAR register of the SAB 82257 but is not output.
- **DMA transfer with no source pointer** (write operation), i.e. data byte/word is output from the DAR register of the SAB 82257 (loaded during the setup routine).

Characteristics:
- data flows through the SAB 82257
- dissimilar bus width support
- memory-to-memory transfers
- transfer rate up to 4 Mbytes/s

Figure 65
Two-Cycle Transfer

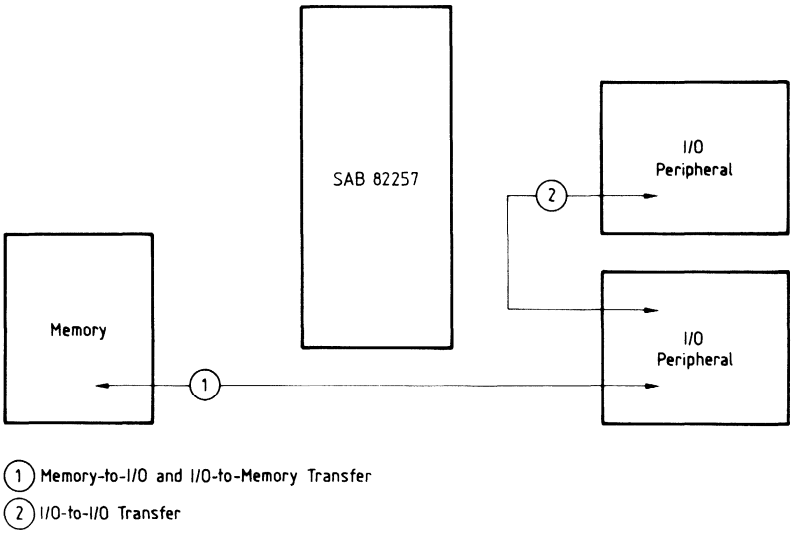


(1) DMA Transfer with No Source Pointer
(2) Memory-to-Memory Transfer
(3) Memory-to-I/O and I/O-to-Memory Transfer
(4) DMA Transfer with No Destination Pointer
(5) I/O-to-I/O Transfer

Single-Cycle Transfer on DMA Channel
The single-cycle DMA transfer supports very fast peripherals at more than 4 Mbytes/s transfer rate. The maximum speed in this mode is 8 Mbytes/s. Data is directly transferred from source to destination, i.e. the data flows past (not through) the DMA controller and cannot be assembled or disassembled on-the-fly. Therefore dissimilar bus width between source and destination is not supported. Single-cycle transfer is always performed with external synchronization by DMA request (DREQ) from an I/O device. The maximum transfer speed is reached only with a continuous DMA request.

While the requesting device is serviced (and addressed) via the acknowledge signal $\overline{\text{DACK}}$, the pointer to the other location (memory or I/O) is issued and its bus cycle is executed by the SAB 82257. It is the I/O device's duty, due to pin limitation, to know whether its cycle is a read cycle or a write cycle and to generate its command signal out of the bus command signals. Since $\overline{\text{DACK}}$ covers the whole command signal – even if it is generated by the bus controller – all related timing conditions of peripherals are fulfilled.

Figure 66
Single-Cycle Transfer



Single-cycle transfers can be performed on the system bus also in remote mode, if the peripheral is local and the memory in system space.
This is very advantageous for a high transfer rate between local space and system space, since the system bus arbitration has to be performed only once for every continuous DMA request, e.g. transfer from a local FIFO memory buffer to system memory. During a two-cycle data transfer, a change of system bus arbitration would be necessary for each bus cycle.

On single-cycle transfer channels, data chaining and command chaining is supported, too. Termination of single-cycle transfer can be controlled internally (byte counter expired) or externally ($\overline{EOD}$ signal).

Characteristics:
- data flows past the SAB 82257
- external synchronization by DREQ signal
- transfer rate up to 8 Mbytes/s

7.2 Synchronization of Data Transfer

7.2.1 Survey

Two general modes of synchronization for DMA transfer on the SAB 82257 have to be distinguished:

- **External synchronization**
  via the DMA request pins (DREQ 0 to 3).
- **Internal synchronization**
  i.e. free-running data transfer.

The preferred mode of synchronization is the **external synchronization** of DMA transfers. The external DMA request may be issued from source (source synchronization) or from destination (destination synchronization) device. It can initiate a single or a two-cycle DMA transfer. The external synchronization allows to control input/output operations in the cycle of the peripheral device and to occupy the bus only at that time when the peripheral is really able to receive or to transmit data.

The second general mode of synchronization is the **internal synchronization** of DMA transfers (free-running). This means that a DMA transfer is performed without any external synchronization. This is advantageous for memory-to-memory transfers (two-cycle transfers). Free-running data transfer is also performed during a continuous DMA request (one or two cycles.

7.2.2 Associated Control Register Bits

The following control register bits are used in order to control synchronization of DMA transfers:

- General Mode Register Bits (GMR)
  - CYCn (bits 4, 5, 6, 7): Two-cycle/single-cycle transfer mode of channel n.
- Channel Command Register Bits (CCR)
  - SYN (bits 14, 15): Source/destination synchronization or free-running.

7.2.3 External Synchronization via DREQ Signals

A DMA request initiates the execution of one or – in case of continuous request – more DMA cycles (see timing diagrams in chapter "Device Specifications"). Depending of the channel's CYC bit in GMR one or more

- single-cycle transfer or
- two-cycle transfers

are performed.

In case of **single-cycle DMA transfer** the following assumption is made:

requesting device: = I/O device.

Therefore the SYN bits in the channel command register (CCR) are only needed to identify the requesting device – source or destination. Thus the other location – destination or source – has to be addressed during data transfer. This addressing is done with a pointer out of the source pointer location in the command block.
During single-cycle transfer, no assembly or disassembly of bytes can be performed. A DMA request, therefore, always initiates the whole byte or word transfer from source to destination.
If the DREQ signal is not cleared after acknowledge with a $\overline{\text{DACK}}$ signal, more than one DMA transfer will be performed (see section 5.3.1 "Communication via DREQn/$\overline{\text{DACKn}}$ Signals"). This results in the highest possible transfer rate for the SAB 82257.

In the case of **two-cycle DMA transfer** no assumption is made concerning the requesting device.

Since in two-cycle transfer mode the assembly/disassembly function is enabled, the SYN bits are needed to control the real number of bus cycles to be performed with one DMA request. An example shall illustrate this:

Example
It has been assumed that source transfer width is 16 bit (W) and destination width is 8 bit (B).
- In the case of source synchronization three bus cycles will be executed with one DREQ (1W, 2B).
- In the case of destination synchronization only one (1B = first byte) or two (1B = second byte, 1W) data cycles will be executed with one DREQ. Note that in this case the SAB 82257 performs prefetching of source data.

Note
- As described in the subsequent sections, control of the real number of bus cycles during "two-cycle transfer" is rather complicated, because this also depends on such parameters as address (odd/even), modification (increment/decrement), byte count, etc.
- A continuous DMA request causes free-running DMA transfers until the DREQ signal is cleared or a termination condition occurs.

7.2.4 Internal Synchronization

The original control of the free-running transfer mode is done internally and is selected via the SYN bits in CCRn. In this case, the DMA channel works in NOSYNC mode. After finishing the setup routine, the channel starts the data transfer by itself. Thereby the channel controls

the data transfer according to the assembly/disassembly rules, but without any external synchronization. (The $\overline{\text{READY}}$ signal may be used as a kind of external synchronization). Thus the channel performs bus transfers and runs with maximum speed – unless interrupted by a higher priority request – until a termination condition occurs.
As described in preceding sections, the start of the NOSYNC mode can also be triggered by **external synchronization**:

Continuous DREQ
In this case the free-running data transfer will be performed until reset of DREQ or until a termination condition occurs. In this case – and only here – also free-running or single cycle transfers are possible.

7.3 Masking of Transfer Requests

A data transfer is executed only if – after the synchronization to the internal clocking system – the corresponding transfer request is not masked by its associated control register bits.

Associated Control Register Bits

The following control register bits are used for masking transfer requests:

- General Status Register (GSR)
  DMST bits: Channel stopped or in organizational processing, or DMA transfer in progress.

Transfer requests on DMA channels are masked with the DMST bits in GSR. Internal or external transfer requests are only accepted if the channel's status is "DMA transfer in progress".

Requests are masked as long as the channel is stopped or in organizational processing.

Since a stop command from the CPU directly effects the channel's DMST bits, the data transfer will also be stopped immediately. On the active channel, the DMST code "in organizational processing" expresses all channel states where data transfer is not allowed.

Example
After a channel start during setup routine, or after termination during command chaining, requests are masked.

An unmasked transfer request can also be delayed by the priority control logic and by bus arbitration.

7.4 Bus Request Control

Before any data transfer is executed, the bus must be allocated to the SAB 82257 otherwise a bus arbitration with a HOLD/HLDA sequence has to be performed.
Only exception: resident bus accesses in remote mode.

7.4.1 Associated Control Register Bits

The bus request (HOLD) is controlled by the following parameters:

- General Mode Register (GMR)
  - RM (bit 2): SAB 82257 operates in local/remote mode.
- General Burst Register (GBR)
- General Delay Register (GDR)
- General Status Register (GSR)
  - S/Rn bits: SYS/RES for bus requests to control space, used in remote mode.
- Channel Command Register (CCR)
  - M/IO (bit 0, bit 4): For source and destination: in remote mode used for system/resident space indication.

7.4.2 Bus Request Control in Local Mode

In local mode an unmasked transfer request immediately activates or renews the bus request signal HOLD. Unless HLDA is already active, the data transfer has to wait until the CPU sends the acknowledge signal. In case of continous transfer requests, either internal (free-running) or external, also the bus request is a continuous request even if the transfer is interrupted by a higher priority channel request.

A new HOLD request or a continuous HOLD request may also be delayed or interrupted by programmed general delay and general burst. The GBR and GDR are enabled if they are programmed with values not equal to zero. In that case, a continuous bus request is limited to the programmed number of contiguous bus cycles. After such a transfer interrupt the general delay counter loaded from GDR becomes active.

This additional delay has to be considered with respect to the maximum latency time of highest priority DMA requests (see section 6.2.3 in chapter "Programming and Control").

7.4.3 Bus Request Control in Remote Mode

In remote mode bus requests are generated **only** for system bus accesses. For these accesses the SAB 82257 generates HOLD before getting onto the local bus. Only when it gets a HLDA it starts the bus cycle and occupies its local bus.
System bus accesses are indicated by the M/IO bits in CCR for source and destination.
For maximum system bus transfer rate a continuous system bus request is necessary. The SAB 82257 in remote mode can only accomplish a continuous system bus request (HOLD) if either

- single-cycle transfer with pointer in system space is used with continuous DMA request, or
- two-cycle transfer is specified and both, source and destination, are located in the system space, combined with continuous external or internal DMA request.

In both cases, holding of system bus can be limited and controlled by programming GBR and GDR.
The GBR and GDR are enabled if they are programmed with values not equal to zero. In that case, a continuous bus request is limited to the programmed number of contiguous bus cycles. After such a transfer limitation the general delay counter loaded from GDR becomes active.

In remote mode resident bus accesses are **not** arbitrated. Thus a minimum latency time can be achieved after a new DMA request. In all other cases, DMA requests need at first a bus arbitration. The data transfer can be accomplished according to the synchronization and assembly/disassembly principles.

7.5 Data Transfer

There are two data transfer types which can be performed by each of the four high-speed channels:

- **Single-Cycle Transfer**
  In single-cycle mode the bytes or words (16 bit) are transferred directly from data source to data destination in a single bus cycle per transfer. This mode allows a total data rate of up to 8 Mbytes/s, and that as single-channel data rate or as cumulative data rate of multiple channels (simultaneous operation of several channels in single-cycle mode). Thus the advantage here is that of speed.
- **Two-Cycle Transfer**
  In two-cycle mode the source data is always stored in the SAB 82257 before being sent out to the destination. A special feature of two-cycle transfer is automatic assembly and disassembly of data in bytes and words, meaning that data can be read as one 16-bit word and written as 2 bytes, or vice versa. This is often desirable when using 8-bit wide peripherals in a 16-bit system. In two-cycle transfer mode, data may also be transferred from one memory region to another – which is impossible with single-cycle transfer.

7.5.1 Two-Cycle Transfer

Associated Transfer Parameters and Control Register Bits

The following registers and control register bits are used in order to control data transfer cycles for two-cycles transfer mode:

- General Mode Register (GMR)
  - MEMBUS/SYSBUS (bit 0): Physical system/memory bus width.
  - IOBUS/RESBUS (bit 1): Physical resident / I/O bus width.
  - RM (bit 2): Local/remote mode.
  - CYCn (bits 4, 5, 6, 7): 0, i.e. two-cycle DMA transfer mode of channel n.
- Channel Command Register (CCR)
  - M/IO (bit 0, bit 4): For source and destination: Data in I/O / memory space for local mode. Data in resident/system space for remote mode.
  - INC (bit 2, bit 6): DEC (bit 1, bit 5): For source and destination: No modification, increment, decrement or don't use pointer.
  - W/B (bit 3, bit 7): For source and destination: Logical bus width byte/word.
- Source Pointer Register (SPR)
- Destination Pointer Register (DPR)
- Byte Count Register (BCR)

Transfer Width

Logical Bus Width

The logical bus width W/B (bit 3 and bit 7 in CCR) determines the quantity – bytes or words – which should be transferred of the bus during a data bus cycle.
Thereby

- the logical bus width of source (bit 3) determines the source quantity (not data type) which should be transferred from source location to SAB 82257 and
- the logical bus width of destination (bit 7) defines the quantity to be transferred from SAB 82257 to destination.

This independent indication of logical bus width for source and destination implies that the destination quantity need not be the same as the source quantity.

Physical Bus Width

- In local mode
  the physical bus width is determined by the GMR bits MEMBUS (bit 0) and IOBUS (bit 2).
  Thereby
  - the MEMBUS bit defines the physical bus width of the memory bus, and
  - the IOBUS bit that of the I/O bus.
- In remote mode
  the physical bus width is determined by the GMR bits SYSBUS (bit 0) and RESBUS (bit 1).
  Thereby
  - the SYSBUS bit defines the physical bus width of the system (and processor) bus, and
  - the RESBUS bit that of the resident bus.

Before any data bus cycle is executed, the logical bus width of the addressed location is compared with the related physical bus width. The following table describes the effective transfer width and the data bus pins used, in relationship to physical and logical bus width.

Table: Effective Transfer Width

Physical Bus Width	Logical Bus Width			
	8 Bits		16 Bits	
	Effective Transfer Width	Data Pins	Effective Transfer Width	Data Pins
8 Bits	8-Bit Byte-Transfer	D7 to D0	8-Bit Byte Transfer	D7 to D0
16 Bits	8-Bit Byte Transfer If A0 = 0 If A0 = 1	 D7 to D0 D15 to D8	16-Bit Word Transfer	D15 to D0

Note

- In the above table the indicated word transfer only defines the quantity which should be transferred. The real quantity which is transferred by a data bus cycle depends additionally on other influences, such as address (odd/even), byte count and the direction of modification. All this results in the data cycle control.

- Using a logical bus width of 8 bit on a 16-bit physical bus, bytes are transferred alternatively via the lower (even address) or the higher (odd address) half of the bus depending on the LSB of the address (A0) (See also chapter 4, section "Bus High Enable").

Data Cycle Control

Dissimilar effective transfer widths and special transfer conditions influence the real quantity transferred during data cycles within two-cycle transfers. Thereby the data width of source cycles and destination cycles is controlled by the assembly register operations. The SAB 82257 has one 16-bit data assembly register (DAR) per channel. All source data are loaded into the DAR, manipulated in some cases (e.g. swap) and then transferred to the destination.

For data cycle control, two transfer modes have to be distinguished:

- **Forward Transfer Mode**
  Data transfer mode with incrementation of source pointer and/or destination pointer (both pointers may be steady).
- **Reverse Transfer Mode**
  Data transfer mode with decrementation of source pointer and/or destination pointer (one pointer may be steady).

Explanation
The following symbols will be used within this section:

B	= Byte transfer
B,B	= Two successive byte transfers
W	= True word transfer
B/B	= Artificial word transfer
⟶	= Normal transfer
⟶x⟶	= Transfer with possible swap
S	= Source pointer
D	= Destination pointer
X+	= Incrementation of X pointer
X=	= X pointer remains unchanged
X−	= Decrementation of X pointer

Transfer Acceleration (align):
Transfer acceleration is executed if the pointer to a 16-bit location contains an odd address and is incremented. In this case, a single byte is transferred which results in an even address in the pointer mentioned. Now word transfers are possible with increasing throughput (= acceleration). The double-framed fields of the table mark the cases where transfer acceleration is possible.

Exception
If the other pointer also points to a 16-bit location and holds an even address, no acceleration is performed because it would be pointless in this case.

Swapping of DAR Bytes

During reverse transfer the bytes of the DAR may be swapped logically to maintain the order of data during transfer. Swaps are performed in the indicated cases (⟶x⟶).

Artificial Word Transfer

An artificial word transfer is regarded as word transfer but realized as two successive byte transfers. Pointer modification, however, is performed as with word transfers. Contrary to two-byte transfers (B,B) the byte transfers of an artificial word transfer are unseparable bus cycles and are initiated by a single DMA request. The B,B transfer requires two DMA requests, if it is synchronized.

This table surveys the possible modes of data cycle control.

Table
Channel Transfer Operation Survey

S → D	8 → 8	8 → 16	16 → 8	16 → 16
E → E	B → B	B,B ⤫ W	W ⤫ B,B	W → W
E → 0	B → B	B,B ⤫ B/B	W ⤫ B,B	W → B/B
0 → E	B → B	B,B ⤫ W	B/B ⤫ B,B	B/B → W
0 → 0	B → B	B,B ⤫ B/B	B/B ⤫ B,B	B/B → B/B

Forward Transfer Mode

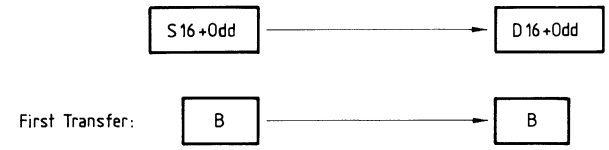
The forward transfer mode is the standard mode of data transfer. Forward transfers are data transfers with incrementation of source and destination pointer (S+ and D+). They also include constellations where one pointer is incremented (S+ or D+) and the other remains unchanged (D= or S=). If both pointers are not modified (S= and D=) also a forward direction of data transfer will be assumed. Thus, the forward transfers include data transfer with pointer modifications such as

S+ → D+
S+ → S=
S= → D+
S= → D=

- The table on page 141 shows that in case of **odd addresses** (with pointer incrementation) combined with 16-bit transfer width at first only **a byte transfer** is executed. After this byte transfer and incrementation of pointers, even addresses allow whole 16 bit transfers.

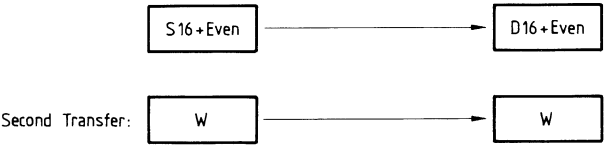
Example
Source and destination transfer width are 16 bit, both pointers have odd addresses and are incremented.

Figure 67



After initial byte transfer both pointers are modified by 1 (byte transfer). Now both pointers have even addresses and the following transfers are word transfers (after each word transfer both pointers are modified by 2). The last transfer is a byte transfer, if the original byte count was even (word transfer with original byte count odd).

Figure 68



Note
This kind of transfer acceleration cannot be used if S16 and/or D16 should remain unchanged (S16 = and/or D16=).

During each transfer the byte count register (BCR) is decremented according to whether a byte (decrementation by 1) or word (decrementation by 2) is transferred. Thus the BCR always indicates the number of bytes that still are to be transferred. Word transfers or corresponding B/B transfers are only performed, if the byte count register contains at least

the number 2. If the byte count is one, only a byte transfer is executed before termination or data chaining. This is also valid during prefetching in case of destination synchronization.

- If the transfer which is 16 bit, B/B transfers are chained together (unseparable).
- If the transfer width is 8 bit, B/B transfers can be chained but this depends on synchronization control.

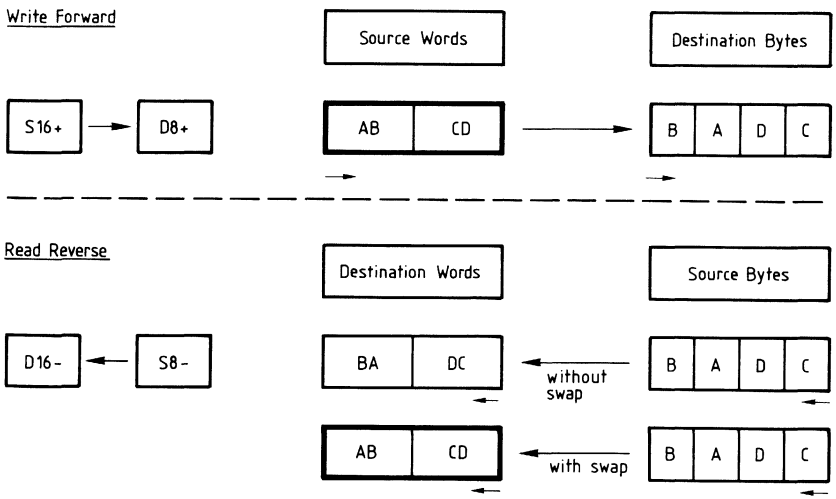
Reverse Transfer Mode

Reverse transfers are data transfers with decremented of source and/or destination pointer (S– or D–). They also include constellations where one pointer is decremented (S– or D–) and the other remains unchanged (D= or S=). Thus, the reverse transfers include data transfers with pointer modification such as

S– → D–
S– → D=
S= → D–

For reverse data transfers, it is assumed that the operation "write forward" followed by a "read reverse" must not change the original byte string. Therefore, a swap of bytes has to be executed if source and destination have dissimilar transfer width.
The following example illustrates the necessity of a swap.
The example in figure 69 shows only the logical swap of bytes, which is necessary in comparison with forward transfers. Physically it may also be possible that no swap has to be performed under the following condition: If read reverse of source bytes is started with an odd source address and physical source bus width is 16 bit.

Figure 69
Example of Reverse Transfer



During each transfer, the byte count register (BCR) is decremented according to whether a byte or word is transferred. Thus the BCR always indicates the number of bytes still to be transferred. If the byte count is one, only a byte transfer is executed before termination or data chaining.

Note

- B/B transfers on 16-bit transfer width:
  Address is incremented (not decremented) after first byte transfer (usually word transfer with odd addresses). Thus in case of decrementation the initial word address does not also address the highest byte.
- Contrary to incrementation:
  If the byte count is 1 after transfer on 16-bit bus, then the pointer is decremented by 1 for the next (byte) transfer. Nevertheless, swap is performed in cases of —x→
- If no modification has to be performed on pointer to 16-bit location and byte count is 1 after transfer on 16-bit bus, then the pointer, nevertheless, is incremented by 1 for the next (byte) transfer. Contrary to forward transfers this is the only restriction to be considered on a 16-bit bus without address modification.

Standardized Handling

Basically all two modes of data cycle control are covered by some standardized handling:

- Assembly/disassembly of bytes supports especially transfers on buses with dissimilar effective transfer widths, where 16-bit locations are addressed by even addresses.
- Transfer acceleration is supported only in case of odd address to 16-bit locations if the related pointer is incremented (S16+ of D16+). No transfer acceleration is supported in case of odd address to 16-bit locations if odd pointers are decremented (S16− or D16−) or if odd pointers are not modified (S16= or D16=).
- The entry direction of source bytes into the data assembly register DAR is in all cases controlled by the source pointer modification direction:
  - If the source pointer is really or implicitly (forward transfer) *incremented,* the bytes are loaded alternately into the low and high-byte position of the DAR, beginning with low byte.
  - If the source pointer is really or implicitly (reverse transfer) *decremented* and effective transfer width is 8 bit, bytes are loaded beginning with the high-byte position of DAR, which results in swapping of the DAR bytes.
- The output of bytes is controlled by the destination pointer modification direction for all B/B data cycles to destination:
  - If the destination pointer is really or implicitly *incremented,* bytes are alternately fetched from DAR byte positions, beginning with low byte.
  - If the destination pointer is really or implicitly *decremented* and effective transfer width is 8 bit, B/B transfers begin always with the high byte of DAR.
- No swap is performed during 16-bit source or destination data cycles.
- B/B and W transfers are performed only if the byte count contains at least number 2. When the byte count contains 1 only a byte transfer is executed before chaining or termination.

Note
The described data cycle control in this section is **not** valid
- in single-cycle transfer mode and
- in cases where no source or destination pointer is needed at all (reverse byte swap is executed).

Addressing
Source and destination are addressed by 24-bit real addresses, the source pointer and the destination pointer. Logical addressing is not supported. Addresses are always byte addresses. Thus a real word transfer requires an even address.

- In local mode
  addresses may point into memory or into the I/O space. This can be selected for source and destination separately with the two M/IO bits in type 1 channel command. The M/$\overline{IO}$ signal accompanies all issued addresses. Since in local mode M/$\overline{IO}$ is not used for internal control and I/O data cycles do not differ from memory data cycles, the I/O space may also be mapped into memory space and vice versa.
- In remote mode
  addresses may point to the global or the local space, i.e. the SAB 82257 has to distinguish between system bus accesses and resident bus accesses. This is controlled by the M/IO parameter in the type 1 channel command for source and destination separately. Therefore, mapping has to be performed for I/O accesses on system bus and memory accesses on resident bus.

Pointer Modification
Source and destination data pointer are

- incremented or
- decremented or
- not modified at all

according to the values of INC/DEC bits in type 1 channel command. All modification of pointer registers is done after the corresponding data bus cycle.

In cases of **incrementation** and **decrementation**, the pointer registers are modified by 1 after a byte transfer and by 2 after a word transfer.

In case of **no modification** at all, the pointer remains unchanged even if the address has to be incremented for a second byte transfer on a 16-bit bus with odd address.

Only exception
In case of odd byte count and reverse transfer on a 16-bit bus the pointer is incremented by 1 for the last byte transfer.

Note
- In 186 mode of the SAB 82257 only 20-bit addresses are available and issued onto the address bus. The low-order 16 bits of the addresses are mulitplexed on the data bus, too.
- In both 186 and 286 mode the SAB 82257 does not check and indicate an address out of range condition. Address overflow and underflow during block transfer results in an address wrap around.

Block Length Control with Byte Count

The initial value of byte count determines the number of bytes which should be transferred during a block transfer. The initial byte count is indicated

- in the **channel command block (CCB)**
  containing a 24-bit count, therefore a maximum of (16M-1) bytes can be transferred with one command, or
- in the **list elements** of data chaining list
  containing a 16-bit byte count, therefore a maximum of (64K-1) bytes can be transferred with one list element.

If the initial value of the byte count

- is zero:
  - in the **channel command block** (no data chaining enabled), no data transfer is executed after setup routine and the termination routine is immediately initiated (command chaining);
  - in a **chained list element** (tested during data chaining) data chaining is stopped and block transfer is terminated. Note that in case of data chaining the whole block length is determined by the sum of byte counts of list elements which are processed before a list element with a zero byte count occurs.
- is one:
  - the following data transfer operation is always a byte transfer operation.

In the following case the initial value of byte count has to be even:

In reserve transfer mode, if data chaining is enabled and the effective transfer width of source or destination is 16 bit. In that case all chaining list elements must also contain even byte counts.

Modification of the byte count register (BCR) is always performed

- after destination data cycles in the case of destination synchronization of free-running transfer, or
- after source data cycles in the case of source-synchronized data transfer.

Thereby the byte count register is decremented by one or two according to the transfer quantity – byte or word – during the respective data cycle.

After modification the byte count is checked for whether the contents.

- is one, i.e. the following (last) source fetch is always a byte transfer operation (this is also considered within prefetching),
- or zero, i.e. either termination of data transfer or data chaining is initiated.

Note
In 186 mode only a maximum of (1M – 1) bytes can be transferred with one command. This is not checked by the SAB 82257.

Data Transfer without Destination Pointer – Data Read

A special modification of the two-cycle data transfer is the data transfer without destination pointer. This means, the source data – bytes or words – are read into the data assembly register (DAR) of the SAB 82257, but not written out to a destination. Therefore only the source bus cycle of the two-cycle data transfer is executed.

With the possibility of transfer without destination, the SAB 82257 supports additional I/O control functions:

- Skip
  For example: A peripheral data block in a not random-accessible memory such as a magnetic tape should be skipped.
- Register read
  For example: A peripheral register (e.g. status byte) should be read in.

The transfer with no destination pointer is programmed

- by setting the INC/DEC field for destination equal to 1 1 in CCR (bits 6, 5),
- by defining **no** "destination synchronization" indicated by the SYN bits (bits 14, 15) in CCR,
- by setting all other destination parameters such as W/B, M/$\overline{IO}$, destination pointer in CCR and command block equal to zero.

The control of transfer width, of source cycle, of addressing and pointer modification and of block length with byte count is analog to two-cycle transfer and identical to single-cycle transfer (see section 7.5.2).

Data Transfer without Source Pointer – Write Constant

Another modification of two-cycle data transfer is the transfer without source pointer. Thereby the whole block transfer is performed with the same byte/word constant (literal) out of the data assembly register DAR of the SAB 82257. The DAR is loaded during the setup routine with the low word of the source pointer field out of the type 1 command block.

Thus no source cycles but only destination cycles are executed.
With the possibility of transfer without source the SAB 82257 supports additional I/O control functions:

- Erase
  For example: A memory block or a peripheral data block should be erased or filled with a certain constant.
- Register Write
  For example: A peripheral register (e.g. comand register) should be loaded with a literal.

The transfer with no source pointer is programmed

- by setting the INC/DEC field for source equal to 1 1 in CCR (bits 2, 1),
- by setting all other source parameters such as W/B, M/$\overline{IO}$ in CCR equal to zero,
- by loading the byte/word constant in the low-order word of the source pointer in command block. Note that the low-order word consists either of the word constant or of **two** indentical byte constants.

The control of transfer width, of source cycle, of addressing and pointer modification and of block length with byte count is analog to two-cycle transfer and identical with single-cycle transfer (see section 7.5.2).

7.5.2 Single Cycle Transfer

Associated Transfer Parameters and Control Register Bits

The following registers and control register bits are used in order to control data transfer cycle for the single-cycles transfer mode:

- General Mode Register (GMR)
  - MEMBUS/SYSBUS (bit 0): Physical system/memory bus width.
  - IOBUS/RESBUS (bit 1): Physical resident / I/O bus width.
  - RM (bit 2): Local mode/remote mode.
    Note
    Single-cycle transfers can be performed in remote mode with the system bus, if the peripheral is local and the memory in system space.
  - CYCn (bits 4, 5, 6, 7): = 1, i.e. single-cycle DMA transfer mode of channel n.
- Channel Command Register (CCR)
  Note
  The single-cycle transfer is always programmed with the source parameters only. The destination parameters in the command block have to be all **zero**.
  - M/$\overline{IO}$ (bit 0): Data in I/O / memory space for local mode. Data in resident/system space for remote mode.
  - INC (bit 2)
    DEC (bit 1) : Source pointer modification.

    Note
    INC/DEC = 1 1 (no pointer) is not allowed.
  - W/B (bit 3): Logical bus width word/byte.
  - SYN (bits 15, 14): External control of the direction of data flow:
    SYN = 0 1: (source synchronization) external **write operation.**
    SYN = 1 0: (destination synchronization) external **bus read.**
    SYN = 1 1: (free-running) is not allowed.
- Source Pointer Register (SPR)
  Note
  In single-cycle transfer mode the SPR contains the address – source or destination – which is needed for the transfer.
- Destination Pointer Register (DPR)
  Note
  Has to contain zero for single-cycle transfer mode.
- Byte Count Register (BCR)

Transfer Width

Before any data bus cycle is executed, the logical bus width of the addressed location (W/B bit) is compared with the related physical bus width (MEMBUS/SYSBUS, IOBUS/RESBUS). To identify the effective transfer width with these source parameters the same rules are used as described for two-cycle transfer (see section Two-Cycle Transfer Width).

Since data cannot be assembled or disassembled during block transfer the effective transfer width

- is *always bytes*, if effective transfer width defined by source parameters is 8 bit (byte)
- is *always words*, if effective transfer width defined by source parameters is 16 bit (word).

Data Cycle Control

In single-cycle transfer mode only one data cycle, either the source cycle or the destination cycle, is executed per DMA transfer. Since it has been assumed that the requesting device is an I/O device, the SYN bits in CCR are only needed to identify the requesting device, i.e. the selection of source or destination is controlled by the synchronization bits SYN. If source has to be addressed (destination synchronization), a read cycle is performed, and in case of destination (source synchronization) a write cycle. Addressing has to be considered if a 16-bit physical bus width (MEMBUS/SYSBUS, IOBUS/RESBUS in GMR are equal to 1) is combined with an 8-bit logical width (W/B = 0 in CCR). Since in that case always byte transfers have to be executed (according to the table in section Two-Cycle Transfer Width), the least significant bit of the address pointer determines the byte position on the 16-bit data bus. Therefore, if the address is incremented or decremented, an 8-bit peripheral can only be connected to a 16-bit bus by means of a byte swap logic (e.g. two SAB 8286A components).

Odd or even addressing must also be considered if the effective transfer width is **16 bit**:

Even Addresses	Odd Addresses
Real word cycles are performed (only this makes sense).	Two-byte cycles are executed, if addresses are decremented or not modified. At first, one byte transfer and then word transfers (align case) are executed, if addresses are incremented.

During single-cycle transfer, data is directly transferred from source to destination and the SAB 82257 never issues data by itself. Each data cycle is internally controlled as a read cycle **independent** of whether the executed bus cycle is a write or a read cycle.

Addressing and Pointer Modification

In a channel command block the one address which is needed for single-cycle transfer is always stored at the **source** pointer location. The control bits for memory / I/O space and for address modification (INC, DEC bits) are always used from the **source** part of the CCR. Addressing and pointer modification is executed as described for two-cycle transfer in section 7.5.1.

Only exception
INC, DEC = 1 1 (no pointer) is not allowed.
Address modification is performed after the respective data bus cycle.

Block Length Control with Byte Count

The byte count register is modified after each transfer. The other control principles of block length control are the same as described for two-cycle transfer (including data chaining) in section 7.5.1.

7.6 Data Chaining

Data chaining allows to handle a set of data blocks as one logical block and vice versa.

Source Chaining allows gathering of a series of single blocks and transferring them to one logical block.

Destination Chaining allows scattering one logical data block and transferring the parts to different locations.

The sequence of the single blocks is determined by a link list.

Two types of data chaining are supported:

- list chaining
- linked list chaining

In case of **list chaining** (see figure 70), the link list is a sequence of entries which specify each the length and location of a single data block.
Application:
Transmission or reception of "data packages" consisting of fixed elements like header, control block, data, etc.

In case of **linked list chaining** (see figure 71), the link list need not be continuous. Therefore beneath the length and location of the associated data block each list entry has to specify the location of the next link list entry.
Application:
Vector graphics where different parts of the display are stored in different data blocks and can be added or removed from the display dynamically.

Data chaining is stopped, if the byte count field of the actual list element contains zero.

7.6.1 Associated Control Register Bits and Parameters

Additionally to those of data transfer control, the following control bits and parameters are used in order to perform SAB 82257 data chaining features during data transfer:

- Channel Command Register (CCR)
  - LLC/LC (bit 9, bit 8): 0 0 No chaining.
    0 1 List chaining.
    1 0 Linked list chaining.

    Note
    Only one of the bits LC and LLC must be set.
  - SC (bit 10): Select chaining: destination/source data chaining.

    Note
    The SC bit has to be one for single-cycle data chaining, since the list pointer is always stored on source pointer location in command block in this mode.
- List Pointer Register (LPR).

Figure 70
(Destination) List Chaining

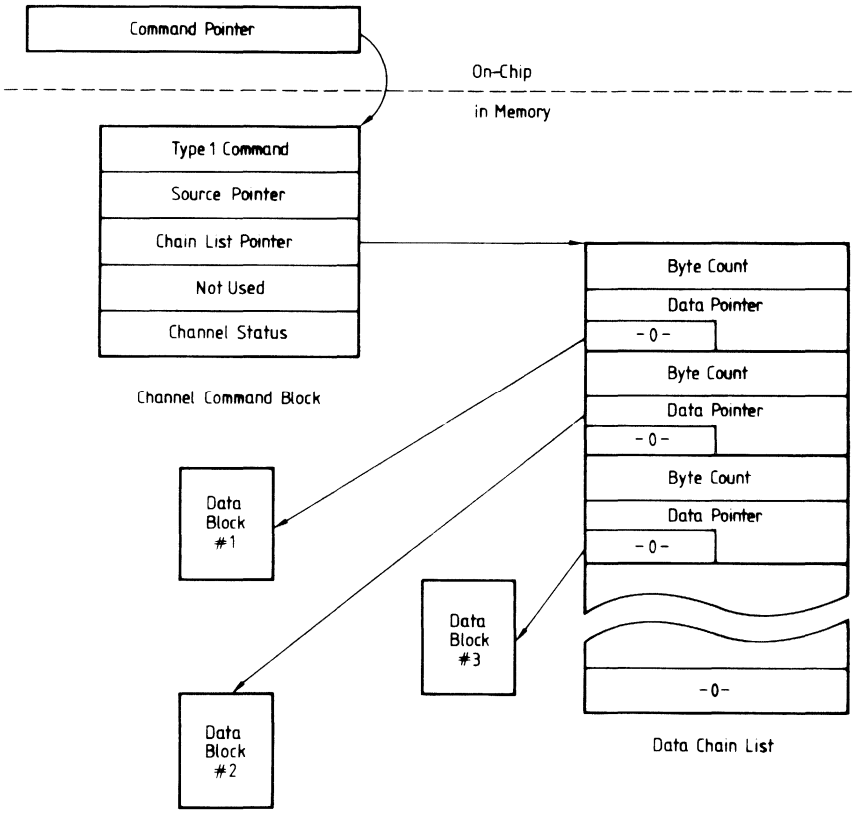
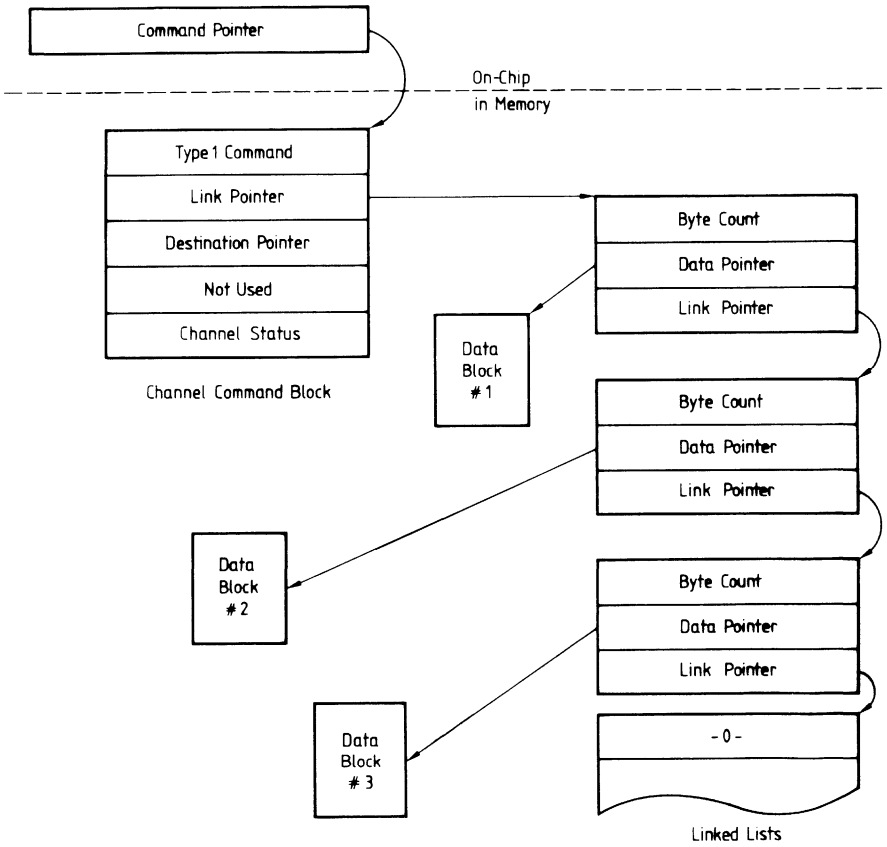


Figure 71
(Source) Linked List Chaining



7.6.2 List Chaining

List data chaining can be enabled by the LC flag (bit 8) in the type 1 channel command, for each channel command block. List chaining is normally used for memory-to-I/O or I/O-to-memory data transfers. Thereby different memory blocks are linked together to form one data block for the peripheral (source chaining), or the data block from the peripheral is scattered to different memory blocks (destination chaining). The different data blocks in memory are specified in the chain list.

The data chain list is built as a sequence of list elements defining data blocks which should be linked together (see figure 73). Each list element consists of three words and contains

- a 16-bit byte count and
- a 24-bit data pointer.

Thus, each list element describes up to 64 KBytes space (or segment) in memory. The data chain list ends with a list element with zero byte count.

If list chaining is enabled in the type 1 channel command, the source pointer (if source chaining, indicated by SC bit = 1) or the destination pointer (if destination chaining, indicated by SC bit = 0) is **replaced** by a pointer to the chain list (see figure 72). While doing data chaining the byte count in the channel command block is not used.

During the setup routine after channel start, data pointer and byte count are fetched from the first list element of the data chain list instead of the command block. Then the list pointer in the list pointer register (LPR) is incremented by 6 to the next list element.

The SAB 82257 processes each list element as an autonomous block transfer, specified with

- data pointer (source or destination) in list element,
- "second" pointer (destination or source) first in command block and then – after first block transfer – in pointer register SPR or DPR),
- byte count in list element,
- unchanged channel command.

Such a block transfer specified by a list element (elementary block transfer) is part of the whole block transfer. Thereby this elementary block transfer is executed and controlled in exactly the same way as specified in section 7.5 "Data Transfer". This includes the treatment of address boundary, logical/physical transfer width, transfer acceleration, transfer direction control, and so on. Important here is the control of the last byte transfer on a bus with 16-bit transfer width and address modification control of the "second" pointer.

There exists a very specific case:

If the last data transferred on a 16-bit bus is a byte (e.g. initial byte count or address boundary) this has to be observed since the SAB 82257 continues data transfer after chaining

- with the new data address out of the new list element, and
- with the "second" pointer, which can be modified or not.

If no modification has to be performed on the "second" pointer, the original address as it is stored in the command block is used. This means in case of a last-byte transfer (first half of a word) on 16-bit bus, the next elementary block is started with a word transfer and not with a byte transfer (no acceleration possible).

If the "second" pointer has to be modified, the chained elementary block transfer starts with the new address after the last incrementation (or decrementation as programmed). Therefore in reverse transfer mode with 16-bit effective transfer width, the byte counts in the list elements have to be even.

After a complete block transfer specified by a list element, i.e. byte count has reached zero and last byte or word is transferred, list data chaining is executed if

- LC bit in channel command register (CCR) is set, and
- no other termination condition exists.

The execution flow during list data chaining is as follows:

1. Fetch byte count out of next list element, addressed by LPR,
2. increment LPR by 2,
3. check new byte count,
   - if BC = 0, stop chaining and go to termination processing,
   - if BC more than 0:
4. fetch new data address – low word – with LPR and load it
   - if SC = 1, into SPR,
   - if SC = 0, into DPR,
5. increment LPR by 2,
6. fetch new data address – high byte – with LPR and load it
   - if SC = 1, into SPR,
   - if SC = 0, into DPR,
7. increment LPR by 2,
8. continue data transfer with new parameters.

7.6.3 Linked List Chaining

Linked list data chaining is a modification of list chaining and can be enabled by the LLC flag (bit 9) in the type 1 channel command for each channel command block. In case of linked list chaining the list elements, defining data blocks which should be linked together, are not combined in one chaining list, but distributed in memory (see figure 75).

Each list element consists of five words and contains

- a 16-bit byte count,
- a 24-bit data pointer and
- a 24-bit link pointer (a pointer to the next list element).

Thus – as in case of list chaining – each list element (see figure 75) describes up to 64 Kbytes space (or segment) in memory.

A list element with zero byte count field indicates end of link.

If linked list chaining is enabled in a type 1 channel command, the source pointer (if source chaining, indicated by SC bit = 1) or the destination pointer (if destination chaining, indicated by SC bit = 0) is **replaced** by a link pointer to the list element (see figure 74). While doing data chaining the byte count from the channel command block is not used.

During the setup routine after channel start data pointer and 16-bit byte count are fetched from the list element pointed to by the link pointer in source (source chaining) or destination (destination chaining) pointer location, instead of the channel command block.

Then the list pointer to the next list element is fetched out of the current list element, i.e. the list pointer is not incremented as in list chaining.

Figure 72
Channel Command Block Configurations

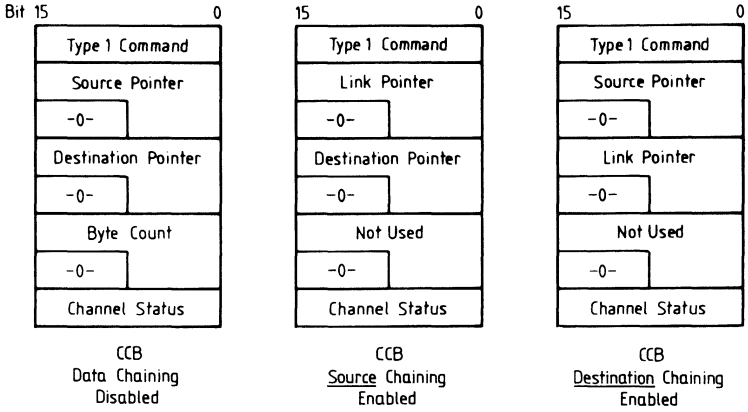
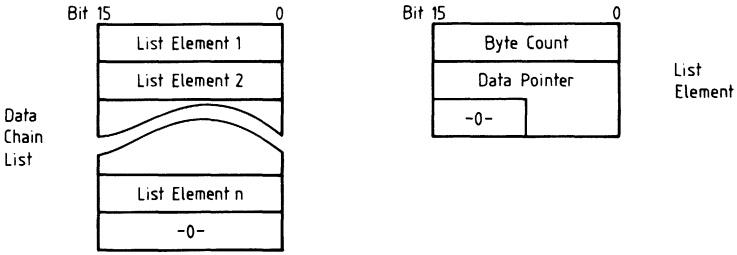


Figure 73
Data Chain List and List Element



The SAB 82257 processes each list element as an autonomous block transfer specified with

- data pointer (source or destination) in list element,
- "second" pointer (destination or source) first in command block and then – after first block transfer – in pointer register SPR or DPR),
- byte count in list element,
- unchanged channel command.

Such a block transfer specified by a list element (elementary block transfer) is part of the whole block transfer. Thereby this elementary block transfer is executed and controlled in exactly the same way as specified in section 7.5 "Data Transfer". This includes the treatment of logical/physical transfer width, transfer direction control, and so on (see also section 7.6.2).

After a complete block transfer specified by a list element, i.e. byte count has reached zero and last byte or word is transferred, linked list data chaining is executed if

- LLC bit in channel command register (CCR) is set, and
- no other termination condition exists.

The execution flow during linked list data chaining is as follows:

1. Fetch byte count of next list element, addressed by LPR,
2. increment LPR by 2,
3. check new byte count,
   - if BC = 0, stop chaining and go to termination processing,
   - if BC more than 0:
4. fetch new data address – low word – with LPR and load it
   - if SC = 1, into SPR,
   - if SC = 0, into DPR,
5. increment LPR by 2,
6. fetch new data address – high byte – with LPR and load it
   - if SC = 1, into SPR,
   - if SC = 0, into DPR,
7. increment LPR by 2,
8. fetch new list pointer – low word – with LPR and load it into temporary register,
9. increment LPR by 2,
10. fetch new list pointer – high byte – with LPR and load it into temporary register,
11. load the whole new list pointer out of temporary register into LPR (LPR points now to byte count of the next list element),
12. continue data transfer with new parameters.

Figure 74
Channel Command Block Configurations

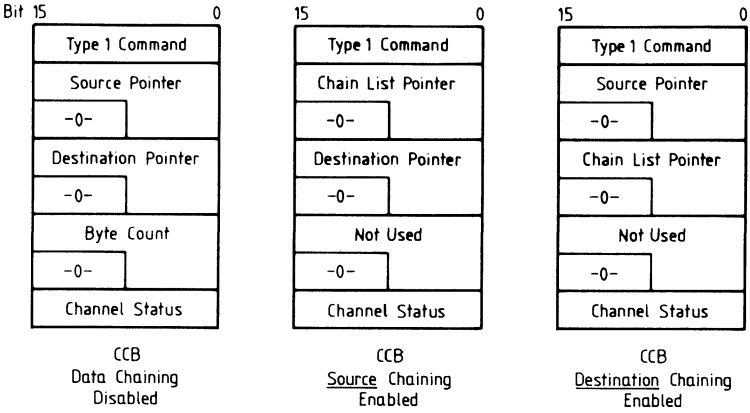
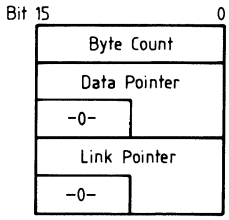


Figure 75
List Element



7.7 Termination of Data Transfer

7.7.1 Termination Conditions

A data transfer can be terminated due to one or two of the following indications and conditions (see figure 77):

1. **Byte Count End**
   Two options:
   Condition ①: Byte count is zero and data chaining is not enabled (standard termination condition).
   Condition ②: Data chaining is enabled and new fetched byte count is zero.
2. **External Termination**
   Condition ③: External termination via the channel's $\overline{EOD}$ line, if enabled by the EXT bit in CCR.

   As described in section 5.3.2 the end of DMA ($\overline{EOD}$) pins can be used to receive an asynchronous external terminate signal to terminate a running DMA transfer. For this purpose, the $\overline{EOD}$ lines have to be forced low for a minimum of 2 T-states by external circuitry (see figure 76).

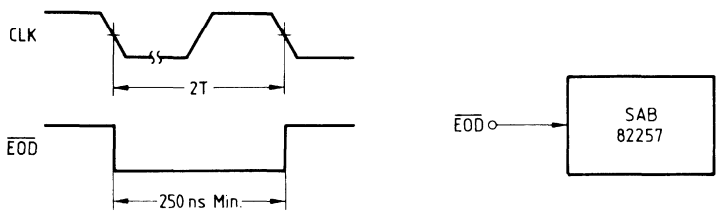
   An external termination is enabled by the SAB 82257 during the channel status "DMA in progress" as indicated in the general status register (GSR).

   Additionally, an external termination is only processed if it is enabled by the EXT bit in the type 1 channel command.

The data transfer can also be stopped by the CPU loading the general command register (GCR) with a STOP command. In that case the channel is not really terminated. Therefore this channel stop condition is not included in the following termination description.

The terminate conditions are signalled by the corresponding bit for each type in the channel status register (CSR), see next section.

Figure 76
End of DMA Signal ($\overline{EOD}$ Input)



7.7.2 Associated Control Register Bits

The following control register bits and parameters are used in order to control termination of DMA transfer due to

1. **Byte Count End**
   - Channel Status Register (CSR)
     - BC (bit 0): Transfer terminated because of exceeded byte count.
   - Channel Command Register (CCR)
     - LC (bit 8), LLC (bit 9), SC (bit 10) } Enable data chaining and its modifications.
     - EOD (bit 11): Enable synchronous End-of-DMA signal.
   - General Status Register (GSR)
     - DMSTn: DMA status.
2. **External Termination**
   - Channel Status Register (CSR)
     - ET (bit 1): External termination ($\overline{EOD}$ not programmed).
   - Channel Command Register (CCR)
     - EXT (bit 12): Enable external termination [the peripheral device uses the channel's $\overline{EOD}$ line (strobe pulse) for external termination signal].
   - General Status Register (GSR)
     - DMSTn: DMA status.

7.7.3 Initiation of Termination

Before starting the common termination processing, the termination indication (see section 7.7.1) has to be evaluated and – in some cases – specific operations have to be performed.

Transfer termination due to condition ①:
If $\overline{EOD}$ is enabled in CCR, a strobe pulse is transmitted on the channel's $\overline{EOD}$ line. This $\overline{EOD}$ pulse of two processor cycles' length is synchronous

- to the last data cycle of the synchronizing device in case of source or destination synchronization, or
- to the last destination data cycle in the case of no synchronization (free-running).

Therefore this strobe pulse is called synchronous $\overline{EOD}$ in contrast to the asynchronous $\overline{EOD}$ which is generated by a type 2 command.
After the last data cycle, processing of termination is started immediately.

Transfer termination due to condition ②:
The processing of termination is started immediately.

Transfer termination due to condition ③:
If external termination is enabled, an external $\overline{EOD}$ signal is expected during channel status "DMA in progress", as indicated by the DMST bits in GSR.

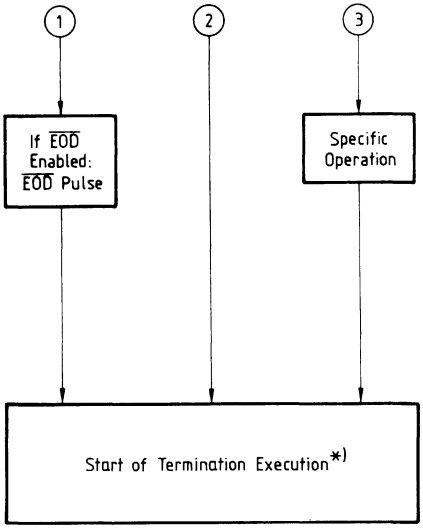
DMA is in progress from beginning after setup until start of termination.

If an external $\overline{EOD}$ is received, an already running bus cycle ($\overline{EOD}$ is synchronous to the bus cycle) is terminated. The following processing is the same as for an $\overline{EOD}$ which is not synchronous to a bus cycle (which needs a new prioritization).

It is assumed that during synchronized DMA transfer an external $\overline{EOD}$ is issued by the synchronizing device and that this device does not request another bus cycle.

Therefore
- in case of *destination synchronization* execution is started immediately;
- in case of *source synchronization* a byte or word which is already fetched from source and stored in the data assembly register (DAR) will still be transferred to destination before termination execution is started;
- in case of *no synchronization* (free-running) the DMA transfer is treated as source-synchronized data transfer.

Figure 77
Termination Conditions



*) see section 7.7.4.

7.7.4 Execution of Transfer Termination

The common termination routine is identical for all channels and transfer modes and it includes the following operations:

1. Change channel status – indicated by DMST bits in CCR – from "DMA in progress" into "organizational processing" (masking of following data requests).
2. Generation of termination status bits (BC, ET) in channel status register (CSRn). One or two of the termination status bits can be activated.
3. Store CSR into status word location of type 1 channel command block.
4. Increment channel command pointer in CPR by 16.
5. Without change of the channel's status bits in GSR, i.e. channel still remains in state "organizational processing":
   Fetch next type 1 or type 2 channel command with the new command pointer.
   Execution of command chaining.

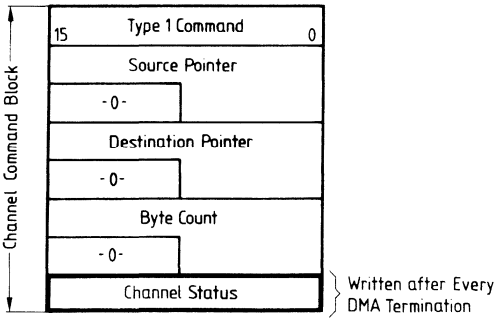
The execution of termination belongs to the last data transfer request or to external $\overline{EOD}$ request. Therefore no extra priority request control is performed for termination execution. With the execution of transfer termination the type 1 channel command (transfer command) processing is finished.

7.7.5 Saving Status on DMA Termination

On termination of a DMA transfer, the channel status is written into the command block for examination by the CPU. This channel status (see section 6.3.2 under Channel Status Register) contains information about.

- the termination condition(s),
- the occurrence of errors and
- halted state.

Figure 78
Status Information within CCB



Concurrent Channel Operation

8 Concurrent Channel Operation

8.1 Survey

The SAB 82257 allows concurrent processing of up to four DMA channels. Thereby the concurrent processing is controlled by internal and external requests and a priority system to handle them, since only one channel can actually be serviced at a time.

Each channel has a certain priority for its data transfers and organizational tasks. Data transfers or organizational tasks which are processed by a lower priority channel can be interrupted by requests for a higher priority channel after each bus cycle (only exception: unseparable bus cycles). If the highest priority routine is finished, the one of the waiting requests will succeed, which then has highest priority.

The channel's priority can either be **fixed** or **variable** depending on the order given in the general mode register (GMR).

A certain priority of a channel is valid not only for the data transfers of the channel but also for its organizational processing. The justification for this is illustrated by two examples:

1. A highest priority setup routine should be finished very quickly – only possible if it is not interruptable by a lower priority channel – since the fast peripheral may already have DMA requests.
2. The maximum speed of data chaining is only possible if it cannot be interrupted by a lower priority data transfer.

Consequently, the SAB 82257 offers a fully nested processing of internal and external requests. Thereby the priority logic must handle competing requests for the same channel or for different channels.

Concerning the priority system of the SAB 82257 also the arbitration of the local bus and – in remote mode – of the system bus should be considered in some way.

The SAB 82257's bus access time can be controlled by the general burst register (GBR) and the general delay register (GDR). Thus even the highest priority channel may be interrupted or delayed by the SAB 82257's bus arbitration unit. But within some organizational routines it is necessary to have unseparably chained transfers and to prevent the CPU from accessing the SAB 82257's control space in external memory.

For example:
The CPU must not change the source pointer between SAB 82257's two (or three in case of 8-bit bus) fetch cycles.

The SAB 82257 supports such a lock mechanism by means of a continuous bus request (HOLD signal), independent of whether the general burst counter is exceeded or the HLDA signal is cancelled. The SAB 82257 activates this "lock mechanism" during execution of the following routines *(unseparable bus cycles):*

- read 24-bit pointer and byte count from control space,
- word transfer on odd addresses, which is realized by two bus cycles where each transfer is a byte.

As a result, these unseparable chained transfers cannot be interrupted by higher priority requests.

The switching from one channel to another after a bus cycle does not affect the data transfer rate. It is important to notice that channel switching occurs only when both channels are ready to run.

8.2 Associated Control Register Bits

The relative priority among the four channels for DMA operation is determined by the following register bits:

- General Mode Register (GMR)
  - PRI (bits 9, 8): Channel priority.

PRI Bit 9	PRI Bit 8	Priority (0 = highest) of Channel 3	2	1	0	Comments
0	0	3	2	1	0	All channels have fixed priority (channel 0: highest priority).
0	1					Not valid
1	0					
1	1	R	R	R	R	All channels have rotating priority (i.e. all channels appear to have the same priority).

8.3 Control of Channel Priority

Each channel has a certain priority for its data transfers and organizational tasks according to the order given in the general mode register (GMR). As specified in the PRI field (bits 8, 9) of the general mode register (GMR) each channel can have either

- fixed priority or
- variable priority.

In case of **fixed** priority the channel's priority corresponds to its number. If channel 0 has fixed priority then it is always the highest priority channel and a fixed channel 3 has always lowest priority.

Thus, for example, if all channels have fixed priority, i.e. PRI is equal to 00, the channel priorities are as follows:

Channel	Priority (0 = highest)
Channel 0	0
Channel 1	1
Channel 2	2
Channel 3	3

In case of **variable** priority, the priorities change between all four channels (PRI = 11). Thereby the change of priority is organized in a modified "rotating" method. This means, that the priorities of variable channels are rotated after each bus cycle – which is not unseparably chained with the following bus cycle – of a variable channel. Thereby the channel with the last bus cycle gets the lowest priority of the variable channels.

The following table shall illustrate this organization:

Initial Priority (after reset)	Priority after Execution of Bus Cycle on Channel Number			
	0 →	1 →	2 →	3 →
Highest – 0	1	0	0	0
1	2	2	1	1
2	3	3	3	2
Lowest – 3	0	1	2	3

Note

- After RESET signal, the variable channel priorities are determined according to the channel numbers (fixed priorities).
- A new start of a variable priority channel does not cause any change in priority.
- Channel switching has absolutely no effect on the data transfer rate.
- It is important to remember that channel switching occurs only when the respective channels are ready to run.

8.4 Priority Control of Requests

The processing of internal or external requests is controlled by a fully nested priority system which includes all 4 channels.

Since more than one request can compete for the same channel, the requests must be prioritized also in relation to their types reflecting their relative importance.

The following table gives a survey.

Types of Channel Requests	Priority (0 = highest)
Channel STOP (command from CPU out of GCR).	0
External Asynchronous Termination Request (via $\overline{\text{EOD}}$).	1
Internal CONTINUE Request of Previously Interrupted Sequence.	2
Internal (without synchronization) Data Service Request.	3
External (with synchronization) Data Service Request.	3
Channel WAIT (idle).	4

The slave operations, where the SAB 82257 is addressed by the CPU, have highest priority of all activities. The SAB 82257 is immediately halted (remote mode) for taking over the slave part of the bus cycle transaction.

Thus a received STOP command is executed without any channel prioritization.

A START or CONTINUE command as well as all other types of channel requests will only be executed if the corresponding channel has highest priority of all requested channels.
Note that a channel START (setup) or CONTINUE command will be accepted only by a stopped channel and therefore no competition with other requests for the same channel exists.

A not interruptable block transfer on a lower priority channel is not supported.

Note
Data chaining and internal termination (byte count expired)need not be prioritized because these routines belong to the data service request processing (priority 3). Also error conditions (see chapter „Error Detection" are always connected with certain channel activities.

Interrupt Control

9 Interrupt Control

9.1 Survey

The SAB 82257 has 4 $\overline{EOD}$ pins, one for each channel, to interrupt the CPU and communicate with the system environment. Since the $\overline{EOD}$ pins are multiple function pins, their application is programmable. Thereby input and output functions have to be distinguished.

- As an **input**, the end of DMA ($\overline{EOD}$) pin can be used to receive an asynchronous external terminate signal to terminate a running DMA transfer if it is enabled with the EXT bit (bit 12) in the type 1 channel command.

  This function of the $\overline{EOD}$ signal is described in chapter "DMA Transfer".

- As an **output**, the $\overline{EOD}$ pin can be used to send out a pulse which interrupts the CPU (in this case the CPU can be interrupted by the peripheral as well as by the SAB 82257) and/or signals a specific status to the peripheral, e.g. transfer aborted or end of a block or send/receive next block, etc. Thus the SAB 82257 has one programmable external signal per channel.

In the following description, only the output functions of $\overline{EOD}$ pins are discussed. In this case two basic functions have to be distinguished:

- **INTOUT function** (interrupt output)
- **$\overline{EOD}$ function** (end of DMA)

9.2 Associated Control Register Bits

The **INTOUT** function is controlled with the following control register bits:

- General Mode Register (GMR)
  - MINTn (bit 10, bit 11, bit 12, bit 13): Mask the interrupts from channel n.
  - ENCI (bit 14): Common interrupt enabled/disabled.
- General Status Register (GSR)
  - INTn (bit 2, bit 6, bit 10, bit 14): Interrupt status.
- Channel Command Register (CCR) for type 2 commands
  - IT (bit 10): Generate interrupt.

The **$\overline{EOD}$** function is controlled with the following control register bits:

- Channel Command Register (CCR) for type 2 commands
  - ED (bit 11): Generate $\overline{EOD}$ pulse.

  Note
  Bit 14 and bit 15 in CCR equal 0 0.
- Channel Command Register (CCR) for type 1 commands
  - $\overline{EOD}$ (bit 11): Enable $\overline{EOD}$ output.

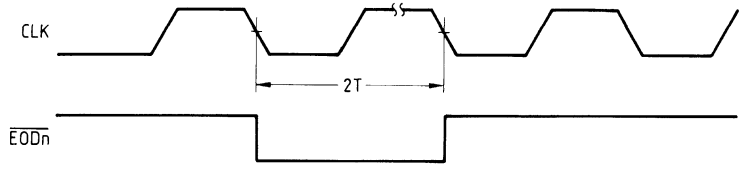
  Note
  Bit 14 and bit 15 in CCR do not equal 0 0.

9.3 Basic Signals

9.3.1 End of DMA Signal ($\overline{EOD}$)

$\overline{EOD}$ is a channel-specific, active low pulse of 2 T-states' length and is always enabled by software.
$\overline{EOD}$ signals can be generated synchronous to transfer as a result of type-1-command byte count termination (time-critical signalling) or asynchronous to transfer by a specific type 2 command as typically required for interrupting the CPU at the end of a command chain.

Figure 79
$\overline{EOD}$ Output Timing



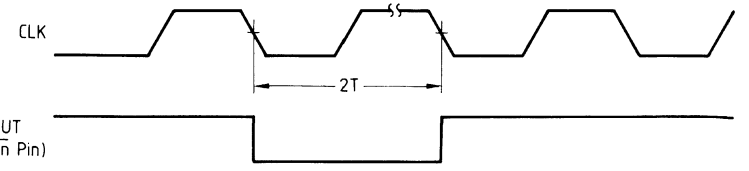
9.3.2 Interrupt Out Signal (INTOUT)

INTOUT can be hardware-generated, i.e. the signal is activated by the hardware when a fatal error occurs on any channel, or it can be enabled by software as a result of a type 2 command. The channel which is generating INTOUT is indicated in GSR with the channel's INT bit.
INTOUT remains active until all INT bits in GSR are reset by the CPU with the general command "clear interrupt".

The output processing of INTOUT depends on the ENCI bit and the four MINT bits in GMR:

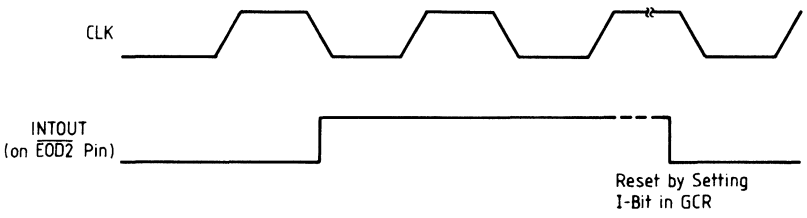
- If ENCI = 0, i.e. common interrupt not enabled, an internally produced INTOUT is issued on the specific $\overline{EOD}$ pin as active low $\overline{EOD}$ pulse, if the channel's interrupt mask bit (MINT) is zero.

Figure 80
Channel-Specific INTOUT Timing



- If ENCI = 1, i.e. common interrupt enabled, the INTOUTs of all four channels are issued on the $\overline{EOD2}$ line ($\overline{EOD}$ pin of channel 2), if the mask bits (MINT) of the generating channels are zero. In that case INTOUT is a static active high signal.

Figure 81
Common INTOUT Timing



Note
Common interrupt (ENCI = 1) should be used, if the $\overline{EODs}$ of the channel serve other purposes than CPU interruption, e.g. for communication between SAB 82257 and peripherals (see chapter "Communications Mechanism" sections 5.2 and 5.3).

9.4 Hardware-Generated Interrupt

A hardware-generated interrupt is an **INTOUT** (issued according to the ENCI bit) generated by the hardware itself, if a fatal error condition was detected. The INTOUT signal is only issued if it is not masked by the channel's MINT bit in the general mode register. The initiating fault condition is always a "command not valid" condition, detected during setup or command chaining (see chapter "Error Detection").

The interrupt sequence is as follows:

1. Set status bit FE (bit 6) in the channel's CSR.
2. Stop channel and indicate this through the channel's DMST bits in GSR.
3. Set the channel's INT bit in general status register (GSR).
4. If interrupt enabled by the channel's MINT bit in GMR: Issue INTOUT according to ENCI bit.
5. Reset INT bit, if a "clear interrupt" for that channel is written into the general command register by CPU.
6. Reset INTOUT, if no other interrupt is pending, i.e. no other INT bit is set in GSR.

Note
The channel's INT bit in general status register (GSR) is activated *independently* of the MINT bit in GMR.

9.5 Software-Generated Interrupt

Software generated interrupts are

- $\overline{\text{EOD}}$ signals generated as a result of
  - a type 1 command byte count termination and/or
  - a type 2 command,
- INTOUT signals generated as a result of a type 2 command.

Interrupts Generated as a Result of a Type 1 Command

Interrupts generated as a result of a type 1 channel command are always synchronous $\overline{\text{EOD}}$ signals controlled by byte count.

If byte count is exceeded and the EOD bit (bit 11) in the channel command register is set, an $\overline{\text{EOD}}$ signal is transmitted with the last transfer cycle. Thereby the $\overline{\text{EOD}}$ signal is issued with the last source cycle in case of source synchronization of DMA transfer, or together with the last destination cycle in case of destination synchronization or free-running DMA transfer.

Note

If data chaining is enabled, type 1 $\overline{\text{EOD}}$ signals should not be used for interrupts since multiple $\overline{\text{EOD}}$ signals are issued – at every exceeding byte count in list elements.

Interrupts Generated as a Result of a Type 2 Command

All type 2 channel commands allow to generate asynchronous $\overline{\text{EOD}}$ signals or INTOUT signals.

After command execution

- an $\overline{\text{EOD}}$ signal is activated, if the ED bit in CCR equals one, and
- an INTOUT signal is activated, if the IT bit in CCR equals one and if interrupt is not masked by the channel's MINT bit in general mode register (GMR).

Note

The channel's INT bit in general status register (GSR) is activated *independently* of the MINT bit.

9.6 Summary

In case of an $\overline{\text{EOD}}$ from a DMA channel, the signal itself determines the appropriate channel number (the $\overline{\text{EOD}}$ signal is issued on the channel-specific $\overline{\text{EOD}}$ pin).

In case of INTOUT from a DMA channel the issuing channel is indicated in the general status register (GSR) by means of the INT bit.

Additional information available to the CPU is contained in

- the **General Status Register** (GSR)
  In the GSR the DMST bits indicate whether the channel is stopped or not, thus determining a stop interrupt or an intermediate interrupt.
- the **Command Pointer Register** (CPR)
  In case of a stop interrupt (indicated by DMST bits in GSR) the CPR points to the channel command executed last.
- the **Channel Status Register** (CSR)
  In the CSR the termination condition or an error detection is indicated.

Additional context information in registers available to the CPU is for example

- last byte count,
- source address or
- destination address.

Error Detection

10 Error Detection

The SAB 82257 performs limited error checking for type 1 command errors (fatal errors) during "setup and "command chaining". Thereby the checked fatal error conditions always are "command not valid" conditions. If such a fatal error condition is detected, the affected channel processing is interrupted and the fatal error bit FE (bit 6) in the channel's status register CSR is set.

10.1 Fatal Errors

Fatal errors are detected during decoding of type 1 channel commands in combination with the general mode register (GMR). Thereby four conditions are used for detection, and allowed combinations of them lead to four different transfer executions, such as two-cycle DMA or single-cycle DMA. All other combinations (besides the four allowed ones) generate a fatal error.

10.1.1 Error Conditions

The four source conditions, which are decoded, are the following:

- single-cycle transfer (from GMR)
- transfer without destination pointer (from CCR)
- transfer without source pointer (from CCR)
- synchronization error

The following table gives the valid combinations of these four conditions in accordance with transfer operation:

Intended Transfer Operation	Conditions[1]			
	Single Cycle	No Dst. Pointer	No Source Pointer	Sync. Error[2]
Two-Cycle DMA Transfer	0	0	0	–
Transfer without Destination Pointer	0	1	0	0
Transfer without Source Pointer	0	0	1	0
Single-Cycle DMA Transfer	1	0	0	0
All Other Combinations Lead to Fatal Errors!				

[1] 1: TRUE
0: FALSE

[2] The synchronization error is predecoded and activated in the following cases:
- Single-Cycle DMA combined with free-running synchronization mode
- Transfer without destination pointer combined with destination synchronization
- Transfer without source pointer combined with source synchronization

10.1.2 Reaction on Fatal Errors

If a fatal error condition is detected on a channel, the following steps are performed (see also section 9.4 "Hardware-Generated Interrupt"):

1. Set error bit FE (bit 6) in the channel's status register (CSR).
2. Stop channel and indicate this through the channel's DMST bits in general status register (GSR).
3. Set channel-specific INT bit in the general status register (GSR). The channel's INT bit is activated *independently* of the MINT bit in GMR.
4. Send interrupt (INTOUT) if not masked by channel's MINT bit in general mode register (GMR) according to ENCI bit in GMR.

For error investigation the CPU should react on error INTOUT with:

1. Read GSR (what channel? channel stopped?).
2. Read CSR (error?).
3. Read CPR and investigate channel command (type 1 command).

10.2 Non-Fatal Errors

Non-fatal errors, are errors, not recognized by the SAB 82257 and therefore not indicated in the affected channel's status register CSR, which *do not* prevent the SAB 82257 DMA controller from working as it should.
Most of these errors (improper commands) result in a predefined default action.
The following gives some typical examples of non-fatal errors:

Error	Action
● Remote mode coincides with 186 mode	RM bit is not inhibited (but read/write pins are also used as outputs).
● Physical bus width is programmed to 8 bit and logical bus width to 16 bit.	Logical bus width will be set to 8 bit.
● List chaining and linked list chaining are enabled.	Execution of linked list data chaining.
● Data chaining and last data transfer was a byte transfer to or from 16-bit device (odd byte count) during reverse transfer.	Data transfer is continued after chaining.

Operating Instructions

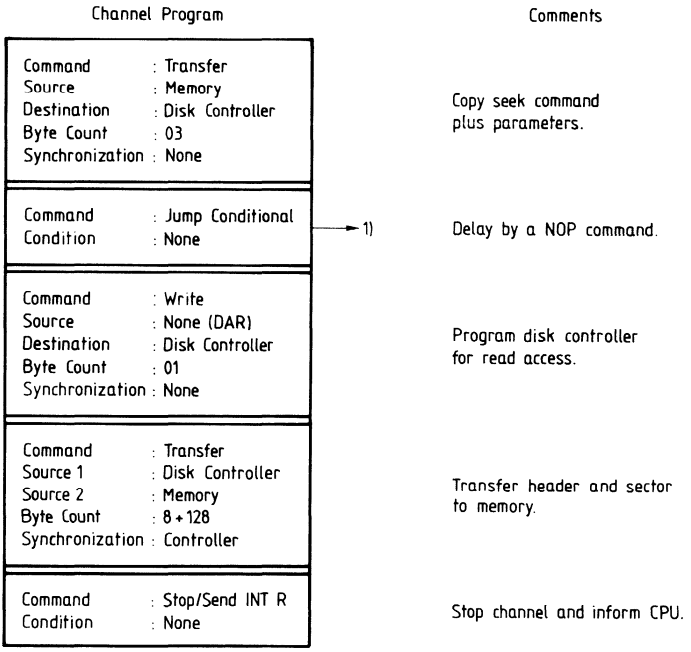
11 Operating Instructions

11.1 Channel Programming Examples

A sequence of operations can be programmed for each channel using command chaining. This can be executed without CPU intervention and is achieved by using a mixture of type 1 and type 2 commands. Although the commands are easy to use, powerful programs can be generated.

This section gives brief examples illustrating how to program such a command chain. The examples can easily be adapted to various applications. In addition, the second example adds a complete channel program, written in ASM86, that contains all codes for the SAB 82257 as well as for the associated processor which is necessary to run the channel. The examples show how standard features of ASM86 are an aid to program the SAB 82257 in a structured manner.

Figure 82
Channel Program for a Read from Disk Operation



[1] The target location for the jump command is arbitrary, as the jump is never executed.

11.1.1 Example 1: Read from Disk

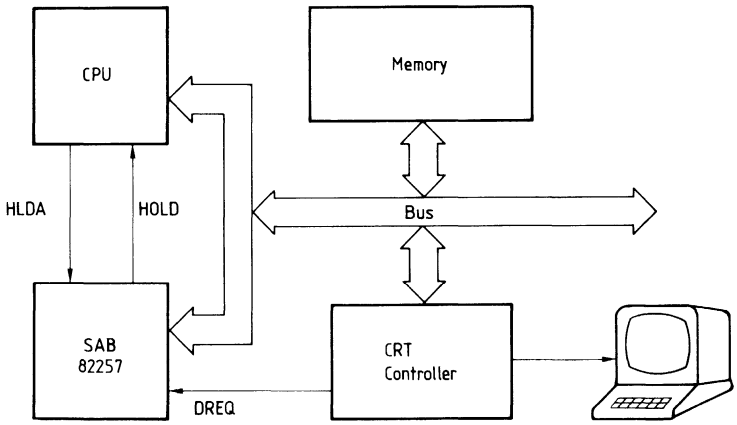
This example introduces a channel program that reads a specific sector from a disk device. To initiate the access, the SAB 82257 copies a seek command with some parameters from an arbitrary memory location to the disk controller. These parameters have to be provided by the CPU in advance. A NOP command (or several of them, if required) allows the disk controller to get ready. A read command starts the actual transfer of data. This command is treated as a literal (no source). The transfer itself is activated by the disk controller (source synchronization). After the transfer has been completed, the channel is stopped and the CPU is informed by an interrupt (common interrupt on $\overline{EOD2}$ pin).

Note
The header could easily be separated from the sector data using destination data chaining.

11.1.2 Example 2: CRT Refresh

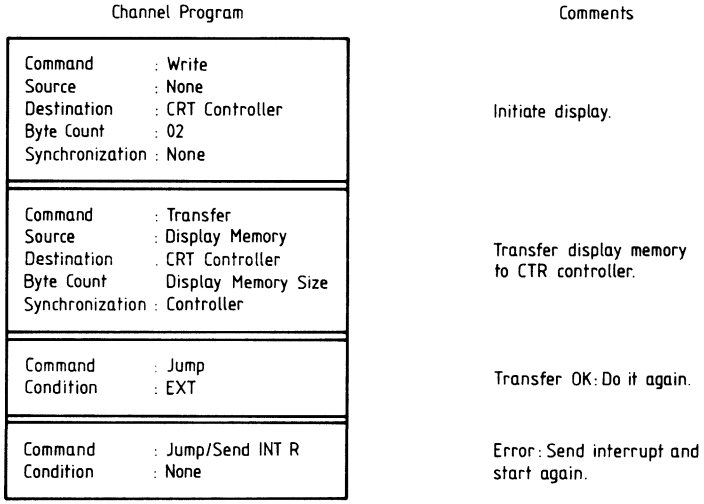
In this example the SAB 82257 independently performs a CRT refresh operation. This operation includes
- start display on CRT,
- automatically refresh the display from memory,
- inform the CPU about synchronization errors.

Figure 83
CRT Refresh



In this case, a synchronization error will occur if, for some reason, the CRT controller does not demand a new screen (start of display memory) after transfer of the complete display memory. The vertical retrace signal could then be used to activate the appropriate $\overline{EOD}$ input. In case of an error, the operation could be stopped. The channel program in this example sends an interrupt to the CPU and automatically resynchronizes the refresh by simply jumping to the start.

Figure 84
Channel Program for a CRT Refresh Operation



ASM86 allows defining of structures for both kinds of commands: type 1 and type 2. Figure 85 shows a possible realization of these structures.

Figure 85
ASM86 Structures for Channel Command Blocks (CCB)

```
TYPE1_BLOCK  STRUC          ;Transfer Command Block
;---------------------------;-----------------------------
CMD_1        DW    ?        ;Type1 Command
SPTR_L       DW    ?        ;Source-Pointer
SPTR_H       DW    ?        ;dto.
DPTR_L       DW    ?        ;Destination-Pointer
DPTR_H       DW    ?        ;dto.
BCNT_L       DW    ?        ;Byte-Count
BCNT_H       DW    ?        ;dto.
STATUS       DW   00        ;Channel Status Field
TYPE1_BLOCK  ENDS

TYPE2_BLOCK  STRUC          ;Organizational Comand Block
;---------------------------;-----------------------------
CMD_2        DW    ?        ;Type2 Command
REF_L        DW    ?        ;Jump Pointer/Displacement
REF_H        DW    ?        ;Jump Pointer/0000
TYPE2_BLOCK  ENDS
```

Once these structures are defined, a CCB can be specified by simply referencing the structure name and filling in the required parameters:

```
MOV  AX,TRANSFER.STATUS     ;Read Channel Status from CCB
```

The CPU can access the CCBs by addressing the appropriate structure elements:

```
TRANSFER      TYPE1_BLOCK   <0C08FH,120AH,00,8CC3H,00,02,00>
```

The following pages list the complete channel program (written in ASM86) for the example 2. The code of the channel program for the SAB 82257 is listed as well as the code for the CPU necessary to initialize the channel program (location references) and the SAB 82257 (registers).

Figure 86
ASM86 CRT Refresh Program

```
8086/87/88/186 MACRO ASSEMBLER   SAB82257_CHANNELPROGRAM_EXAMPLE                14:00:56  03/18/86  PAGE   1


INDX-541 (V3.0) 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE SAB82257_CHANNELPROGRAM_EXAMPLE
OBJECT MODULE PLACED IN A:EX257.OBJ
ASSEMBLER INVOKED BY:  ASM86 A:EX257.SRC PAGEWIDTH(114) DEBUG

LOC  OBJ                  LINE     SOURCE

                             1     ;+------------------------------------------------------------------------+
                             2     ;|                                                                        |
                             3     ;|      Example program for CRT refresh operation using the SAB 82257.    |
                             4     ;|      Hans Sulzer , Technical Support Group , Microcomputer Components  |
                             5     ;|                          Siemens AG , Munich , 1986                    |
                             6     ;|                                                                        |
                             7     ;|- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -|
                             8     ;|                                                                        |
                             9     ;| The channel program consists of 4 channel command blocks (CCB):        |
                            10     ;|                                                                        |
                            11     ;|  CCB_START      : issues a start-command to the CRT controller         |
                            12     ;|  CCB_REFRESH    : transfers the display memory to the CRT controller   |
                            13     ;|                   (actual refresh operation)                           |
                            14     ;|  CCB_JUMP_ON_EXT : normal loop back to refresh command block           |
                            15     ;|  CCB_JUMP_INTR  : error loop back sending (common) interrupt to CPU    |
                            16     ;|                                                                        |
                            17     ;|- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -|
                            18     ;|                                                                        |
                            19     ;|  1. Declarations of registers, CCB structures, etc.                    |
                            20     ;|  2. Channel program                                                    |
                            21     ;|  3. CPU updates the channel program                                    |
                            22     ;|  4. CPU initializes the SAB 82257 and starts refreshing                |
                            23     ;|  5. Conversion of segment/offset pointer to 24-bit real address        |
                            24     ;|                                                                        |
                            25     ;+------------------------------------------------------------------------+
                            26
                            27     NAME SAB82257_CHANNELPROGRAM_EXAMPLE
                            28
                            29
                            30     ;+------------------------------------------------------------------------+
                            31     ;| Assigning names to the SAB 82257 registers.                            |
                            32     ;+------------------------------------------------------------------------+
1000                        33     SAB82257        EQU  1000H      ;Base address, depends on decoding
                            34                                     ;SAB82257 mapped into I/O space
1000                        35     GCR             EQU  SAB82257+00H ;General command register
1008                        36     GMR             EQU  SAB82257+08H ;General mode register
100A                        37     GBR             EQU  SAB82257+0AH ;General burst register
100C                        38     GDR             EQU  SAB82257+0CH ;General delay register
1020                        39     CPR_0           EQU  SAB82257+20H ;Command pointer register channel 0
2000                        40     CRT_CNTRL       EQU  2000H      ;Input register CRT controller
                            41
                            42  +1 $ EJECT
```

```
8086/87/88/186 MACRO ASSEMBLER    SAB82257_CHANNELPROGRAM_EXAMPLE                14:00:56  03/18/86  PAGE    2


LOC  OBJ                  LINE     SOURCE

                            43     ;+------------------------------------------------------------------------------+
                            44     ;!  D u m m i e s                                                               !
                            45     ;+------------------------------------------------------------------------------+
  0000                      46     LOC_REF       EQU  0000H            ;Location reference,
                            47                                         ;unknown at assembly time
  07D0                      48     DIS_MEM_SIZE  EQU  25 * 80          ;Size of the display memory
  0050                      49     ST_LINE_SIZE  EQU   1 * 80          ;Size of status line
  0001                      50     START_CMD     EQU  0001H            ;Start command for CRT controller
                            51
                            52     ;+------------------------------------------------------------------------------+
                            53     ;!  D a t a - A r e a                                                           !
                            54     ;+------------------------------------------------------------------------------+
----                        55     WORKING_SPACE SEGMENT
                            56
0000 (2000                  57     DISPLAY_MEMORY DB DIS_MEM_SIZE DUP(?);Copy of screen
     ??
     )
07D0 (80                    58     STATUS_LINE    DB ST_LINE_SIZE DUP(?);Copy of status line
     ??
     )
----                        59     WORKING_SPACE ENDS
                            60
                            61     ;+------------------------------------------------------------------------------+
                            62     ;!  Definition of the channel command block templates                          !
                            63     ;+------------------------------------------------------------------------------+
                            64
----                        65     TYPE1_BLOCK  STRUC          ;Transfer Command Block
                            66     ;--------------------------;------------------------------
0000                        67     CMD_1        DW   ?         ;Type1 Command
0002                        68     SPTR         DD   ?         ;Source-Pointer
0006                        69     DPTR         DD   ?         ;Destination-Pointer
000A                        70     BCNT         DD   ?         ;Byte-Count
000E                        71     STATUS       DW  00         ;Channel Status Field
----                        72     TYPE1_BLOCK  ENDS
                            73
----                        74     TYPE2_BLOCK  STRUC          ;Organizational Comand Block
                            75     ;--------------------------;------------------------------
0000                        76     CMD_2        DW   ?         ;Type2 Command
0002                        77     REF_L        DW   ?         ;Jump Pointer/Displacement
0004                        78     REF_H        DW   ?         ;Jump Pointer/0000
----                        79     TYPE2_BLOCK  ENDS
                            80
                            81 +1  $ EJECT
```

```
8086/87/88/186 MACRO ASSEMBLER    SAB82257_CHANNELPROGRAM_EXAMPLE                14:00:56  03/18/86  PAGE    3


LOC  OBJ                  LINE     SOURCE

                            82     ;+------------------------------------------------------------------------------+
                            83     ;|  Actually defining the channel program for this example                      |
                            84     ;+------------------------------------------------------------------------------+
----                        85     CRT_REFRESH   SEGMENT
                            86
0000 8FC0                   87     CCB_START     TYPE1_BLOCK  <0C08FH,START_CMD,CRT_CNTRL,02>
0002 01000000
0006 00200000
000A 02000000
000E 0000
                            88                                ;Free running transfer without source,
                            89                                ;bus widths: 16 bit
                            90     ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0010 8D95                   91     CCB_REFRESH   TYPE1_BLOCK  <958DH,LOC_REF,CRT_CNTRL,00>
0012 00000000
0016 00200000
001A 00000000
001E 0000
                            92                                ;Destination synchronized transfer from memory
                            93                                ;with external termination enabled
                            94                                ;using source data list chaining
                            95     ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0020 0220                   96     CCB_JUMP_EXT  TYPE2_BLOCK  <2002H,(CCB_REFRESH-CCB_JUMP_EXT),00>
0022 F0FF
0024 0000
                            97                                ;Jump relative to CCB_REFRESH on EXT
                            98     ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0026 0324                   99     CCB_JUMP_INTR TYPE2_BLOCK  <2403H,(CCB_REFRESH-CCB_JUMP_INTR),00>
0028 EAFF
002A 0000
                           100                                ;Jump relative to CCB_REFRESH and issue Interrupt
                           101     ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
002C 0000                  102     CCB_STOP      TYPE2_BLOCK  <00,00,00>
002E 0000
0030 0000
                           103                                ;Terminal STOP command
                           104
                           105 +1  $ EJECT
```

```
8086/87/88/186 MACRO ASSEMBLER    SAB82257_CHANNELPROGRAM_EXAMPLE                14:00:56  03/18/86  PAGE    4


LOC  OBJ                  LINE     SOURCE

                           106     ;--------------------------;Definition of data blocks to be transferred
0032 0007                  107     DATA_CHAIN_LIST  DW DIS_MEM_SIZE     ;1st element
0034 00000000              108                      DD LOC_REF
0038 5000                  109                      DW ST_LINE_SIZE     ;2nd element
003A 00000000              110                      DD LOC_REF
003E 0000                  111                      DW 00               ;End of list
                           112
----                       113     CRT_REFRESH   ENDS
                           114     ;--------------------------------------------------------------------------------
                           115
----                       116     CPU_PROGRAM   SEGMENT
                           117     ASSUME CS:CPU_PROGRAM,ES:CRT_REFRESH
                           118
                           119     ;+------------------------------------------------------------------------------+
                           120     ;|  1.: Update pointers within data chain list (actual data addresses)          |
                           121     ;+------------------------------------------------------------------------------+
0000 B8----          R     122     MOV  AX,SEG DISPLAY_MEMORY                  ;1st element is screen
0003 50                    123     PUSH AX
0004 B80000                124     MOV  AX,OFFSET DISPLAY_MEMORY
0007 50                    125     PUSH AX
0008 E86500                126     CALL PHYSICAL_ADDRESS
000B 58                    127     POP  AX
000C 26A33400              128     MOV  ES:DATA_CHAIN_LIST+02,AX
0010 58                    129     POP  AX
0011 26A33600              130     MOV  ES:DATA_CHAIN_LIST+04,AX
                           131
0015 B8----          R     132     MOV  AX,SEG STATUS_LINE                     ;2nd element is status line
0018 50                    133     PUSH AX
0019 B80007                134     MOV  AX,OFFSET STATUS_LINE
001C 50                    135     PUSH AX
001D E85000                136     CALL PHYSICAL_ADDRESS
0020 58                    137     POP  AX
0021 26A33A00              138     MOV  ES:DATA_CHAIN_LIST+08,AX
0025 58                    139     POP  AX
0026 26A33C00              140     MOV  ES:DATA_CHAIN_LIST+10,AX
                           141
                           142 +1  $ EJECT
```

```
8086/87/88/186 MACRO ASSEMBLER    SAB82257_CHANNELPROGRAM_EXAMPLE                14:00:56  03/18/86  PAGE    5

LOC  OBJ                  LINE     SOURCE

                          143      ;+-------------------------------------------------------------------------+
                          144      ;|  2.: Update pointers within channel program (data chain list address)   |
                          145      ;+-------------------------------------------------------------------------+
002A B8----         R     146      MOV  AX,SEG DATA_CHAIN_LIST                ;Source pointer replaced by
002D 50                   147      PUSH AX                                    ;link pointer
002E B83200               148      MOV  AX,OFFSET DATA_CHAIN_LIST
0031 50                   149      PUSH AX
0032 E83B00               150      CALL PHYSICAL_ADDRESS
0035 58                   151      POP  AX
0036 26A31200             152      MOV  ES:WORD PTR CCB_REFRESH.SPTR+00,AX
003A 58                   153      POP  AX
003B 26A31400             154      MOV  ES:WORD PTR CCB_REFRESH.SPTR+02,AX
                          155
                          156      ;+-------------------------------------------------------------------------+
                          157      ;|  3.: Initialize SAB 82257 registers                                     |
                          158      ;+-------------------------------------------------------------------------+
003F BA0810               159      MOV  DX,GMR                ;System configuration:
0042 B8A378               160      MOV  AX,0111100010100011B  ;SAB 82257 in local mode with 16-bit buses,
0045 EF                   161      OUT  DX,AX                 ;fixed priority, channels 1,3:single cycle transfer,
                          162                                 ;common INTOUT,  channels 0,2:double cycle transfer
0046 BA0A10               163      MOV  DX,GBR                ;Program bus load to a maximum
0049 B88000               164      MOV  AX,80H                ;of 66 percent
004C EF                   165      OUT  DX,AX
004D BA0C10               166      MOV  DX,GDR
0050 B84000               167      MOV  AX,40H
0053 EF                   168      OUT  DX,AX
                          169
                          170      ;+-------------------------------------------------------------------------+
                          171      ;|  4.: Specify channel program                                            |
                          172      ;+-------------------------------------------------------------------------+
0054 B8----         R     173      MOV  AX,SEG CCB_START
0057 50                   174      PUSH AX
0058 B80000               175      MOV  AX,OFFSET CCB_START
005B 50                   176      PUSH AX
005C E81100               177      CALL PHYSICAL_ADDRESS
005F BA2010               178      MOV  DX,CPR_0              ;Load low word of command pointer
0062 58                   179      POP  AX
0063 EF                   180      OUT  DX,AX
0064 42                   181      INC  DX
0065 42                   182      INC  DX
0066 58                   183      POP  AX
0067 EF                   184      OUT  DX,AX                 ;Load high word of command pointer
                          185
                          186      ;+-------------------------------------------------------------------------+
                          187      ;|  5.: Start DMA on channel 0                                             |
                          188      ;+-------------------------------------------------------------------------+
0068 BA0010               189      MOV  DX,GCR                ;Start channel 0 and clear interrupts
006B B81A00               190      MOV  AX,1AH
006E EF                   191      OUT  DX,AX
                          192
                          193 +1   $ EJECT
```

```
8086/87/88/186 MACRO ASSEMBLER    SAB82257_CHANNELPROGRAM_EXAMPLE                14:00:56  03/18/86  PAGE    6


LOC  OBJ                  LINE     SOURCE

                          194      ;+------------------------------------------------------------------------------+
                          195      ;|  Start execution of another CPU task here                                    |
                          196      ;+------------------------------------------------------------------------------+
006F F4                   197      HLT                     ;Dummy, as no other task is present
                          198
                          199      ;+------------------------------------------------------------------------------+
                          200      ;|  Conversion of segment/offset pointer to 24-bit physical address             |
                          201      ;+------------------------------------------------------------------------------+
0070                      202      PHYSICAL_ADDRESS  PROC NEAR
0070 55                   203                PUSH    BP              ;Save current registers
0071 50                   204                PUSH    AX
0072 53                   205                PUSH    BX
0073 51                   206                PUSH    CX
0074 8BEC                 207                MOV     BP,SP           ;Access to parameters on stack
0076 8B5E0C               208                MOV     BX,SS:[BP]+12 ;Segment part of original pointer
0079 8B4E0A               209                MOV     CX,SS:[BP]+10 ;Offset part of original pointer
007C 8BC3                 210                MOV     AX,BX
007E D1E0                 211                SHL     AX,1            ;Multiply segment by 16
0080 D1E0                 212                SHL     AX,1
0082 D1E0                 213                SHL     AX,1
0084 D1E0                 214                SHL     AX,1
0086 03C8                 215                ADD     CX,AX           ;Add base low word to offset
0088 33C0                 216                XOR     AX,AX
008A 8AC7                 217                MOV     AL,BH
008C D0E8                 218                SHR     AL,1            ;Adjust segment remainder
008E D0E8                 219                SHR     AL,1
0090 D0E8                 220                SHR     AL,1
0092 D0E8                 221                SHR     AL,1
0094 150000               222                ADC     AX,00           ;Add carry to base high byte
0097 894E0A               223                MOV     SS:[BP]+10,CX ;Put back low word of address
009A 89460C               224                MOV     SS:[BP]+12,AX ;Put back high word of address
009D 59                   225                POP     CX              ;Restore current registers
009E 5B                   226                POP     BX
009F 58                   227                POP     AX
00A0 5D                   228                POP     BP
00A1 C3                   229                RET                     ;Conversion complete
                          230      PHYSICAL_ADDRESS  ENDP
                          231
----                      232      CPU_PROGRAM   ENDS
                          233      END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

11.2 Operating in Local Mode

In local mode the SAB 82257 is connected to the local bus together with a processor (SAB 8086/8088/80186/80188/80286). In this mode all the buses and interface components are shared by the SAB 82257 and the processor. Only one of the two (or more) may use the local, and hence the memory and I/O bus, at a time. The arbitration between the different bus masters on the local bus is done using HOLD/HLDA lines (SAB 82257 working with SAB 80186/80188/80286 processors) or the $\overline{RQ}/\overline{GT}$ line (SAB 82257 working with SAB 8086/8088 processors).

To distinguish between I/O bus accesses and memory bus accesses, the M/IO parameter in the channel command is used. The M/IO parameter controls these accesses for source and destination separately.

To allow the SAB 82257 to transfer channel programs from memory space to I/O space for later execution starting there, it is also necessary to switch dynamically from memory bus to I/O bus for organizational accesses to the channel programs.

Therefore the MEMBUS/IOBUS information is included in the general START channel command. Note that each channel can have the organizational blocks in memory or I/O space.

The memory bus need not have the same physical bus width as the I/O bus. In an 8-bit Multibus system, the I/O bus can have 16-bit width, or – more often – in a 16-bit Multibus system the I/O bus can be 8 bit. In the general mode register (GMR), two bits indicate the physical bus widths:

- MEMBUS (bit 0) and
- IOBUS (bit 1).

MEMBUS is used for memory bus accesses and IOBUS for I/O bus accesses. Thereby the SAB 82257 compares the logical bus width (indicated in the type 1 channel command for DMA transfers) with the physical width of that bus which is used.

Example

- If the physical bus width is 8 bit and the logical bus width is 16 bit, byte transfers will be performed automatically.
- If the physical bus width is 16 bit and the logical bus width is 8 bit, byte transfers are performed in that data bus half which is addressed by the least significant address bit.

Logical bus width pertains to DMA transfers only. All organizational transfers are performed in bytes or words depending on the physical bus width as indicated by MEMBUS and IOBUS.

MEMBUS always defines the physical bus width of the bus the SAB 82257 shares with the processor. Thus the width of the communication between CPU and SAB 82257 does not depend on the physical width of the IOBUS.

11.3 Operating in Remote Mode

In remote mode the SAB 82257 is interfaced to the CPU via the system bus. The SAB 82257 is the only master of the local/resident bus. The CPU can communicate with the SAB 82257 via the slave interface connected to the system bus.

When the SAB 82257 is in remote mode, the HOLD/HLDA sequence is redefined as follows:

- For access to the resident bus the SAB 82257 does not generate HOLD and starts the access without receiving HLDA.
- For all accesses to the system bus the SAB 82257 generates HOLD before getting on to the local bus. Only when receiving HLDA, it starts the bus cycle and occupies its local bus. This ensures a deadlock-free arbitration of the local bus.

If the Multibus is used as system bus, the SAB 82289 should be used as bus arbiter. Then HOLD from the SAB 82257 is directly connected (inverted) with the $\overline{S0}$ input of the SAB 82289, thus initiating the Multibus arbitration. After Multibus arbitration, the SAB 82257 uses the (inverted) $\overline{AEN}$ signal from the SAB 82289 as HLDA input. While waiting for HLDA, the SAB 82257 can always be accessed as a slave by the CPU.

The SAB 82257 has to distinguish between resident bus accesses and system bus accesses. This is controlled by the M/IO parameter in the channel command for both source and destination separately. In remote mode therefore memory accesses are treated as system bus accesses and I/O accesses are treated as resident bus accesses. Externally, the HOLD signal can be used for distinction between accesses to system or resident space.

In remote mode the SAB 82257 has also to decide whether organizational accesses should be performed on the resident bus or on the system bus.

To allow the SAB 82257 to transfer channel programs from system space to resident space for later execution from there, it is necessary to switch dynamically from system bus to resident bus for organizational accesses. Therefore the SYSBUS/RESBUS information is included in the general START channel command. Each channel can have the organizational blocks in system or resident space, so the SYSBUS/RESBUS distinction made for remote mode is used the same way as the M/$\overline{IO}$ distinction in local mode.

The resident bus need not have the same physical bus width as the system bus. In an 8-bit Multibus system the resident bus can have 16 bit width, or – more often – in a 16-bit Multibus system the resident bus can be 8 bit. In the general mode register (GMR), two bits indicate the physical bus widths:

- SYSBUS (bit 0) and
- RESBUS (bit 1).

SYSBUS is used for all system bus accesses and RESBUS for resident bus accesses. This also means that the SAB 82257 compares the logical bus width (indicated in the type 1 channel command for DMA transfers) with the physical width of the bus used.
If the physical width is 8 bits and the logical bus width is 16 bits, byte transfers are performed automatically. If the physical width is 16 bits and the logical bus width is 8 bits, byte transfers are performed on that data bus half which is addressed by the least significant address bit (same as in local mode, but there SYSBUS is used for memory bus width and RESBUS for I/O bus width indication).

Logical bus width pertains to DMA transfers only. All organizational transfers are performed in bytes or words depending on the physical bus width as indicated by SYSBUS and RESBUS. In case of an 8-bit physical bus the SAB 82257 uses the lower half of the data bus.

In remote mode as well as in local mode, SYSBUS (MEMBUS) always defines the physical bus width of the bus the SAB 82257 shares with the CPU. Thus the width of communication between CPU and SAB 82257 is independent of the physical width of the resident bus.

The CPU is only allowed to occupy the local bus if this bus is not accessed by the SAB 82257. Therefore the CPU has to wait until the current local bus cycle is terminated. In remote mode the chip select input $\overline{CS}$ serves as attention signal which causes the SAB 82257 to release the local bus after the current bus cycle. The SAB 82257 indicates the released state by means of the BREL signal. (The BREL signal is used to start the local bus cycle for the CPU).

Special logic is required for systems which need a CPU access to other devices on resident bus. In such systems, the CPU needs not only access to the SAB 82257 itself but also to its resident bus.
Such a CPU access to the resident bus is organized with the SAB 82257 operating as resident bus arbiter:

- The chip select input $\overline{CS}$ of SAB 82257 is used like a HOLD request input, and
- the BREL output exactly functions as a HOLD output.

The BREL signal will be active only as long as $\overline{CS}$ is active. Therefore $\overline{CS}$ should only be of minimum length, for example the length of a Multibus read or write command.

The bus release signal BREL uses the SAB 82257's M/$\overline{IO}$ pin since the M/$\overline{IO}$ function is not needed in remote mode. After reset, this pin is in tristate and an external pullup resistor forces it high. Although reset forces the SAB 82257 into local mode, the CPU can now communicate with the SAB 82257 because

- BREL switches the latches and transceivers of the Multibus interface to a direction from Multibus to the SAB 82257's local bus and
- the SAB 82257 does not execute any bus cycle by itself until a channel is started.

Thus the CPU can load the SAB 82257's general registers beginning with the general mode register (GMR). If the RM bit is set in GMR, the SAB 82257 switches to remote mode and the M/$\overline{IO}$ pin gets the BREL function.

11.4 Connecting Peripherals

Each of the 4 independent channels of the SAB 82257 has three dedicated control lines (see figure 86):

- $\overline{EOD}$ (end of DMA) line, a flexible bidirectional line used as an external terminate line to stop running DMA or as an interrupt signal to the CPU or a peripheral announcing the end of a DMA operation, and the conventional
- DREQ (DMA request) and
- $\overline{DACK}$ (DMA acknowledge) lines used for synchronized DMA transfer.

The DREQ and $\overline{DACK}$ signals are directly compatible with most peripheral controller circuits.

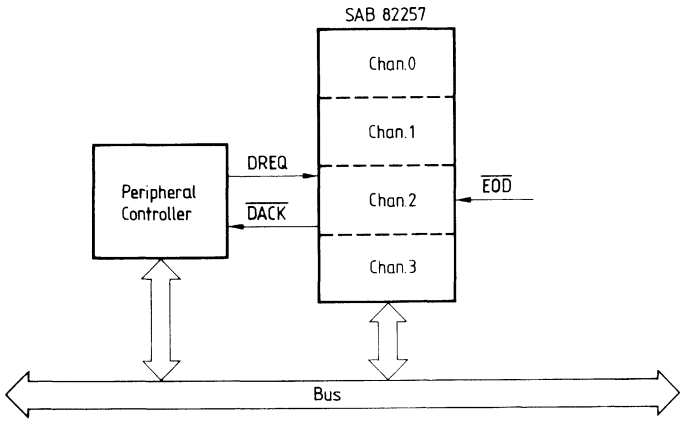
The following list shows some examples of peripheral controllers manufactured by Siemens or Intel Corporation:

Intel	Siemens
8271	SAB 1791
8272[1)]	SAB 1793
8273	SAB 1795
8274	SAB 1797
8275	SAB 2791
8291[1)]	SAB 2793
8294	SAB 2795
8295	SAB 2797

[1)] These devices react on the leading edge of the $\overline{\text{DACK}}$ signal. The other devices use the leading edge of the command to reset the request.

For using the above circuits in conjunction with the SAB 82257 in 286 mode, the bus cycle must be lengthened to guarantee the minimum command pulse width.
Lengthening must be performed with a certain number of T-states, before $\overline{\text{READY}}$ is issued by the peripheral or a special TTL logic. In this case also the falling edge of DREQ can be delayed by up to the same number of T-states.

Figure 86
Connecting the SAB 82257 to Peripherals



Anyhow, to be sure that with one request only a single data transfer is executed (no continuous request) certain timing rules must be considered. For a correct operation the **maximum delay time** t_{Dmax} from the leading edge of command to the falling edge of DREQ of the peripheral controller must satisfy the formula:

- **286 Mode**
  $t_{Dmax} = -t_{DEL} - t_{SU} + n \times 2 \times t_{CLK}$
- **186 Mode**
  $t_{Dmax} = 1.5 \times t_{CLK} - t_{DEL} - t_{SU} + n \times t_{CLK}$

t_{CLK} = Period of CLK; 1,5 × t_{CLK} means (1 + duty cycle) × t_{CLK}
t_{DEL} = Command output delay in bus controller or SAB 82257 (see pin ratings)
t_{SU} = Setup time of DREQ (falling edge)

The bus cycle can be lengthened by a number of n T-states (ready wait states). By selecting a sufficient number n, the use of comparatively slow peripheral controllers is possible. By increasing the number n the transfer rate is reduced. Therefore another method can be used by peripherals:

The request (DREQ) is reset with the leading edge of $\overline{DACK}$ (see note [1] of the table on page 194) because the $\overline{DACK}$ signal appears earlier than the command. In this case the **maximum delay time** t_{DDmax} from the leading edge of $\overline{DACK}$ to the falling edge of DREQ must satisfy the formula (not allowed for continuous requests for more than one single data transfer):

- **286 Mode**
  $t_{DDmax} = 2 \times t_{CLK} - t_{DDEL} - t_{SU} + n \times 2 \times t_{CLK}$
- **186 Mode**
  $t_{DDmax} = 2.5 \times t_{CLK} - t_{DDEL} - t_{SU} + n \times t_{CLK}$

t_{DDEL} = $\overline{DACK}$ output delay

The $\overline{EOD}$ signal, used by the SAB 82257 as well as by the peripheral, is compatible with most peripheral controller circuits. As an input as well as an output $\overline{EOD}$ is a pulse signal of two T-states' length.

11.5 Performance

11.5.1 Latencies

Preliminaries

1. The latency calculation described in this section does **not** take into account
   - setup delay times,
   - hold delay times,
   - output delay times and
   - delay times due to CPU accesses to the SAB 82257 in remote mode, which are specified in the data sheet. These delay times should be added to get the final latency figures.

2. The following is assumed:
   - The channel which latencies are calculated for, currently has the **highest** priority and will **not** be blocked by other still higher priority requests.
   - The command block and the data chain lists have **even** addresses.
   - Control space accesses will be performed with a **16-bit** bus.
3. In this section all timings are in units of T-states = 125 ns for standard 8 MHz systems.
4. If bus cycles are involved then
   - W = wait time during bus cycle due to slow devices (ready wait states).
   - T_B = time for one bus transfer:
     for 186/188/86/88 system $T_B = 4 + W$
     for 286 system $T_B = 2 + W$

DMA Request Processing in Local Mode
Figure 87 shows the functional flow from DREQ to $\overline{DACK}$ in local mode with the major elementary steps which are processed, and gives the timing for all these steps.
It as assumed that organizational and other unsynchronized transfers (e.g. prefetch) have been completed before the processing of DREQ starts.

DMA Request Processing in Remote Mode
Figure 88 shows the functional flow from DREQ to $\overline{DACK}$ in remote mode as described above.

Setup

- General Command: START command

Parameter	Limit values		
	min.	typ.	max.
Write to Setup	6.5	8	9.5
	+ HOLD/HLDA Sequence		

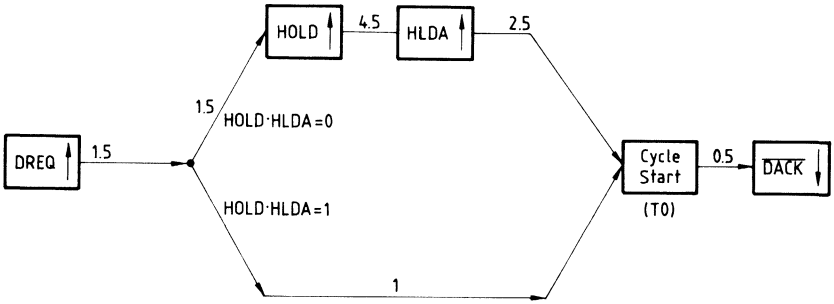
At this stage the START command is ready for the start of the channel setup routine.

- Channel Setup

$$7 \times T_B + 4 + t$$

$$t = \begin{cases} 1 \times T_B + 2, \text{ if list chaining enabled} \\ 3 \times T_B + 2, \text{ if linked list chaining enabled.} \\ 0, \text{ in other cases.} \end{cases}$$

Figure 87
DREQ to $\overline{\text{DACK}}$ Latency in Local Mode



Delays are given in terms of T-states (286 mode)

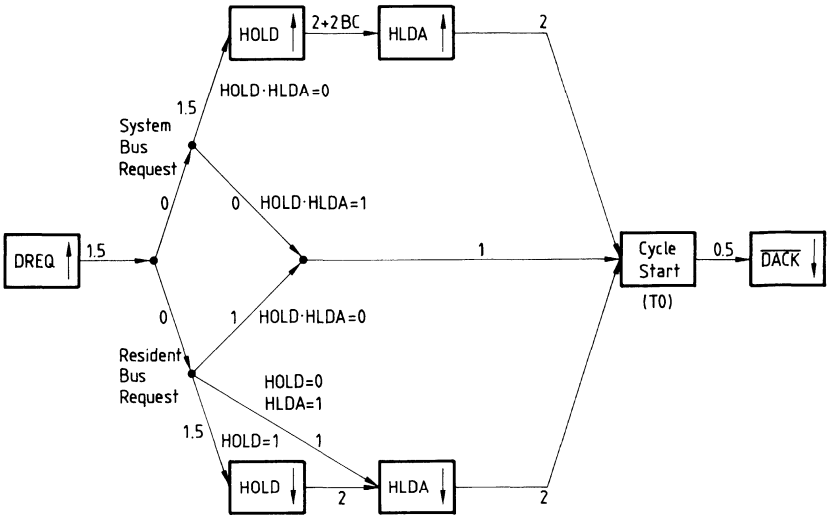
Parameter	Limit values		
	min.	typ.	max.
DREQ to HOLD	2.5	3	3 + W[1) 2)]
HOLD to HLDA	1	4.5	16 + 5 W[3)]
HLDA to Cycle Start	1.5	2.5	2.5
DREQ to Cycle Start (without bus arbitration)	2	2.5	4 + W[1)]
Cycle Start of $\overline{\text{DACK}}$	0.5	0.5	0.5

[1)] Single bus cycle running: 1 + W
Unseparable bus cycles running:
– Word access at odd address: 3 + 2W (same for pointer transfer)

[2)] General burst counter = 0: 2 × GDR
HLDA = 1, HOLD = 0 : Wait for HLDA = 0
HLDA lost : 2

[3)] Assumed repeat and lock prefixes are not combined

Figure 88
DREQ to $\overline{\text{DACK}}$ Latency in Remote Mode



Delays are given in terms of T-states (286 mode)
BC=Bus cycles

Parameter	Limit values		
	min.	typ.	max.
DREQ to HOLD Set	2.5	3	3 + W[1) 2)]
HOLD Set of HLDA Set	2 BC	2 + 2BC	[3)]
HLDA Set to Cycle Start	1.5	2	2.5
DREQ to HOLD Reset	1.5	3	5.5 + W[1)]
HOLD Reset to HLDA Reset	1	2	2
HLDA Reset to Cycle Start	1.5	2	2.5[1)]
DREQ to Cycle Start (without bus change)	2	3.5	5 + W
Cycle Start to $\overline{\text{DACK}}$	0.5	0.5	0.5

[1)] Single bus cycle running: 1 + W
Unseparable bus cycles running:
– word access at odd address: 3 + 2W (same for pointer transfer)
[2)] General burst counter = 0 : 2 × GDR
HLDA = 1, HOLD = 0 : Wait for HLDA = 0
HLDA lost : 2
[3)] Bus arbitration and currently running bus transfers.
BC = Multibus clock cycle.

Data Chaining
For data chaining operation latencies occur when the data block is changed. The latencies are

- for list chaining: $3 \times T_B + 6$
- for linked list chaining: $5 \times T_B + 6$

Termination and Command Chaining

- Type 1 Command
  - Termination
    Store channel status register (CSR) and calculate next command pointer: $1 \times T_B + 6$
  - Chaining
    Same as in the setup routine.
- Type 2 Command
  - Chaining
    Channel command register (CCR) load: $1 \times T_B$
    CCR decode and execution: $2 \times T_B + 2$
    Additionally for relative jump: 4
    Additionally for absolute jump: 4

11.5.2 Transfer Rates

The following table illustrates the maximum rates that can be achieved in different classes of DMA operation. Note that the transfer rates are not effected if the channels alternate.

	SAB 82257 in 286 Mode CLK = 16 MHz	SAB 82257 186 Mode CLK = 8 MHz
Single-Cycle DMA Transfer		
– Word Transfer	8 Mbytes/s	4 Mbytes/s
– Byte Transfer	4 Mbytes/s	2 Mbytes/s
Two-Cycle DMA Transfer		
– Word/Word Transfer	4 Mbytes/s	2 Mbytes/s
– Byte/Byte Transfer	2 Mbytes/s	1 Mbyte/s
Byte/Word Transfer	2.66 Mbytes/s	1.33 Mbyte/s

Note
The transfer rate figures for data chaining depend on the block length of each chained data block and on the organizational time of chaining in relation to the whole block length.

Device Specifications

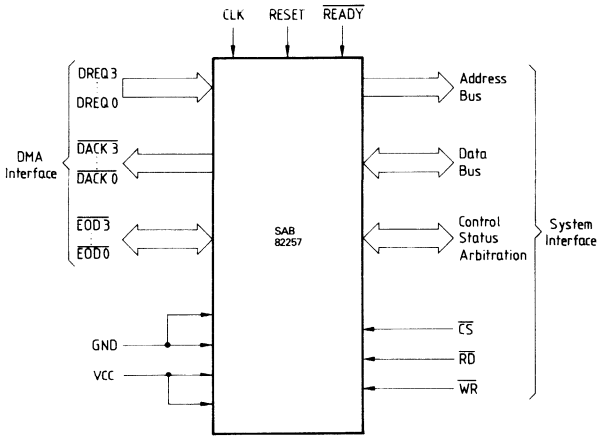
Preliminary

SAB 82257 High-Performance DMA Controller for 16-Bit Microcomputer Systems

SAB 82257 8 MHz
SAB 82257-6 6 MHz

- High-performance 16-bit DMA controller for the 16-bit family processors SAB 80286, SAB 80186/188, SAB 8086/88
- 4 independent high-speed DMA channels
- Adaptive on-chip bus interface for direct connection to processors
- Standalone operation for modular systems
- Programmable bus loading
- Transfer rates up to 8 Mbytes/s (8 MHz system)
- 16 Mbytes addressing range
- 16 Mbytes maximum block size
- Command chaining for automatic processing
- Automatic data chaining (scattering/gathering) for flexible data structures
- Single and double cycle transfers
- Automatic assembly/disassembly of data
- Memory-based communication scheme with CPU

Logic Symbol



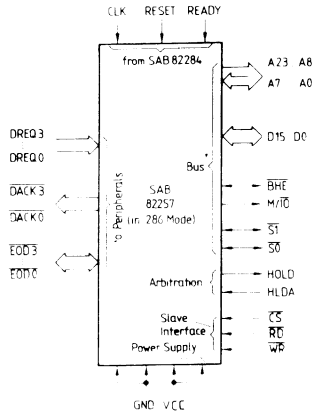
The SAB 82257 is a DMA (direct memory access) controller designed especially for the 16-bit microprocessors SAB 80286 and SAB 8086/186/88/188. In addition, the operation with other processors is supported by the remote mode. It has 4 independent DMA channels which can transfer data at rates up to 8 Mbytes/second at 8 MHz clock in an SAB 80286 system or up to 4 Mbytes/second at 8 MHz in an SAB 8086/80186 system. This great bandwidth allows the user to handle very fast data transfer or a large number of concurrent peripherals. The device is fabricated in advanced +5 V N-channel Siemens MYMOS technology and packaged in a 68-pin package.

Modes of Operation, Adaptive Bus Interface

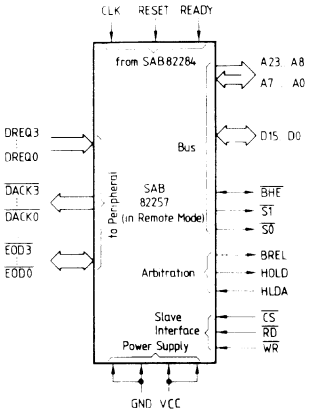
Like the advanced DMA controller SAB 82258, the SAB 82257 has been defined to work with all 16-bit processors, i.e. SAB 80286, SAB 80186/188 and SAB 8086/88 without additional support and interface logic. Hence the local buses of above processors are different in signals, functions and timings, the SAB 82257 has an adaptive bus interface to meet the different requirements of these local buses.

As a result of this, a bus compatibility with identical timing is attained with processors SAB 80286, SAB 80186 and SAB 8086. A compatibility with the 8-bit bus versions of these processors SAB 8088 and SAB 80188 is also guaranteed by defining the physical bus width of the SAB 82257 (per software) as 8 bits. The only difference in operation with SAB 8086 or SAB 80186 is that for SAB 8086 the HOLD pin functions as $\overline{RQ}/\overline{GT}$ line (if HLDA is held high on reset).

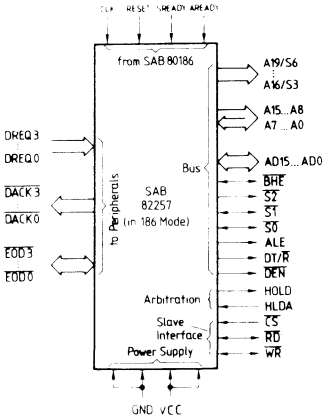
Logic Symbol in 286 Mode



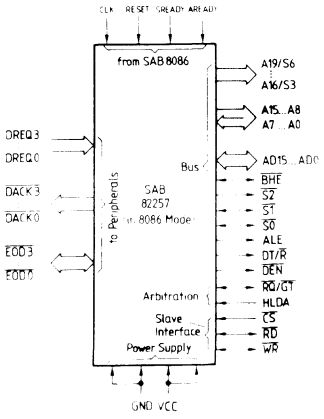
Logic Symbol in Remote Mode



Logic Symbol in 186 Mode



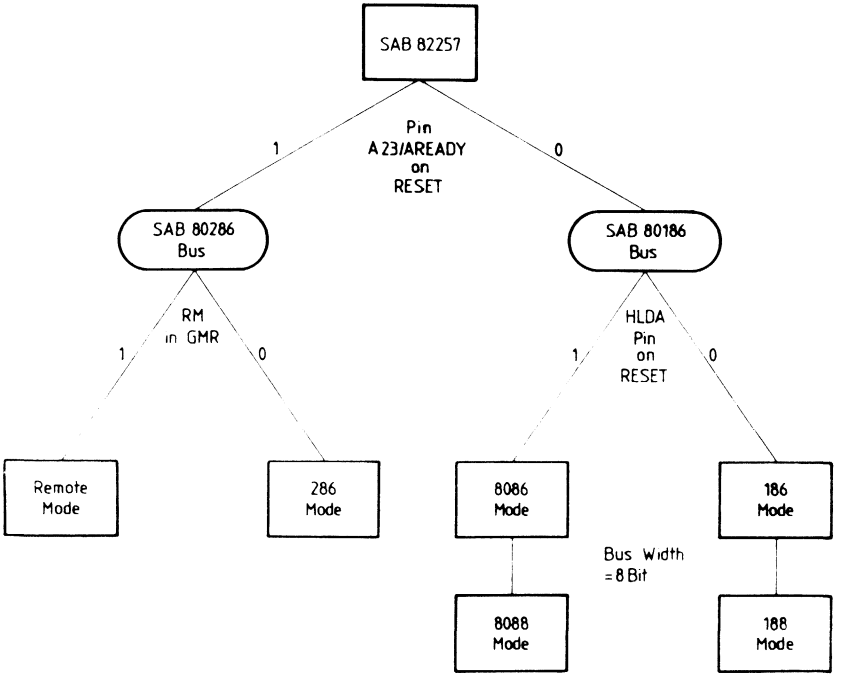
Logic Symbol in 8086 Mode



The SAB 82257 can also be operated in remote or standalone mode, in which case it is not coupled directly to a processor. In remote mode, the SAB 82257 can be operated as sole bus master in a multimaster environment.

The SAB 82257 is programmed to a specific mode of operation by applying defined logic levels to certain pins during reset and by setting the status of several control bits (see figure below).

Mode Selection



Pin Definitions and Functions

Some pins of the SAB 82257 serve for different purposes according to the different modes of bus operation. The table below summarizes the pinouts of the SAB 82257 in the various modes. A detailed description of the general pin functions as well as the mode-specific pin functions is given in the following sections.

Pin Names and Functions

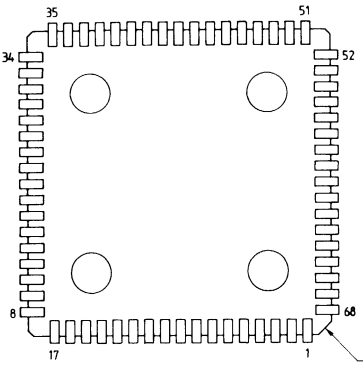
Pin	286 Mode		Remote Mode		186/8086 Mode	
	Symbol	Input (I) Output (O)	Symbol	Input (I) Output (O)	Symbol	Input (I) Output (O)
16	HOLD	O	HOLD	O	HOLD or $\overline{RQ}/\overline{GT}$	O (186) I/O (8086)
17	HLDA	I	HLDA	I	HLDA	I
1	$\overline{BHE}$	I/O	$\overline{BHE}$	I/O	$\overline{BHE}$	I/O
14	M/$\overline{IO}$	O	BREL	O	$\overline{S2}$	O
11	$\overline{S1}$	I/O	$\overline{S1}$	O	$\overline{S1}$	I/O
13	$\overline{S0}$	I/O	$\overline{S0}$	O	$\overline{S0}$	I/O
8	$\overline{CS}$	I	$\overline{CS}$	I	$\overline{CS}$	I
2	$\overline{RD}$	I	$\overline{RD}$	I	$\overline{RD}$	I/O
3	$\overline{WR}$	I	$\overline{WR}$	I	$\overline{WR}$	I/O
10	$\overline{READY}$	I	$\overline{READY}$	I	SREADY	I
59	A23	O	A23	O	AREADY	I
58	A22	O	A22	O	ALE	O
57	A21	O	A21	O	DT/$\overline{R}$	O
56	A20	O	A20	O	$\overline{DEN}$	O
55	A19	O	A19	O	A19/S6	O
54	A18	O	A18	O	A18/S5	O
53	A17	O	A17	O	A17/S4	O
52	A16	O	A16	O	A16/S3	O
51	A15	O	A15	O	A15	O
50	A14	O	A14	O	A14	O
49	A13	O	A13	O	A13	O
48	A12	O	A12	O	A12	O
47	A11	O	A11	O	A11	O
46	A10	O	A10	O	A10	O
45	A9	O	A9	O	A9	O
44	A8	O	A8	O	A8	O
42	A7	I/O	A7	I/O	A7	I/O
41	A6	I/O	A6	I/O	A6	I/O
40	A5	I/O	A5	I/O	A5	I/O
39	A4	I/O	A4	I/O	A4	I/O
38	A3	I/O	A3	I/O	A3	I/O
37	A2	I/O	A2	I/O	A2	I/O
36	A1	I/O	A1	I/O	A1	I/O
35	A0	I/O	A0	I/O	A0	I/O

Pin Names and Functions (cont'd)

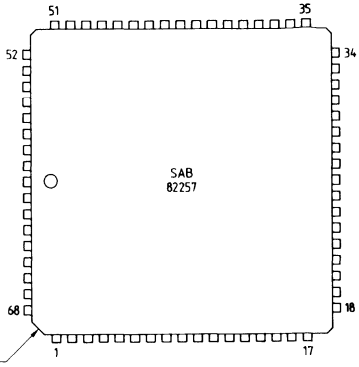
Pin	286 Mode		Remote Mode		186/8086 Mode	
	Symbol	Input (I) Output (O)	Symbol	Input (I) Output (O)	Symbol	Input (I) Output (O)
18	D15	I/O	D15	I/O	AD15	I/O
20	D14	I/O	D14	I/O	AD14	I/O
22	D13	I/O	D13	I/O	AD13	I/O
24	D12	I/O	D12	I/O	AD12	I/O
27	D11	I/O	D11	I/O	AD11	I/O
29	D10	I/O	D10	I/O	AD10	I/O
31	D9	I/O	D9	I/O	AD9	I/O
33	D8	I/O	D8	I/O	AD8	I/O
19	D7	I/O	D7	I/O	AD7	I/O
21	D6	I/O	D6	I/O	AD6	I/O
23	D5	I/O	D5	I/O	AD5	I/O
25	D4	I/O	D4	I/O	AD4	I/O
28	D3	I/O	D3	I/O	AD3	I/O
30	D2	I/O	D2	I/O	AD2	I/O
32	D1	I/O	D1	I/O	AD1	I/O
34	D0	I/O	D0	I/O	AD0	I/O
7	DREQ0	I	DREQ0	I	DREQ0	I
6	DREQ1	I	DREQ1	I	DREQ1	I
5	DREQ2	I	DREQ2	I	DREQ2	I
4	DREQ3	I	DREQ3	I	DREQ3	I
61	$\overline{\text{DACK0}}$	O	$\overline{\text{DACK0}}$	O	$\overline{\text{DACK0}}$	O
62	$\overline{\text{DACK1}}$	O	$\overline{\text{DACK1}}$	O	$\overline{\text{DACK1}}$	O
63	$\overline{\text{DACK2}}$	O	$\overline{\text{DACK2}}$	O	$\overline{\text{DACK2}}$	O
64	$\overline{\text{DACK3}}$	O	$\overline{\text{DACK3}}$	O	$\overline{\text{DACK3}}$	O
65	$\overline{\text{EOD0}}$	I/O	$\overline{\text{EOD0}}$	I/O	$\overline{\text{EOD0}}$	I/O
66	$\overline{\text{EOD1}}$	I/O	$\overline{\text{EOD1}}$	I/O	$\overline{\text{EOD1}}$	I/O
67	$\overline{\text{EOD2}}$	I/O	$\overline{\text{EOD2}}$	I/O	$\overline{\text{EOD2}}$	I/O
68	$\overline{\text{EOD3}}$	I/O	$\overline{\text{EOD3}}$	I/O	$\overline{\text{EOD3}}$	I/O
15	RESET	I	RESET	I	RESET	I
12	CLK	I	CLK	I	CLK	I
9,43	GND	(Ground)	GND	(Ground)	GND	(Ground)
26,60	VCC	(Power Supply)	VCC	(Power Supply)	VCC	(Power Supply)

Pin Configuration

PLCC 68
Bottom view



PLCC 68
Top view



Pin Definitions for All Operating Modes

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$\overline{BHE}$</td><td>1</td><td>I/O</td><td>$\overline{BUS\ HIGH\ ENABLE}$
Indicates transfer of data on the upper byte of the data bus, D15 to D8. Eight-bit oriented devices assigned to the upper byte of the data bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is active low and floats to tristate off when the SAB 82257 does not own the bus.
$\overline{BHE}$ and A0 encodings
<table>
<tr><th>$\overline{BHE}$</th><th>A0</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>Word transfer (D15–D0)</td></tr>
<tr><td>0</td><td>1</td><td>Byte transfer on upper half of data bus (D15–D8)</td></tr>
<tr><td>1</td><td>0</td><td>Byte transfer on lower half of data bus (D7–D0)</td></tr>
<tr><td>1</td><td>1</td><td>Odd-addressed byte on 8-bit bus (D7–D0)</td></tr>
</table></td></tr>
<tr><td>$\overline{RD}$</td><td>2</td><td>I</td><td>$\overline{READ}$
This command in conjunction with chip select enables reading out of the SAB 82257 register which is addressed by the address lines A7 to A0. This signal can be asynchronous to the SAB 82257 clock.</td></tr>
<tr><td>$\overline{WR}$</td><td>3</td><td>I</td><td>$\overline{WRITE}$
This command is used for writing into SAB 82257 registers. This signal can be asynchronous to the SAB 82257 clock.</td></tr>
<tr><td>DREQ0-
DREQ3</td><td>4-7</td><td>I</td><td>DMA REQUEST (0 TO 3)
These input signals are used for synchronized DMA transfers. These signals can be asynchronous to the SAB 82257 clock.</td></tr>
<tr><td>$\overline{CS}$</td><td>8</td><td>I</td><td>$\overline{CHIP\ SELECT}$
Is used to enable the access of a processor to SAB 82257 registers. This access is additionally controlled either by bus status signals or by the read or write command signals. Chip select can be asynchronous to the SAB 82257 clock.</td></tr>
<tr><td>CLK</td><td>12</td><td>I</td><td>CLOCK
It provides the fundamental timing. In 286 mode and remote mode it must be two times the system clock. It can be directly connected to the SAB 82284 CLK output. It is divided by two to generate the SAB 82257 internal clock. The on-chip divide-by-two circuitry can be synchronized to the external clock generator by a low-to-high transition on the RESET input, or by first high-to-low transition on the status inputs $\overline{S0}$ or $\overline{S1}$ after reset. In 186/8086 mode no internal prescaling is done.</td></tr>
</table>

Pin Definitions for All Operating Modes (cont'd)

<table>
<tr><th>Symbol</th><th>Pin</th><th>Input (I)
Output (O)</th><th>Function</th></tr>
<tr><td>$\overline{S0}$, $\overline{S1}$</td><td>11, 13</td><td>I/O</td><td>BUS STATUS LINES (0, 1)
These signals control the support circuits. The beginning of a bus cycle is indicated by $\overline{S1}$ or $\overline{S0}$ or both going active. The termination of a bus cycle is indicated by all status signals going inactive in 186 mode or bus ready signal ($\overline{READY}$) going active in 286 mode. The type of bus cycle is indicated by $\overline{S0}$, $\overline{S1}$ and $\overline{S2}$ (in 186 mode) or M/$\overline{IO}$ (in 286 mode). $\overline{S2}$ and M/$\overline{IO}$ have the same meaning but in 186 mode the $\overline{S2}$ signal can be active only when at least one of $\overline{S1}$ or $\overline{S0}$ is active, whereas in 286 mode the M/$\overline{IO}$ signal is valid with the address on the address lines. The SAB 82257 can generate the following bus cycles by activating the status signals (and M/$\overline{IO}$ in 286 mode):
<table>
<tr><th>M/$\overline{IO}$ or $\overline{S2}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>Cycle Type</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Read I/O-vector (for multiplexer channel)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read from I/O space</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write into I/O space</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>No bus cycle, does not occur in 186 mode</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Does not occur</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read from memory space</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write into memory space</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>No bus cycle</td></tr>
</table>
When the SAB 82257 is not the master of the local bus the status signals are used as inputs for detection of synchronous accesses to the SAB 82257. The following table shows the bus status and $\overline{CS}$, signals and their interpretation by the SAB 82257.
<table>
<tr><th>$\overline{CS}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>Description</th></tr>
<tr><td>1</td><td>X</td><td>X</td><td>SAB 82257 is not selected (no action)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>No SAB 82257 access (no action)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read from an SAB 82257 register</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write into an SAB 82257 register</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>No bus cycle (note 1)</td></tr>
</table>
Note 1: SAB 82257 is selected but no synchronous access is activated. In this case the SAB 82257 monitors $\overline{RD}$ and $\overline{WR}$ signals for detection of an asynchronous access.</td></tr>
<tr><td>RESET</td><td>15</td><td>I</td><td>SYSTEM RESET
An activation of the reset signal forces the SAB 82257 to the initial state. The reset signal must be synchronous to CLK.</td></tr>
</table>

Pin Definitions for All Operating Modes (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
$\overline{DACK0}$- $\overline{DACK3}$	61–64	O	$\overline{DMA\ ACKNOWLEDGE}$ (0 TO 3) Acknowledges the requests on the related DREQn signal. It is activated when the requested transfer(s) is (are) performed.
$\overline{EOD0}$- $\overline{EOD3}$	65–68	I/O	$\overline{END\ OF\ DMA}$ (0 TO 3) These signals are implemented as open drain output drivers with a high impedance pullup resistor and thus can be used as bidirectional lines. As **outputs** the signals are activated for two system clock cycles at the end of the DMA transfer of the corresponding channel (if enabled) or they are activated under program control (EOD output or interrupt output). If the signals are held internally high but forced to low by external circuitry, they act as "End of DMA" **inputs**. The current transfer is aborted and the SAB 82257 continues with the next command. Additionally, a special function is possible with the $\overline{EOD2}$ pin: this pin can also be used as common interrupt signal for all 4 channels. In this mode this signal is not an open drain output but a pushpull output (output only). The other $\overline{EOD}$ pins may be used as $\overline{EOD}$ outputs/inputs as described above.
VCC	26,60		POWER SUPPLY (+5V)
GND	9,43		GROUND (0V)

Pin Definitions for 286 Mode and Remote Mode

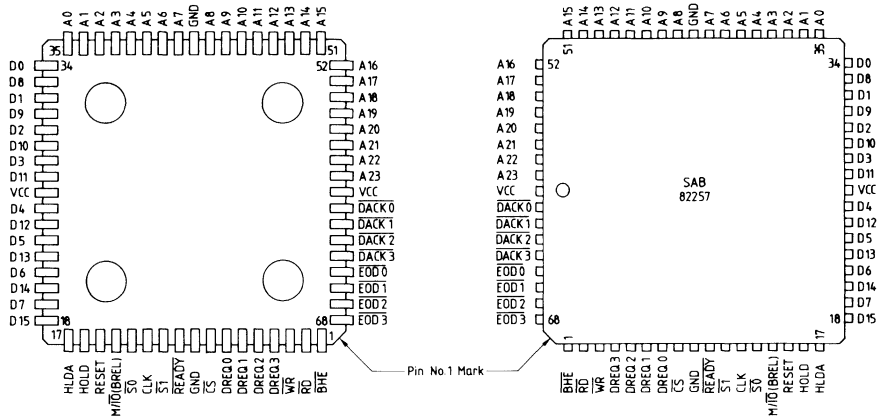
In 286 mode the SAB 82257 bus signals and bus timings are the same as for the SAB 80286 processor. Additional features of the SAB 82257 require a slight change in pin definitions. The processor can access internal registers of the SAB 82257. Therefore the bus signals must support these accesses. This means that some of the bus control signals must be bidirectional and some additional bus control signals are necessary. All additional pins and their functions are listed below.

In **remote mode** most of the bus signals are the same as in 286 mode. Pin 14 (M/$\overline{IO}$) serves as BREL output. The HOLD/HLDA arbitration in remote mode is used only for system bus accesses, the resident bus is accessed directly.
The $\overline{CS}$ input additionally requests access to the local bus of the SAB 82257. These accesses are enabled through the BREL output after the SAB 82257 has released the bus.

Pin Configuration in 286 Mode and Remote Mode

Component Pad View – As viewed from underside of component when mounted on the board

PC Board View – As viewed from the component side of the pc board



Pin Definitions for 286 Mode and Remote Mode (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
$\overline{\text{READY}}$	10	I	$\overline{\text{BUS READY}}$ Terminates a bus cycle. Bus cycles are extended without limit until terminated by $\overline{\text{READY}}$ low. $\overline{\text{READY}}$ is an active low synchronous input requiring setup and hold times relative to the system clock to be met for correct operation.
M/$\overline{\text{IO}}$	14 (286 mode)	O	MEMORY/ $\overline{\text{I/O}}$ SELECT In 286 mode, pin 14 is used to distinguish between memory and I/O space addresses.
BREL	14 (remote mode)	O	BUS RELEASE In remote mode pin 14 is used to indicate when the SAB 82257 has released the control of the local bus.
HOLD	16	O	BUS HOLD REQUEST When true, indicates a request for control of the local bus (286 mode) or the system bus (remote mode). When the SAB 82257 relinquishes the bus it drops the HOLD output. HOLD is connected to the bus arbiter in remote mode.
HLDA	17	I	BUS HOLD ACKNOWLEDGE When true, indicates that the SAB 82257 can acquire the control of the bus. When it goes low SAB 82257 must relinquish the bus at the end of its current cycle. HLDA can be asynchronous to the SAB 82257 clock. HLDA is connected to the bus arbiter in remote mode.
D0-D15	18-25, 27-34	I/O	DATA BUS (0 TO 15) This is the bidirectional 16-bit data bus. For use with an 8-bit bus, only the lower 8 data lines D7-D0 are relevant.
A0-A7	35-42	I/O	ADDRESS BUS (0 TO 7) The lower 8 address lines for DMA transfers. They are also used to input the register address when the processor accesses an SAB 82257 register.
A8-A23	44-59	O	ADDRESS BUS (8 TO 23) Higher address outputs.

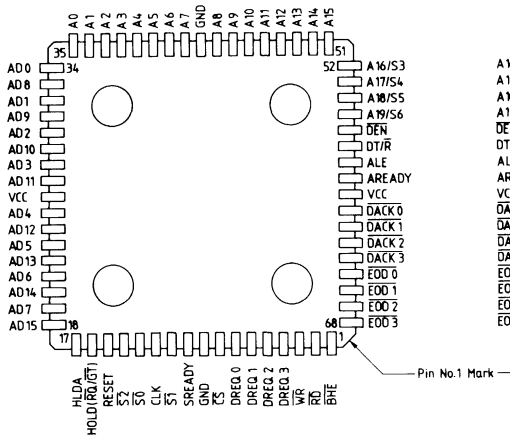
Pin Definitions for 186 Mode and 8086 Mode

In 186 mode and 8086 mode the SAB 82257 multiplexes the address with data and additional status lines.
Pins A0 to A15 retain their original function while pins A20 to A23 serve for different purposes (not used for address in 186/8086 mode).

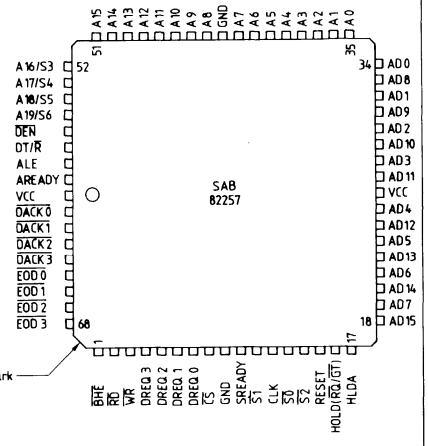
The $\overline{RD}$ and $\overline{WR}$ lines are additionally used as outputs in 186/8086 mode to support minimum mode systems.
Note that the HLDA input can be used to force the SAB 82257 off the bus in 8086 mode, even though the arbitration is done via the $\overline{RQ}/\overline{GT}$ line!

Pin Configuration in 186 Mode and 8086 Mode

Component Pad View – As viewed from underside of component when mounted on the board



PC Board View – As viewed from the component side of the pc board



Pin Definitions for 186 Mode and 8086 Mode (cont'd)

Symbol	Pin	Input (I) Output (O)	Function
ALE	58	O	ADDRESS LATCH ENABLE This signal provides a strobe to separate the address information on the multiplexed AD lines.
$\overline{DEN}$	56	O	$\overline{DATA\ ENABLE}$ This signal is used for enabling the data transceiver.
DT/$\overline{R}$	57	O	DATA TRANSMIT/$\overline{RECEIVE}$ This signal controls the direction of the data transceivers. When low, data is transferred to the SAB 82257, when high the SAB 82257 places data onto the data bus.
$\overline{S2}$	14	O	$\overline{STATUS\ LINE\ 2}$ Signal as for SAB 186/8086/88 processors (see also $\overline{S1}$, $\overline{S0}$ description in 286 mode).
AREADY	59	I	ASYNCHRONOUS READY The rising edge of this signal is internally synchronized, the falling edge must be synchronous to CLK. During reset this signal must be low for entering the 186 mode.
SREADY	10	I	SYNCHRONOUS READY This signal must be synchronized externally. The use of this pin permits a relaxed system-timing specification by eliminating the clock phase which is required for resolving the signal level when using the AREADY input.
CLK	12	I	SYSTEM CLOCK This is the input for the one time system clock. No internal prescaling is done.
AD0– AD15	18–25 27–34	I/O	ADDRESS/DATA BUS (0 TO 15) Lower address and data information is multiplexed on pin AD0 to AD 15. Additionally the demultiplexed address information is available on address pin A0 to A15.
A0–A7 A8–A15	35–42 44–51	I/O O	
A16/S3– A19/S6	52, 55	O	ADDRESS BUS (16 TO 19) / STATUS LINES (3 TO 6) The higher address bits are multiplexed with additional status information.
HLDA	17	I	BUS HOLD ACKNOWLEDGE When true, indicates that the SAB 82257 can acquire the control of the bus. When it goes low the SAB 82257 must relinquish the bus at the end of its current bus cycle. HLDA can be asynchronous to the SAB 82257 clock. In 8086 mode, HLDA can be used to force the SAB 82257 off the bus.
HOLD	16 (186 mode)	O	BUS HOLD REQUEST When true, indicates a request for control of the bus. When the SAB 82257 relinquishes the bus, it drops the HOLD output.
$\overline{RG}$/$\overline{GT}$	16	I/O	$\overline{REQUEST}$/$\overline{GRANT}$ In 8086 mode the HOLD output acts as $\overline{REQUEST}$/$\overline{GRANT}$ line. The $\overline{REQUEST}$/$\overline{GRANT}$ protocol implements a one-line communication dialog required to arbitrate the use of the system bus normally done via HOLD/HLDA. The $\overline{RG}$/$\overline{GT}$ signal is active low and has an internal pullup resistor.

Functional Description

General

The SAB 82257 is an advanced general-purpose DMA controller especially tailored for efficient high-speed data transfers on an SAB 80286 as well as on an SAB 80186/188 or SAB 8086/88 bus.
It supports two basic operating modes:

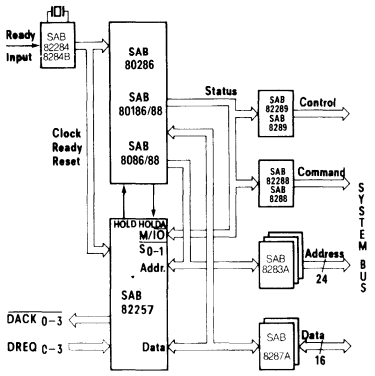
– local mode (tightly coupled to a processor) and
– remote mode (loosely coupled to a processor).

In the first case the SAB 82257 is directly coupled to the CPU and uses the same system support/control devices as the CPU (see figure a) below). This mode is possible with the above-mentioned processors.

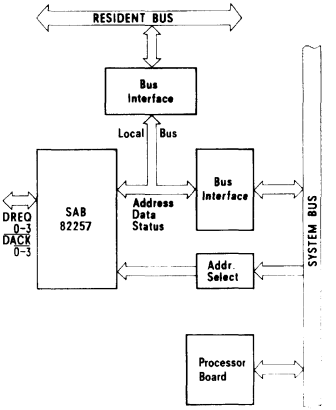
As a second basic operating mode a remote (standalone) mode is supported (see figure b) below). Here the SAB 82257 has his own sets of bus interface circuits and thus can dispose of its own local bus. This allows the DMA controller to work in parallel with the main CPU and therefore overall system performance could be increased. Besides, this mode is very useful for the design of modular systems and allows connecting the SAB 82257 to any other processor via the system bus independent of the processor's unique local bus.

Basic SAB 82257 Operating Modes

a) Local Mode

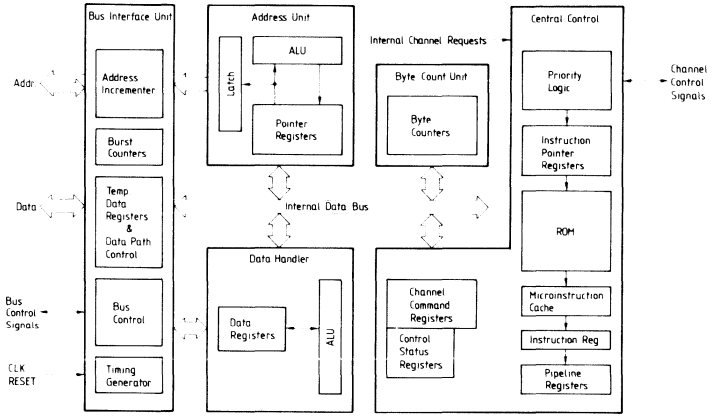


b) Remote Mode



The SAB 82257 has four independent DMA channels that can transfer up to 8 Mbytes/s in the single cycle mode (2 clocks/transfer). In the 2-cycle transfer mode the maximum rate is 4 Mbytes/s. Switching between channels induces no time penalty. Thus the overall maximum transfer rate of 8 Mbytes/s is also valid for multiple channel operation.
This fast operation is possible because of the pipelined architecture of the SAB 82257 that allows the different function units to work in parallel.

Block Diagram of SAB 82257



The SAB 82257 supports two address spaces, memory space and I/O space, each with a maximum address range of 16 Mbytes. In addition, the maximum block length (byte count) is also 16 Mbytes to support applications where large blocks of data have to be transferred (e.g. graphics).

As source or as destination, four parameters can be selected independently:

- address space (memory or I/O)
- physical bus width (8 bits or 16 bits),
- logical bus width (same as physical bus width or 8 bits on a 16-bit physical bus) and
- transfer direction (increasing, decreasing, fixed pointer or constant value).

If the physical bus width of source or destination does not meet the logical bus width an automatic byte/word assembly (word/byte disassembly) takes place if this minimizes the necessary transfers. The same is true if the logical bus widths of source and destination are different.
Transfers between different address spaces can be performed within one cycle or in two cycles, transfers within one address space can be performed only in two cycles.

The transfers can be executed free running or externally synchronized via DREQ where source or destination synchronization is possible.

In summary, this very symmetrical operation of the SAB 82257 gives the user a great amount of design flexibility.

Adaptive Bus Interface

As shown in the figure on page 3, the SAB 82257 bus interface has two basic timing modes: the 286 mode and the 186 mode. In 286 mode the SAB 82257 is directly coupled to an SAB 80286, in 186 mode to an SAB 80186 or SAB 80188. For each of these two modes a slightly different variation exists:

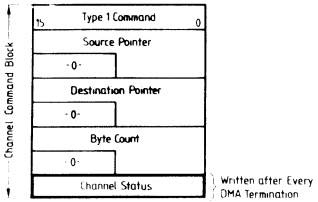
- For the 286 mode, the remote mode, where the SAB 82257 operates as a bus master on the system bus without being directly coupled to a processor. In this mode the SAB 82257 can dispose of its own local bus and the communication with the main processor is done via the system bus. To enable access to SAB 82257 registers by the main processor, the SAB 82257 must release its local bus. This "local bus arbitration" in remote mode is done via the $\overline{CS}$ and BREL lines.
- For the 186 mode the variation is the 8086 mode where the SAB 82257 supports the $\overline{RQ}/\overline{GT}$ protocol and thus can be directly coupled to an SAB 8086 or SAB 8088.

Memory-Based Communication

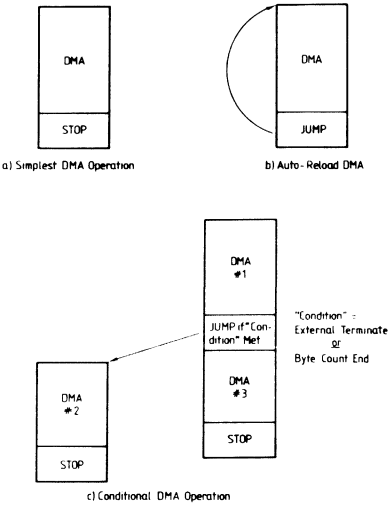
The normal communication between the SAB 82257 and the processor is memory-based. This means that all necessary data for a transfer is contained in a command block in memory accessible for CPU and SAB 82257 (see figure on next page). To start the transfer the CPU loads one of the command pointer registers of the SAB 82257 with the address of the command block and then gives a "start channel command". Getting the command the SAB 82257 loads the entire command block from memory into its on-chip channel registers and executes it.
On completing the operation, channel status information is written back by the SAB 82257 into the channel status word contained in the command block in memory. The command block structure of the SAB 82257 is identical with the structure of the SAB 82258 short command blocks. This allows to portate SAB 82257 software to the SAB 82258 and vice versa (in this case with restrictions).

Memory-Based Communication and Command Chaining

Memory-Based Communication



Command Chaining



Command Chaining

Command blocks for any channel can be chained for sequential execution (see figure above). When the SAB 82257 has completed the execution of a command, it automatically increments the command pointer, and starts to fetch and execute the next command block until a stop command is found. As a result a chain of command blocks can be executed by the SAB 82257 without any CPU intervention. Due to conditional and unconditional STOP and JUMP commands, quite complex sequences of DMA can be executed by the SAB 82257.

Data Chaining

Data chaining permits an automatic, dynamic linking of data blocks scattered in memory. There are two types: list and linked-list data chaining.

If for a DMA the source blocks are to be dynamically linked during DMA it is called source chaining and the effect is that of gathering data blocks and sending them out effectively as one block.

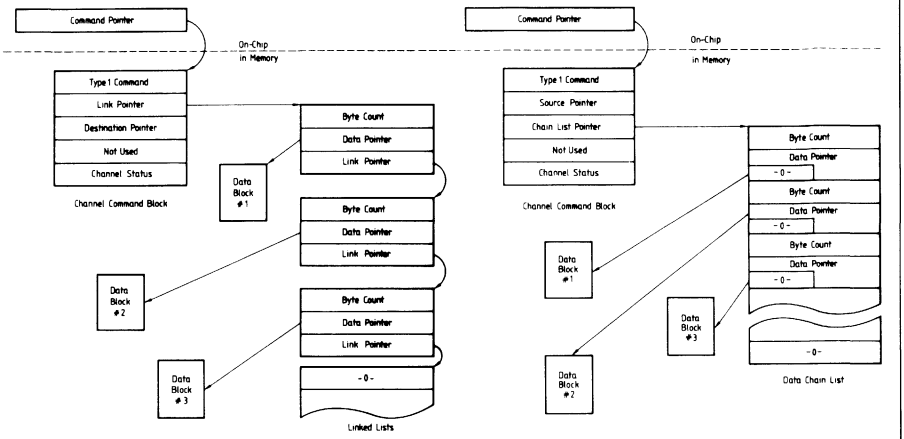
If one source block is dynamically broken up into multiple destination blocks, it is called destination chaining. This results in scattering of a block.

This dynamic linking and unlinking of data blocks makes the logical sequencing of data independent of its physical sequencing in memory.

Data Chaining

a) Linked List Chaining

b) List Chaining



In the case of linked list chaining (see figure a) above) each data block has a descriptor containing information on position of data block in memory, length of data block, and a pointer to the next descriptor.

During data transfer the data block 1 is sent out first, then 2 and so on till a 0 is encountered in the byte count field.

The second type of data chaining is list chaining (see figure b) above).

Unlike linked list chaining, here the data block descriptors are continuous in a block and thus determine the sequence of data blocks. The flexibility lost in terms of predefined sequence is gained in terms of linking time.

Operating the SAB 82257

Reset

When activating the reset input, the SAB 82257 is forced into its initial state. All channels and bus activities are stopped, tristate lines are tristated and the others enter the inactive state.
While the reset input is active, line A23/AREADY and HLDA must be forced to the appropriate levels to select the desired bus interface mode (see figures on page 3, 40 and 52).
After deactivating reset the inactive state is maintained, in addition the state of the SAB 82257 registers is as follows:

- general mode register, general burst register, general delay register, general status register and the four channel status registers are set to zero,
- all other registers and bits are undefined.

Note that the general mode register (GMR) should be loaded first to select the mode of operation before any other activity is started on the SAB 82257.

DMA Interface

The DMA interface consists of three lines:

- DREQ – DMA request,
- $\overline{\text{DACK}}$ – DMA acknowledge and
- $\overline{\text{EOD}}$ – end of DMA

The first two lines work as request and acknowledge lines to control synchronized DMA transfers as known from conventional DMA controllers.

A special feature of the SAB 82257 are the bidirectional $\overline{\text{EOD}}$ lines. Firstly they can be used as inputs to receive an asynchronous external terminate signal to terminate a running DMA. Secondly, as an output, they can be used to send out a pulse which interrupts the CPU and/or signals to the peripheral a specific status (e.g. transfer aborted, or end of a block, or send/receive next block ...).

The $\overline{\text{EOD}}$ output signal can be generated synchronously to a transfer (during the last transfer) or asynchronously to the transfers by a specific command.

In addition the $\overline{\text{EOD}}$ output of channel 2 can be used as a collective interrupt output for all DMA channels while the other three retain their normal function.

Slave Interface

The slave interface is used to access the SAB 82257 internal registers. Although nearly all of the communication between CPU and SAB 82257 is done via memory-based data blocks, some direct accesses to SAB 82257 registers are necessary. For example during the initialization phase the general mode register must be written, or to start a channel the command pointer register and the general command register must be loaded. Also during the debugging phase it is of great benefit to have access to all of the SAB 82257 internal registers.

The slave interface is enabled by the $\overline{\text{CS}}$ input and consists of the following lines:

- $\overline{\text{S0}}$, $\overline{\text{S1}}$ – status lines (inputs)
- $\overline{\text{RD}}$, $\overline{\text{WR}}$ – control lines (inputs)
- A0–A7 – register address (inputs)
- D0–D15 – data lines (inputs/outputs) and
- AD0–AD15 – address/data lines (inputs/outputs) for synchronous access in 186 mode

Note, that all of these lines are outputs if the SAB 82257 is an active bus master.

In 186 mode and 286 mode two types of accesses are possible:

- Synchronous access by means of the status lines. Processor and SAB 82257 are directly coupled and must use the same clock.
- Asynchronous access by using the control lines $\overline{\text{RD}}$ and $\overline{\text{WR}}$ (processor and SAB 82257 may have different clocks).

  In all modes except the synchronous access in 186 mode the register address must be supplied on address pins A0 to A7. Using synchronous access in 186 mode the address information is expected at address/data lines AD0 to AD7.

  In remote mode only the asynchronous access is possible because the SAB 82257 first has to release its local bus to enable the register access. On receiving an access request (activation of $\overline{\text{CS}}$ input) the SAB 82257 releases its local bus as soon as possible and signals this by activating the BREL line. Now the CPU can accomplish its access.

Bus Arbitration

To arbitrate access to the bus between the SAB 82257 and the processor, the signals HOLD and HLDA serve for communication. Normally the SAB 82257 competes for the bus via HOLD, the processor grants access to the bus via HLDA. The HLDA signal can also be deactivated in order to force the SAB 82257 off the bus for a certain reason (kick off). After reactivation of HLDA, the SAB 82257 will again get control of the bus.
In 8086 mode this communication is done by pulses via a single $\overline{\text{RQ}}/\overline{\text{GT}}$ line which uses the HOLD pin. In this case normally the HLDA input has no function. Nevertheless, even in 8086 mode the HLDA input can be used for kick-off. This provides some kind of additional bus arbitration.

Register Set

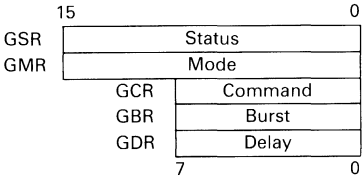
The following figure shows the user visible registers of the SAB 82257. A set of 3 registers, called the general registers, is used for all the 4 channels. The mode register is being written to first after reset and it describes the SAB 82257 environment – bus widths, etc. The general command register (GCR) is used to start and stop the DMA transfer on different channels. The general status register (GSR) shows the status of all the 4 channels; if the channel is running, if interrupt is pending, etc.

There is a set of channel registers for each of the 4 channels. Most channel registers serve as cache registers and need to be accessed only for debugging. During normal operation they are loaded automatically by the SAB 82257 (see next paragraph).

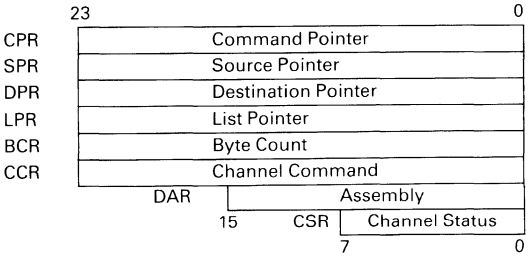
The layout of register addresses is shown in the figure on the next page. All register addresses are even. Locations not designated in that figure are reserved and should not be used.

SAB 82257 Register Set

General Registers



Channel Registers (4 sets; 1 per channel)



Register Address Arrangement

Address Bits 0–5	Address Bits 7, 6 00	01	10	11
0	GCR			
2				
4	GSR			
6				
8	GMR			
A	GBR			
C	GDR			
E				
10	CSR 0	CSR 1	CSR 2	CSR 3
12	DAR 0	DAR 1	DAR 2	DAR 3
14				
16				
18				
1A				
1C				
1E				
20	CPR L0	CPR L1	CPR L2	CPR L3
22	CPR H0	CPR H1	CPR H2	CPR H3
24	SPR L0	SPR L1	SPR L2	SPR L3
26	SPR H0	SPR H1	SPR H2	SPR H3
28	DPR L0	DPR L1	DPR L2	DPR L3
2A	DPR H0	DPR H1	DPR H2	DPR H3
2C				
2E				
30	LPR L0	LPR L1	LPR L2	LPR L3
32	LPR H0	LPR H1	LPR H2	LPR H3
34				
36				
38	BCR L0	BCR L1	BCR L2	BCR L3
3A	BCR H0	BCR H1	BCR H2	BCR H3
3C	CCR L0	CCR L1	CCR L2	CCR L3
3E	CCR H0	CCR H1	CCR H2	CCR H3

GCR = General Command Register
GSR = General Status Register
GMR = General Mode Register
GBR = General Burst Register
GDR = General Delay Register
CSR = Channel Status Register
DAR = Data Assembly Register

CPR = Command Pointer Register
SPR = Source Pointer Register
DPR = Destination Pointer Register
LPR = List Pointer Register
BCR = Byte Count Register
CCR = Channel Command Register

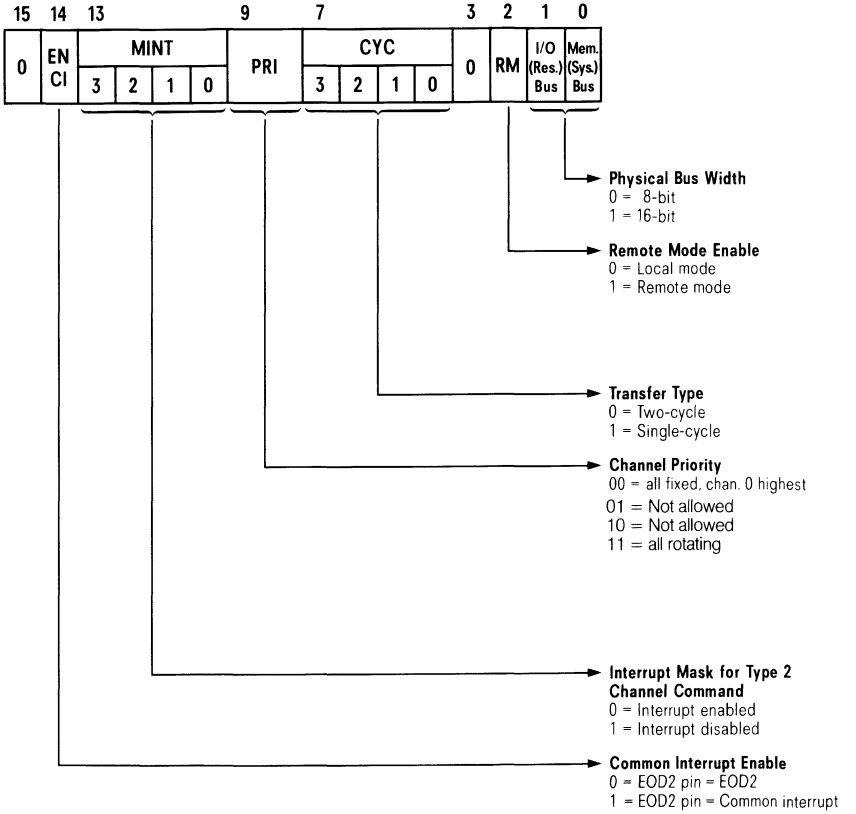
Register Description

General Mode Register

In the general mode register GMR (figure below) the system wide parameters are specified.

This register should be programmed first after reset; with an 8-bit bus program low byte first.

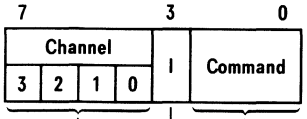
General Mode Register (GMR)



General Command Register

Individual channels are started and stopped by a command written to the general command register GCR (figure below). The GCR is directly loaded by the CPU.

General Command Register (GCR)



Channel Command

000 = NOP		
001 = CONTINUE		Channel(s) after it has been stopped by the STOP command
010 = START		Channel(s) = Command block at system/memory space
011 = START		Channel(s) = Command block at resident I/O space
100 = STOP		Channel(s)
101 =	}	Not allowed
110 =	}	
111 = HALT		Single-step channel(s): Start execution and stop before next command block will be loaded

Interrupt Command

0 = NOP
1 = Clear interrupt(s) of channel(s)

Channel Select

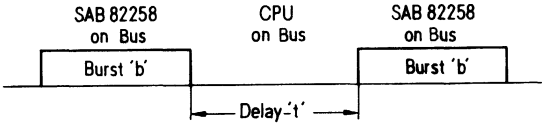
General Burst and Delay Register

It is possible to restrict the bus load generated by the SAB 82257 on the CPU bus by programming the burst and the delay register. The bus load is defined by the formula given in figure a) below. The factor b (burst) is programmed in the general burst register GBR, t (delay time) in the general delay register GDR (see figures b and c).

Since the SAB 82257 can also execute locked bus cycles, the maximum burst length consists of b+3 (8-bit bus) or b+2 (16-bit bus) bus cycles. GBR and GDR must be directly loaded by the CPU. Loading GBR with 0 leads to no bus load limitations for the SAB 82257 (default after reset).

General Burst and Delay Register

a) **Bus Loading**

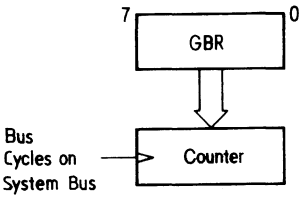


$$\text{Bus Load Due to SAB 82258} = \frac{b}{b+t}$$

b) **General Burst Register (GBR)** - to Program 'b'

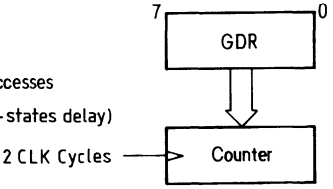
Determines Max. Number of Contiguous Bus Cycles from SAB 82258

If GBR=0, No Limit



c) **General Delay Register (GDR)** - to Program 't'

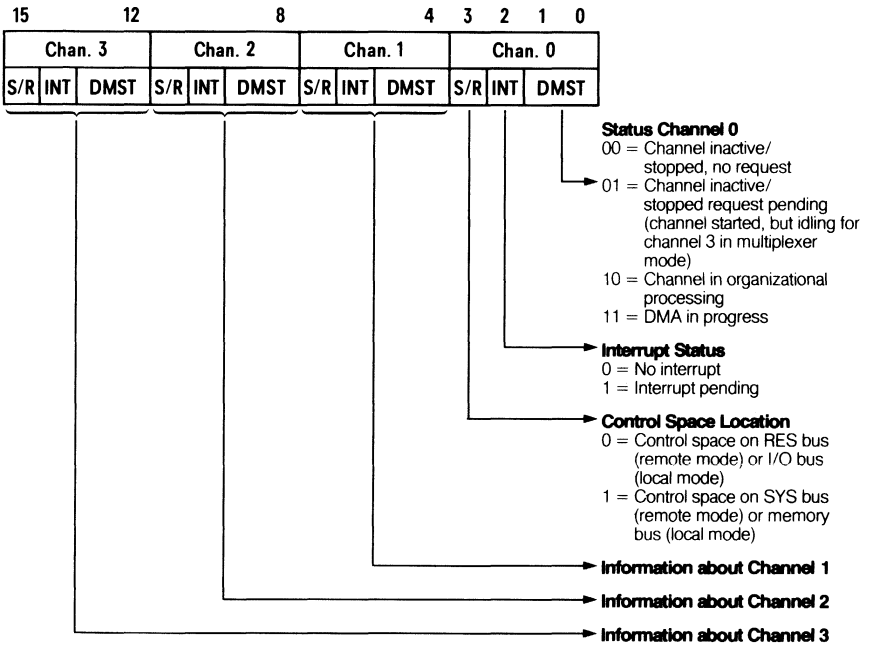
Determines Min. Number of Clock Cycles Between Burst Accesses (default after reset=0, i.e. 4 T-states delay)



General Status Register

The general status register GSR (figure below) shows the current states of all the channels.

General Status Register (GSR)



Channel Commands

The channel commands are contained in the channel command block. 15 bits are used to specify the command. There are two types of channel commands:

- Type 1: for data movement
- Type 2: for command chaining control

The command block for a type 1 command is 26 bytes long (see figure on page 17).

The type 1 command fields (see figure on page 27) contain information on:

a. Bus width of source and destination
b. If source and/or destination address should be incremented or decremented or kept constant during the transfer
c. If source/destination is in memory or I/O space (local mode) or in system or resident space (remote mode)
d. If data chaining (list or linked-list) is to be performed
e. If the data transfer is synchronized (source or destination).

Type 2 command blocks are 6 bytes long (see figure on page 28) of which the first 2 bytes form the command and the rest is either a relative displacement or an absolute address for the JUMP operation. There are two basic type 2 commands (see figure on page 28):

a. JUMP – conditional and non-conditional
b. STOP – conditional and non-conditional

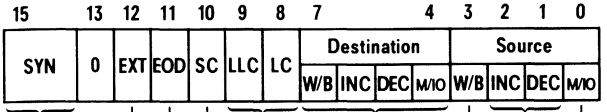
The conditional case tests for either of the 2 condition bits which are altered at the termination of any DMA operation:

- Termination due to byte count end
- Termination due to external terminate

It is thus possible to JUMP or STOP further execution of commands based on any of these conditions and optionally generate $\overline{\text{EOD}}$ or interrupt signal.

The combination of type 1 and 2 commands gives the SAB 82257 a high degree of "programmability". It can thus execute quite complex algorithms with a fairly low demand for CPU service.

Type 1 (DMA) Channel Command



Source Description

Associated Space
0 = I/O or resident
1 = Memory or system

Source Pointer
00 = Pointer not modified
01 = Decrement pointer
10 = Increment pointer
11 = No pointer (constant value)

Logical Bus Width
0 = 8-bit
1 = 16-bit

Destination Description
Same as source description

Data Chaining

LLC	LC	
0	0	No chaining
0	1	List chaining
1	0	Linked list chaining
1	1	Not allowed

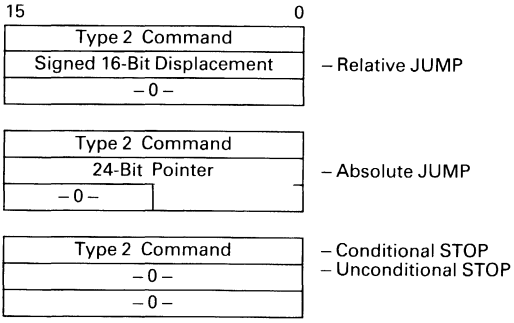
Select Chaining
0 = Destination data chaining
1 = Source data chaining

Enable $\overline{\text{EOD}}$ Output

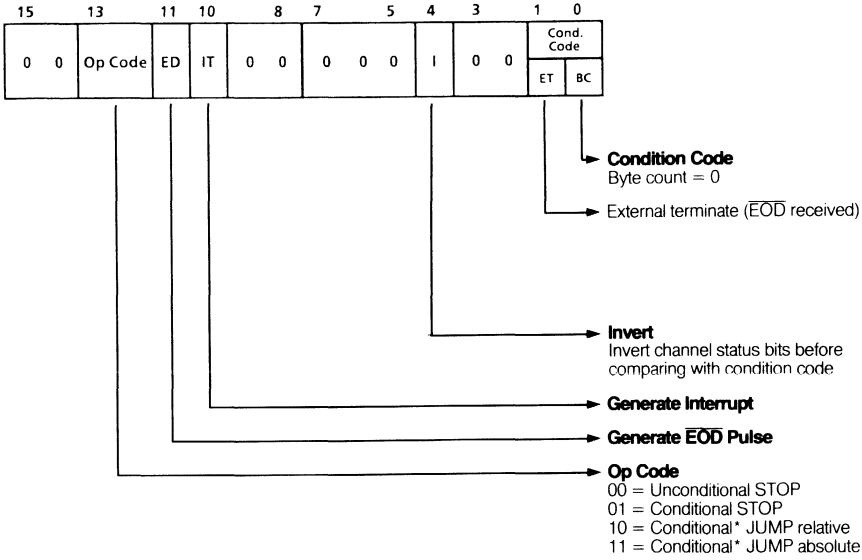
Enable External Terminate Input

Synchronization
00 = Not valid (type 2 command)
01 = Source synchronization
10 = Destination synchronization
11 = No synchronization (free running)

Type 2 Command Blocks (for command chaining control)



Type 2 Command Format

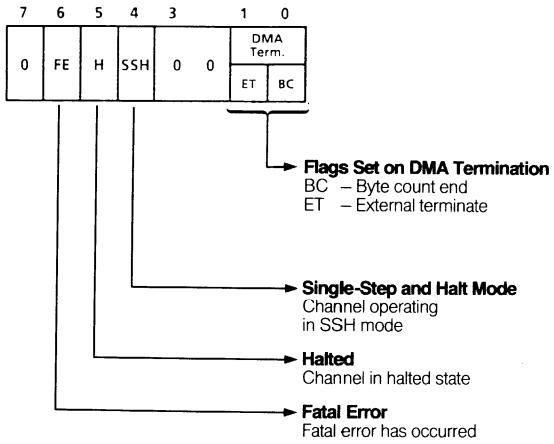


*) Unconditional JUMP when both condition code bits are set 1.

Channel Status Register

For each channel there is a channel status register (see figure below). This register shows the current state of the appropriate channel.

Channel Status Register



Timings

The bus timings in 286 and remote mode are identical to that for SAB 80286, in the 186 and 8086 mode the timings are identical to that for SAB 80186. For exact timings see timing diagrams of AC Characteristics.

Asynchronous control inputs are specified with setup and hold times which are only important to determine whether the SAB 82257 responds to the signal in the current cycle or the next cycle.

The following pages hold two sections of ac characteristics and waveforms. The first section refers to 286 mode and remote mode, the second one to 186 mode and 8086 mode.

Absolute Maximum Ratings [1]

Temperature under bias	0 to 70°C
Storage temperature	−65 to +150°C
Voltage on any pin with respect to ground	−0.5 to +7V
Power dissipation	3.6W

DC Characteristics [2]

TA = 0 to 70°C; TC = 0 to 100°C; VCC = +5V ±10%

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
VIL	Input low voltage (except CLK)	−0.5	+0.8	V	–
VIH	Input high voltage (except CLK)	2.0	VCC+0.5	V	–
VOL	Output low voltage	–	0.45	V	IOL = 3.0 mA
VOH	Output high voltage	2.4	–	V	IOH = −400 µA
ICC	Power supply current	–	450	mA	TA = 25°C, all outputs open
ILI	Input leakage current				
	$\overline{S0}$, $\overline{S1}$, $\overline{S2}$, $\overline{BHE}$, $\overline{RD}$, $\overline{WR}$, M/$\overline{IO}$	–	−200	µA	0 V ≤ VIN ≤ VCC
	HOLD ($\overline{RQ}$/$\overline{GT}$ mode), $\overline{EOD}$	–	1.5	mA	0 V ≤ VIN ≤ VCC
	A23 (AREADY), A21 [3]	–	−1.5	mA	0 V ≤ VIN ≤ VCC
	other pins	–	±10	µA	0 V ≤ VIN ≤ VCC
ILO	Output leakage current	–	±10	µA	0.45V ≤ VOUT ≤ VCC
VCL	Clock input low voltage	−0.5	+0.6	V	–
VCH	Clock input high voltage	3.8	VCC+1.0	V	–
CIN	Capacitance of inputs (except CLK)	–	10	pF	fC = 1 MHz
CIO	Capacitance of I/O or outputs	–	20	pF	fC = 1 MHz
CCLK	Capacitance of CLK input	–	12	pF	fC = 1 MHz

[1] Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

[2] Clock must be applied.

[3] This specification is valid only during RESET.

AC Characteristics SAB 82257 (286 mode)

TA = 0 to 70°C; TC = 0 to 100°C; VCC = +5V ±10%
Any output timing is measured at 1.5V.

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
1	CLK cycle period	62	250	ns	–
2	CLK low time	15	230	ns	at 1.0 V
3	CLK high time	20	235	ns	at 3.6 V
4	Address/control output delay	–	60	ns	CL = 100 pF
5	Status output delay	–	40	ns	CL = 100 pF
6	Sync data setup time	10	–	ns	–
7	Sync data hold time	5	–	ns	–
8	Sync $\overline{\text{READY}}$ setup time	38	–	ns	–
9	Sync $\overline{\text{READY}}$ hold time	25	–	ns	–
10	Sync control input setup time	20	–	ns	–
11	Sync control/address input hold time	20	–	ns	–
12	Sync address setup time	2.5	–	ns	–
13	Data/control output delay	–	50	ns	CL = 100 pF
14	Data/control float delay	–	50	ns	–
15	$\overline{\text{BHE}}$ setup time	60	–	ns	–
16	Write command width	4CLK+40	–	ns	–
17	Async data setup time	2CLK+30	–	ns	–
18	Async address setup time	20	–	ns	–
19	Async data access time	–	5CLK+70	ns	–

AC Characteristics SAB 82257 (286 mode; cont'd)

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
28	Mode select setup time	2CLK+20	–	ns	–
29	Mode select hold time	0	–	ns	–
33	Command recovery time	4CLK+40	–	ns	–
34	CLK rise time	–	15	ns	1.0 to 3.6V
35	CLK fall time	–	15	ns	3.6 to 1.0V
36	DREQ inactive after $\overline{\text{DACK}}$ active	0	–	ns	–
37	$\overline{\text{CS}}$ active response time	–	16CLK +80	ns	Note 1
39	$\overline{\text{CS}}$ active after BREL inactive	0	–	ns	–
42	HOLD active to HLDA active	0	–	ns	–
43	Async input setup time	20	–	ns	Note 2
44	Async input hold time	20	–	ns	Note 2
47	Async HLDA high time	2CLK+40	–	ns	Note 3
49	HOLD output low time	4CLK–50	–	ns	–
50	HLDA low to HOLD low delay	–	16CLK +70	ns	Note 1
53	Read command width	T19	–	ns	–
54	Async access setup time	20	–	ns	–
55	Async access hold time	20	–	ns	–

Note 1: If wait states are inserted, the maximum value has to be extended by the time required for the wait states for 3 bus cycles.

Note 2: These specifications are given for testing purposes only to assure recognition at a specific clock edge.

Note 3: This timing is valid if the signal is not synchronous, i.e. does not meet the specified setup and hold times.

AC Characteristics SAB 82257-6 (286 mode)

TA = 0 to 70°C; TC = 0 to 100°C; VCC = +5V ±10%
Any output timing is measured at 1.5V.

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
1	CLK cycle period	80	250	ns	–
2	CLK low time	20	225	ns	at 1.0 V
3	CLK high time	25	230	ns	at 3.6 V
4	Address/control output delay	–	75	ns	CL = 100 pF
5	Status output delay	–	55	ns	CL = 100 pF
6	Sync data setup time	20	–	ns	–
7	Sync data hold time	8	–	ns	–
8	Sync $\overline{\text{READY}}$ setup time	50	–	ns	–
9	Sync $\overline{\text{READY}}$ hold time	35	–	ns	–
10	Sync control input setup time	30	–	ns	–
11	Sync control/address input hold time	30	–	ns	–
12	Sync address setup time	3	–	ns	–
13	Data/control output delay	0	65	ns	CL = 100 pF
14	Data/control float delay	–	65	ns	–
15	$\overline{\text{BHE}}$ setup time	80	–	ns	–
16	Write command width	4CLK+40	–	ns	–
17	Async data setup time	2CLK+50	–	ns	–
18	Async address setup time	30	–	ns	–
19	Async data access time	–	5CLK+95	ns	–

AC Characteristics SAB 82257-6 (286 mode; cont'd)

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
28	Mode select setup time	2CLK+30	–	ns	–
29	Mode select hold time	0	–	ns	–
33	Command recovery time	4CLK+40	–	ns	–
34	CLK rise time	–	15	ns	1.0 to 3.6V
35	CLK fall time	–	15	ns	3.6 to 1.0V
36	DREQ inactive after $\overline{DACK}$ active	0	–	ns	–
37	$\overline{CS}$ active response time	–	16CLK +10	ns	Note 1
39	$\overline{CS}$ active after BREL inactive	0	–	ns	–
42	HOLD active to HLDA active	0	–	ns	–
43	Async input setup time	30	–	ns	Note 2
44	Async input hold time	30	–	ns	Note 2
47	Async HLDA high time	2CLK+60	–	ns	Note 3
49	HOLD output low time	4CLK–65	–	ns	–
50	HLDA low to HOLD low delay	–	16CLK +95	ns	Note 1
53	Read command width	T19	–	ns	–
54	Async access setup time	30	–	ns	–
55	Async access hold time	30	–	ns	–

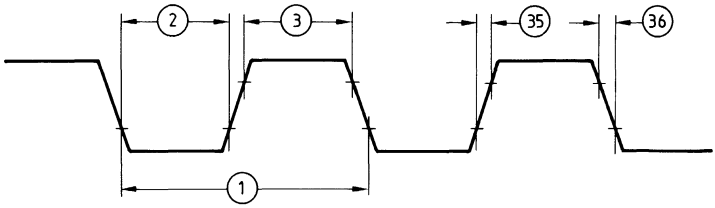
Note 1: If wait states are inserted, the maximum value has to be extended by the time required for the wait states for 3 bus cycles.

Note 2: These specifications are given for testing purposes only to assure recognition at a specific clock edge.

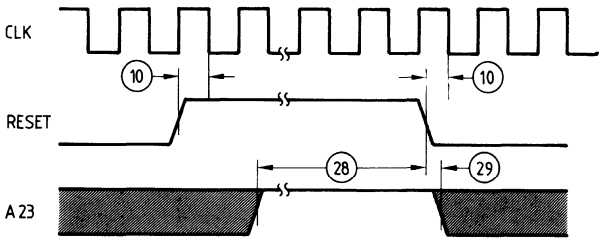
Note 3: This timing is valid if the signal is not synchronous, i.e. does not meet the specified setup and hold times.

Waveforms

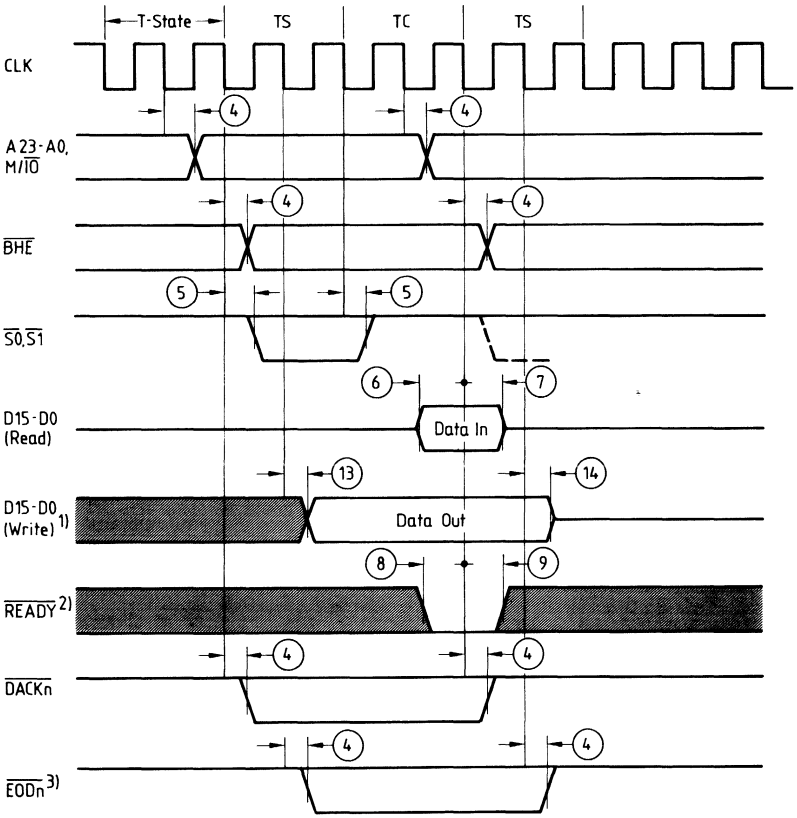
Clock Signal (286 mode)



Mode Selection on RESET (286 mode)



Major Timing for Active Bus Cycles (286 mode)

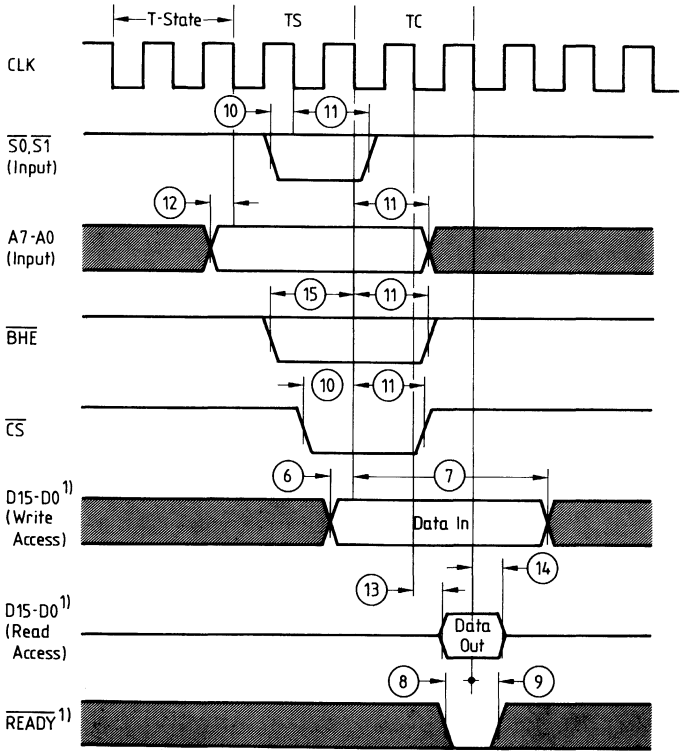


[1] If executing a single cycle transfer, D15 to D0 float like during read cycles!

[2] TC will be repeated if $\overline{\text{READY}}$ is inactive at the sampling point (end of current TC).

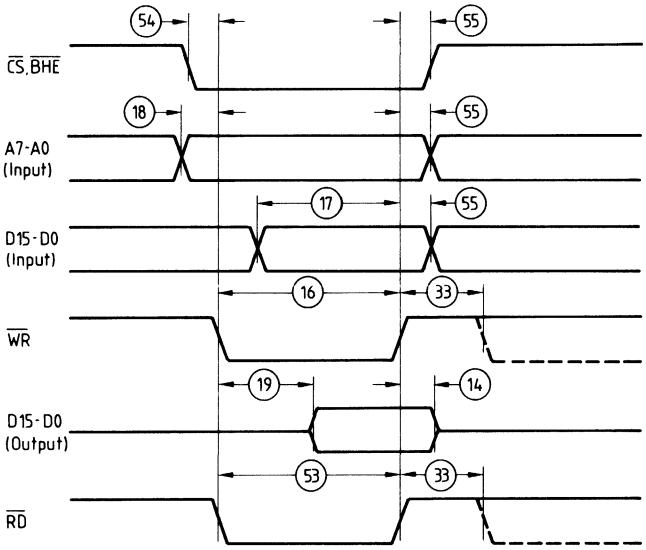
[3] Initiated by terminal count.

Synchronous Access (286 mode)

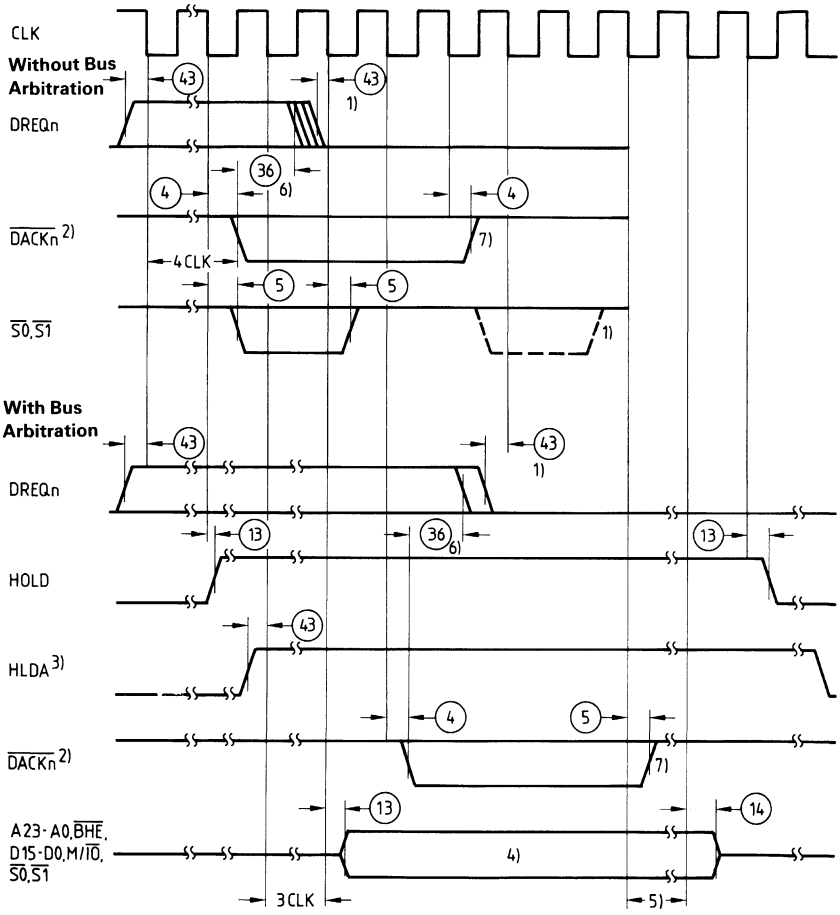


1) The processor will repeat TC, if $\overline{\text{READY}}$ is not active at the sampling point (end of current TC). The SAB 82257 will output data until the end of the repeated TC (read access) or sample the data bus again at the beginning of the repeated TC (write access).

Asynchronous Access (286 mode)



DMA Control (286 mode)



[1] If the trailing edge of DREQn is received later, a continuous request is assumed and subsequent transfers will be executed.

[2] Refers to the highest priority request. Acknowledging of lower priority requests may be delayed by the execution of higher priority requests.

[3] The SAB 82257 can be forced off the bus by driving HLDA inactive (see "Bus Arbitration").

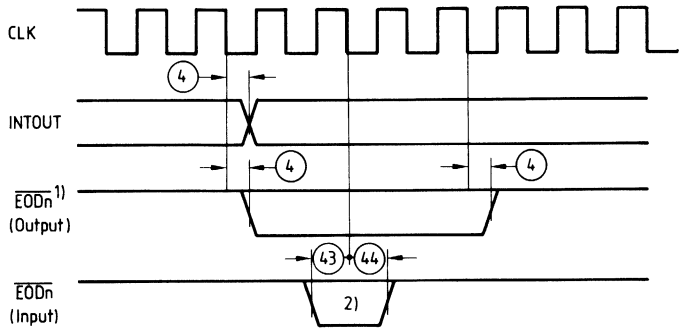
[4] Signals driven active. For exact timing refer to "Major Timing for Active Bus Cycles".

[5] The SAB 82257 may execute additional bus cycles, e.g. for command chaining.

[6] Minimum time to execute bus cycle.

[7] If the SAB 82257 does not perform subsequent bus cycles after this DMA cycle (transfer on another channel or organizational processing), the $\overline{\text{DACKn}}$ signal can be prolonged by two T-states.

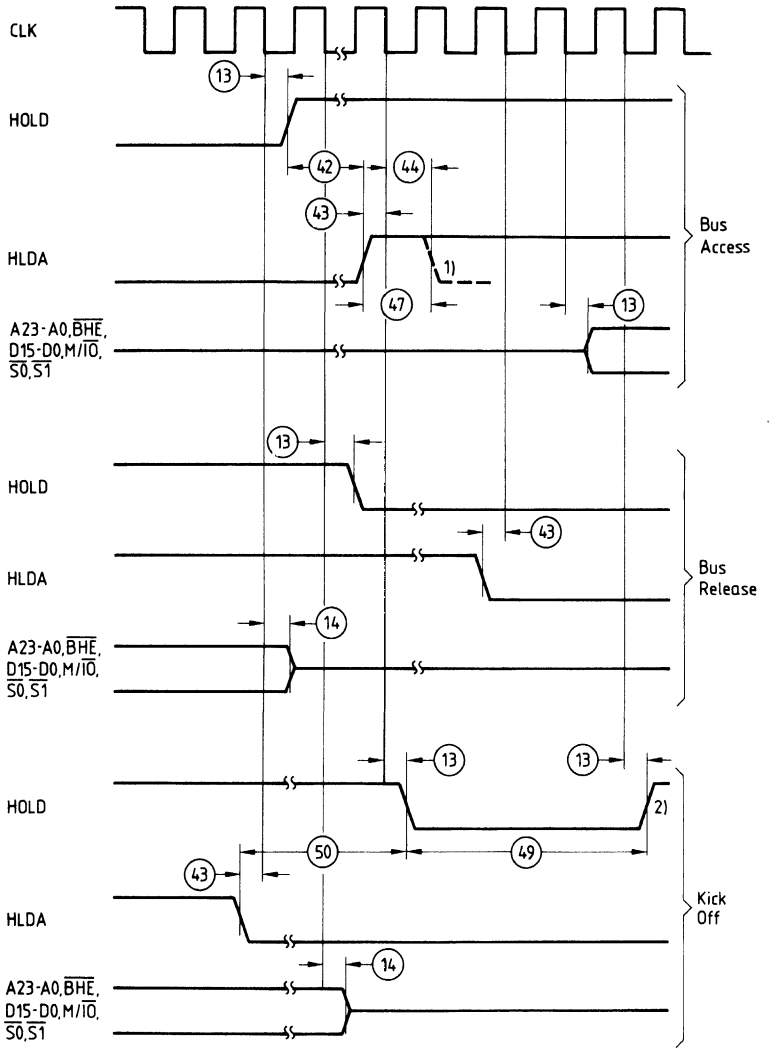
$\overline{\text{EOD}}$/INTOUT Timing (286 mode)



1) Initiated by type 2 command.

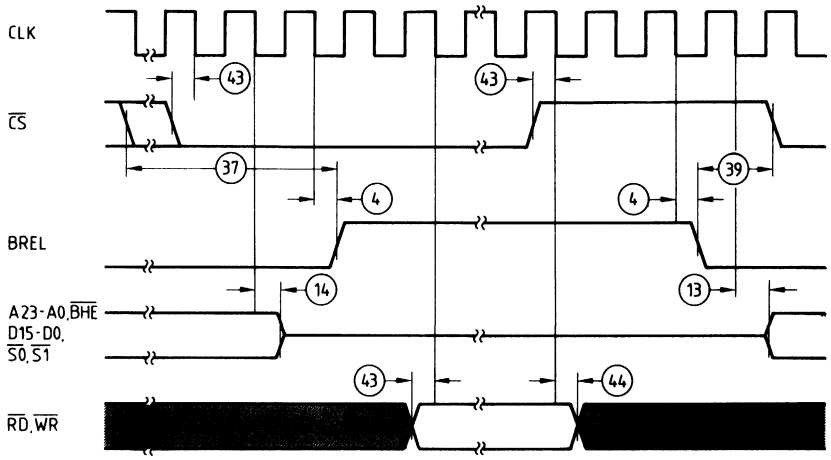
2) $\overline{\text{EOD}}$ input minimum pulse width is 3 CLKs if the signal is asynchronous.

Bus Arbitration (286 mode)



[1] Minimum HLDA high time before kick-off to respond to HOLD signal.
[2] Earliest possible reactivation of HOLD after deactivation of HLDA.

Access in Remote Mode



AC Characteristics SAB 82257 (186 mode)

TA = 0 to 70°C; TC = 0 to 100°C; VCC = +5V ±10%
Any output timing is measured at 1.5V.

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
4	Control output delay	–	60	ns	–
6	Sync address/data setup time	10	–	ns	–
7	Sync data hold time	5	–	ns	–
10	Sync control input setup time	20	–	ns	–
11	Sync control/address input hold time	20	–	ns	–
13	Data/control delay	–	50	ns	CL = 100 pF
14	Data float delay	–	50	ns	–
16	Write command width	2CLK+40	–	ns	–
17	Async data setup time	CLK+30	–	ns	–
18	Async address setup time	20	–	ns	–
19	Async data access time	–	2CLK +T22+70	ns	–
20	CLK cycle period	125	500	ns	–
21	CLK low time	55	–	ns	at 1.5V
22	CLK high time	55	–	ns	at 1.5V
23	CLK rise time	–	15	ns	1.0 to 3.5V
24	CLK fall time	–	15	ns	3.5 to 1.0V
25	AREADY active setup time	20	–	ns	Note 2
26	AREADY hold time	15	–	ns	Note 2
27	AREADY inactive setup time	35	–	ns	–

Note 2: These specifications are given for testing purposes only to assure recognition at a specific clock edge.

AC Characteristics SAB 82257 (186 mode; cont'd)

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
28	Mode select setup time	2CLK+20	–	ns	–
29	Mode select hold time	0	–	ns	–
30	Address/data output delay	10	50	ns	CL = 20 to 200 pF
31	Status output delay	10	55	ns	–
32	Float delay	10	50	ns	–
33	Command recovery time	2CLK+40	–	ns	–
36	DREQ inactive after $\overline{\text{DACK}}$ active	0	–	ns	–
38	ALE output delay	–	40	ns	–
40	Address/control input hold time	10	–	ns	–
41	Address input setup time	10	–	ns	–
42	HOLD active to HLDA active	0	–	ns	–
43	Async control input setup time	20	–	ns	Note 2
44	Async control input hold time	20	–	ns	Note 2
45	HLDA hold time	10	–	ns	–
46	Async HLDA high time	CLK+40	–	ns	Note 3
48	HOLD output delay	5	70	ns	–
51	HOLD output low time	2CLK−70	–	ns	–
52	HLDA low to HOLD low delay	–	12CLK +90	ns	Note 1
53	Read command width	T19	–	ns	–
54	Async access setup time	20	–	ns	–
55	Async access hold time	20	–	ns	–
56	ALE output delay	–	40	ns	–
57	SREADY hold time	15	–	ns	–

Note 1: If wait states are inserted, the maximum value has to be extended by the time required for the wait states for 2 bus cycles.

Note 2: These specifications are given for testing purposes only to assure recognition at a specific clock edge.

Note 3: This timing is valid, if the signal is not synchronous, i.e. does not meet the specified setup and hold times.

AC Characteristics SAB 82257-6 (186 mode)

TA = 0 to 70°C; TC = 0 to 100°C; VCC = +5V ±10%
Any output timing is measured at 1.5V.

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
4	Control output delay	–	75	ns	–
6	Sync address/data setup time	20	–	ns	–
7	Sync data hold time	8	–	ns	–
10	Sync control input setup time	25	–	ns	–
11	Sync control/address input hold time	25	–	ns	–
13	Data/control delay	0	65	ns	CL = 100 pF
14	Data float delay	–	80	ns	–
16	Write command width	2CLK+40	–	ns	–
17	Async data setup time	CLK+50	–	ns	–
18	Async address setup time	30	–	ns	–
19	Async data access time	–	2CLK +T22+85	ns	–
20	CLK cycle period	160	500	ns	–
21	CLK low time	75	–	ns	at 1.5V
22	CLK high time	75	–	ns	at 1.5V
23	CLK rise time	–	15	ns	1.0 to 3.5V
24	CLK fall time	–	15	ns	3.5 to 1.0V
25	AREADY active setup time	20	–	ns	Note 2
26	AREADY hold time	15	–	ns	Note 2
27	AREADY inactive setup time	35	–	ns	–

Note 2: These specifications are given for testing purposes only to assure recognition at a specific clock edge.

AC Characteristics SAB 82257-6 (186 mode; cont'd)

Symbol	Parameter	Limit values		Unit	Test condition
		min.	max.		
28	Mode select setup time	2CLK+30	–	ns	–
29	Mode select hold time	0	–	ns	–
30	Address/data output delay	10	55	ns	CL = 20 to 200 pF
31	Status output delay	10	75	ns	–
32	Float delay	10	55	ns	–
33	Command recovery time	2CLK+40	–	ns	–
36	DREQ inactive after $\overline{\text{DACK}}$ active	0	–	ns	–
38	ALE output delay	–	50	ns	–
40	Address/control input hold time	15	–	ns	–
41	Address input setup time	15	–	ns	–
42	HOLD active to HLDA active	0	–	ns	–
43	Async control input setup time	30	–	ns	Note 2
44	Async control input hold time	30	–	ns	Note 2
45	HLDA hold time	10	–	ns	–
46	Async HLDA high time	CLK+60	–	ns	Note 3
48	HOLD output delay	5	90	ns	–
51	HOLD output low time	2CLK−90	–	ns	–
52	HLDA low to HOLD low delay	–	12CLK +120	ns	Note 1
53	Read command width	T19	–	ns	–
54	Async access setup time	30	–	ns	–
55	Async access hold time	30	–	ns	–
56	ALE output delay	–	55	ns	–
57	SREADY hold time	15	–	ns	–

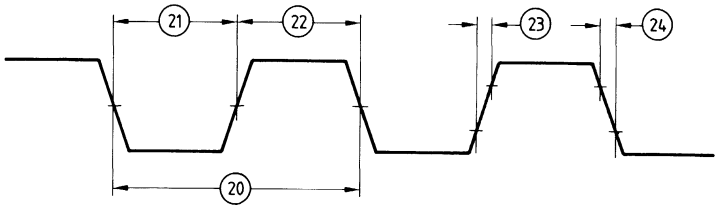
Note 1: If wait states are inserted, the maximum value has to be extended by the time required for the wait states for 2 bus cycles.

Note 2: These specifications are given for testing purposes only to assure recognition at a specific clock edge.

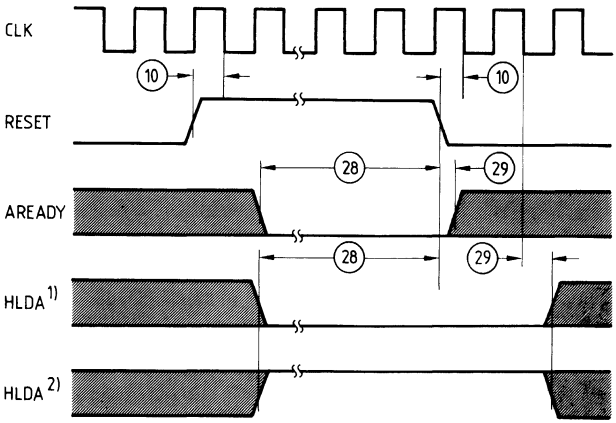
Note 3: This timing is valid, if the signal is not synchronous, i.e. does not meet the specified setup and hold times.

Waveforms

Clock Signal (186 mode)

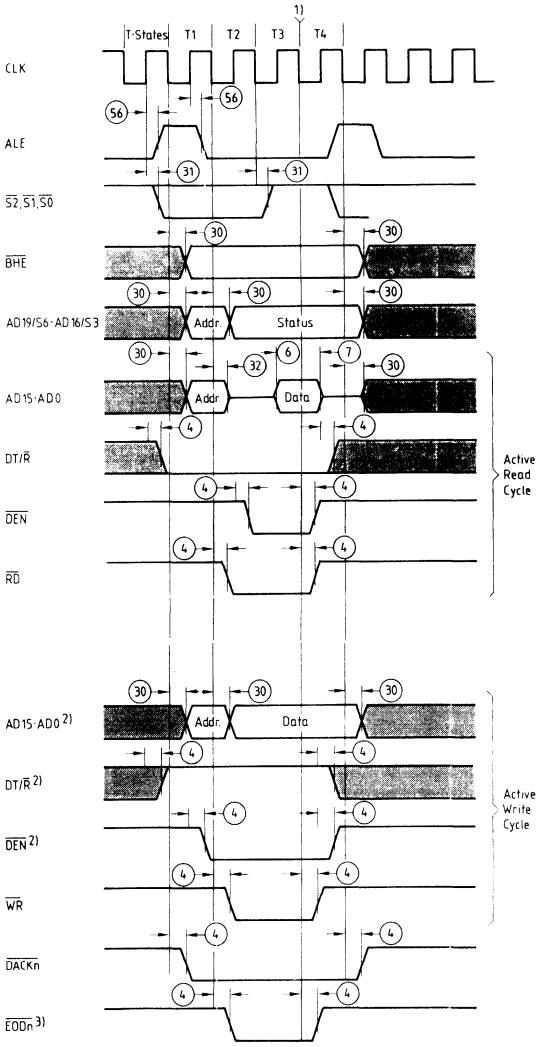


Mode Selection on RESET (186 mode)



1) To operate in 186 mode with HOLD/HLDA protocol.
2) To operate in 8086 mode with $\overline{RQ}/\overline{GT}$ protocol.

Major Timing for Active Bus Cycles (186 mode)

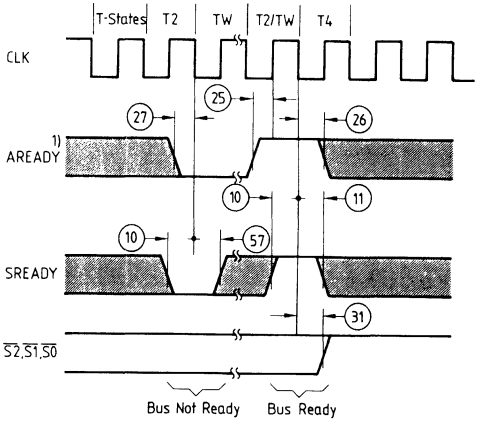


[1] A wait state is inserted after T3 or TW, whenever the bus is not ready at the beginning of T3 or TW (see "Bus Cycle Termination"). The status must be valid just prior to T4.

[2] For a single-cycle transfer the timing of AD15-AD0, DT/$\overline{R}$ and $\overline{DEN}$ is identical to a read cycle. AD15-AD0 will float as during a read cycle.

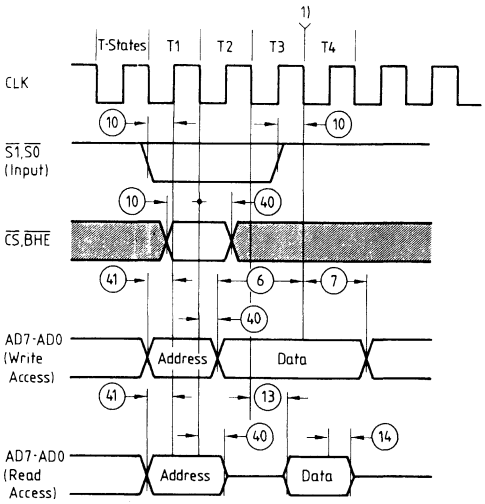
[3] Initiated by terminal count.

Bus Cycle Termination (186 mode)



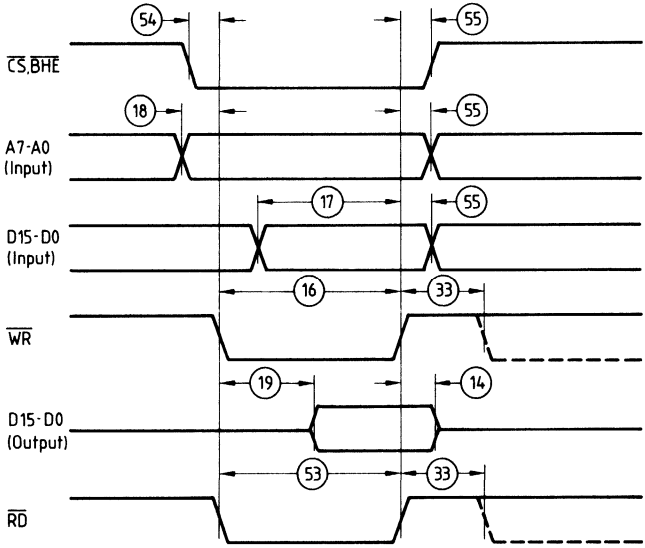
[1] Only the rising edge of AREADY is synchronized internally to CLK. The falling edge must be synchronized externally.

Synchronous Access (186 mode)

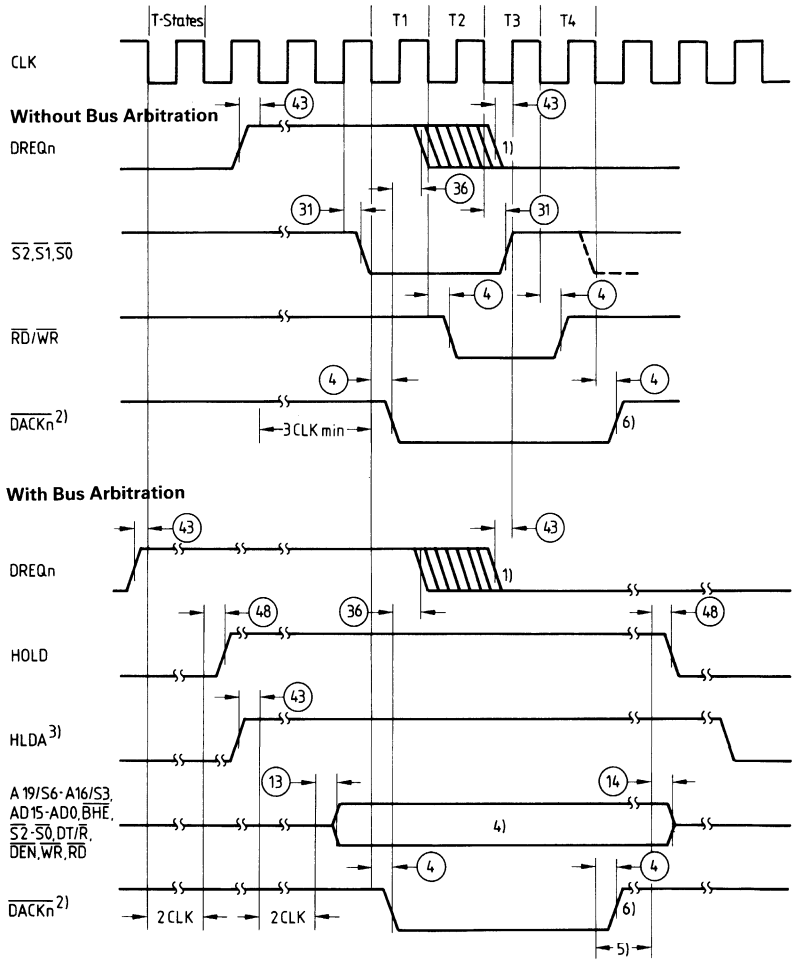


[1] Additional wait cycles may be inserted. Status must be valid just prior to T4.

Asynchronous Access (186 mode)



DMA Control (186 mode)



1) If the trailing edge DREQn is received later, a continuous request is assumed and subsequent transfer will be executed.

2) Refers to the highest priority request. Acknowledging of lower priority requests may be delayed by the execution of higher priority requests.

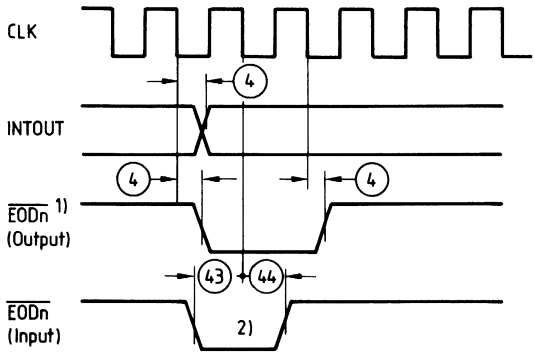
3) The SAB 82257 can be forced off the bus by driving HLDA inactive (see "Bus Arbitration").

4) Signals driven active. For exact timing refer to "Major Timing for Active Bus Cycles".

5) The SAB 82257 may execute additional bus cycles, e.g. for command chaining.

6) If the SAB 82257 does not perform subsequent bus cycles after this DMA cycle (transfer on another channel or organizational processing), the DACKn signal can be prolonged by two T-states.

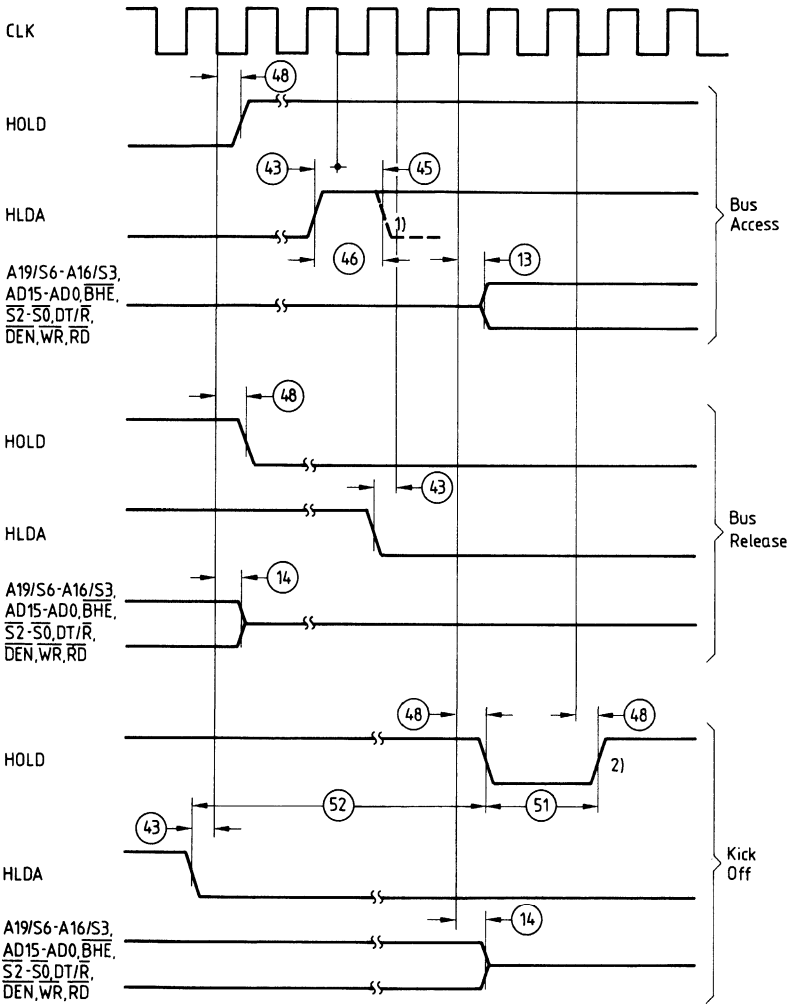
$\overline{\text{EOD}}$/INTOUT Timing (186 mode)



[1] Initiated by type 2 command.

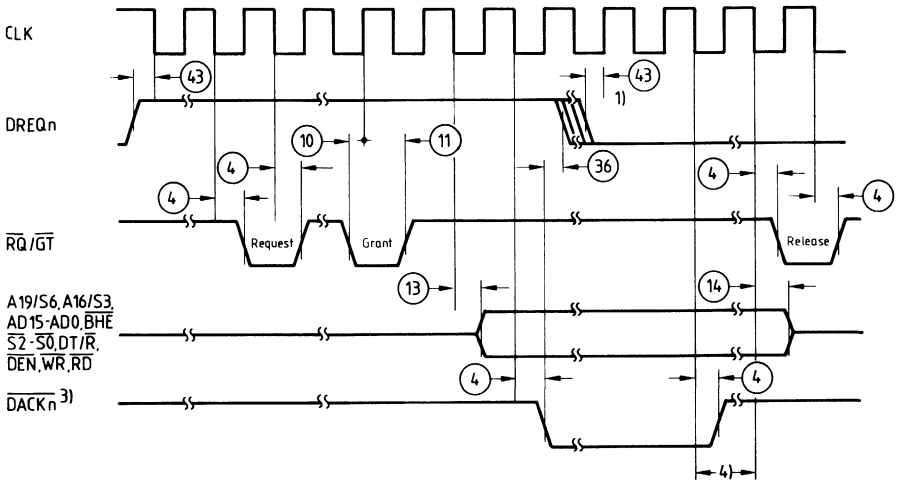
[2] $\overline{\text{EOD}}$ input minimum pulse width is 2CLKs if the signal is asynchronous.

Bus Arbitration (186 mode)



[1] Minimum HLDA high time before kick-off to respond to HOLD signal.
[2] Earliest possible reactivation of HOLD after deactivation of HLDA.

DMA Control with $\overline{RQ}/\overline{GT}$ Protocol (8086 mode)



[1] If the trailing edge DREQn is received later, a continuous request is assumed and subsequent bus cycles are executed.

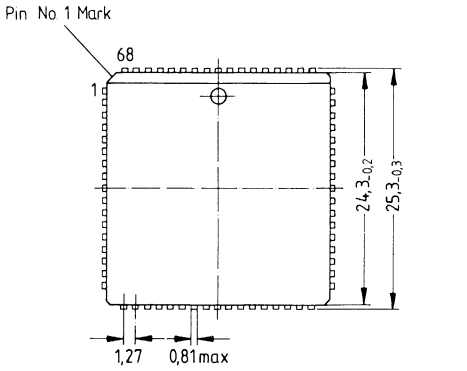
[2] Signals driven active. For exact timing refer to "Major Timing for Active Bus Cycles".

[3] Refers to the highest priority request. Acknowledge of lower priority requests may be delayed by higher priority requests.

[4] The SAB 82257 may execute additional bus cycles, e.g. for command chaining.

Package Outline

Plastic Package, Leaded Chip Carrier, PLCC 68



Dimensions in mm

Ordering Information

Type	Description	Ordering code
SAB 82257-N	High-performance DMA controller, 8 MHz	Q67120-P176
SAB 82257-6-N	High-performance DMA controller, 6 MHz	Q67120-P179

Siemens Worldwide (Addresses)

Siemens AG, Bereich Bauelemente
Balanstraße 73, Postfach 801709, **D-8000 München 80**
☎ (089) 4144-0 Tx 52108-0 FAX (089) 4144-2689

Siemens Worldwide

Federal Republic of Germany and Berlin (West)

Siemens AG
Salzufer 6–8
1000 Berlin 10
☎ (030) 3939-1, Tx 1810-278
FAX (030) 3939-2630
Ttx 308190 = sieznb

Siemens AG
Contrescarpe 72
Postfach 107827
2800 Bremen
☎ (0421) 364-0, Tx 245451
FAX (0421) 364-2687

Siemens AG
Lahnweg 10
Postfach 1115
4000 Düsseldorf 1
☎ (0211) 399-0, Tx 8581301
FAX (0211) 399-2506

Siemens AG
Rödelheimer Landstraße 5–9
Postfach 111733
6000 Frankfurt 1
☎ (069) 797-0, Tx 414131
FAX (069) 797-2253

Siemens AG
Lindenplatz 2
Postfach 105609
2000 Hamburg 1
☎ (040) 282-1, Tx 215584-0
FAX (040) 282-2210

Siemens AG
Am Maschpark 1
Postfach 5329
3000 Hannover 1
☎ (0511) 129-0, Tx 922333
FAX (0511) 129-2799

Siemens AG
Richard-Strauss-Straße 76
Postfach 202109
8000 München
☎ (089) 9221-0
Tx 529421-19
FAX (089) 9221-4390

Siemens AG
Von-der-Tann-Straße 30
Postfach 4844
8500 Nürnberg 1
☎ (0911) 654-0, Tx 622251
FAX (0911) 654-3436, 3464

Siemens AG
Geschwister-Scholl-Straße 24
Postfach 120
7000 Stuttgart 1
☎ (0711) 2076-1, Tx 723941
FAX (0711) 2076-706

EUROPE

Austria
Siemens Aktiengesellschaft
Österreich
Postfach 326
A-1031 Wien
☎ (0222) 7293-0, Tx 1372-0

Belgium
Siemens S.A.
chaussée de Charleroi 116
B-1060 Bruxelles
☎ (02) 536-2111, Tx 21347

Denmark
Siemens A/S
Borupvang 3
DK-2750 Ballerup
☎ (02) 656565, Tx 35313

Finland
Siemens Osakeyhtiö
Fach 8
SF-00101 Helsinki 10
☎ (0) 1626-1, Tx 124465

France
Siemens S.A.
B.P. 109
F-93203 Saint-Denis CEDEX 1
☎ (1) 8206120, Tx 620853

Great Britain
Siemens Ltd.
Siemens House
Windmill Road
GB-Sunbury-on-Thames
Middlesex TW 16 7HS
☎ (09327) 85691, Tx 8951091

Greece
Siemens AE
Voulis 7
P.O.B. 3601
GR-10247 Athen
☎ (01) 3293-1, Tx 216291

Ireland
Siemens Ltd.
Unit 8–11 Slaney Road
Dublin Industrial Estate
Finglas Road
Dublin 11
☎ (01) 302855, Tx 24129

Italy
Siemens Elettra S.p.A.
Via Fabio Filzi, 29
Casella Postale 10388
I-20100 Milano
☎ (02) 67661, Tx 330261

Netherlands
Siemens Nederland N.V.
Postb. 16068
NL-2500 BB Den Haag
☎ (070) 782782, Tx 31373

Norway
Siemens A/S
Østre Aker vei 90
Postboks 10, Veitvet
N-0518 Oslo 5
☎ (02) 153090, Tx 18477

Portugal
Siemens S.A.R.L.
Avenida Almirante Reis, 65
Apartado 1380
P-1100 Lisboa-1
☎ (01) 538805, Tx 12563

Spain
Siemens S.A.
Orense, 2
Apartado 155
E-28080 Madrid
☎ (01) 4552500, Tx 43320

Sweden
Siemens AB
Hälsingegatan 40
Box 23141
S-10435 Stockholm
☎ (08) 161-100, Tx 19880

Switzerland
Siemens-Albis AG
Freilagerstraße 28
Postfach
CH-8047 Zürich
☎ (01) 495-3111, Tx 558911

Turkey
ETMAŞ Elektrik Tesisati ve
Mühendislik A.Ş.
Meclisi Mebusan Caddesi 55/35
Findikli
P.K. 1001 Karakoey
Istanbul
☎ 009011/452090, Tx 24233

AFRICA

South African Republic
Siemens Limited
Siemens House,
P.O.B. 4583
2000 Johannesburg
☎ (011) 7159111, Tx 4-22524

AMERICA

Argentina
Siemens S.A.
Avenida Pte. Julio A. Roca 516
Casilla Correo Central 1232
RA-1000 Buenos Aires
☎ (01) 00541/300411, Tx 21812

Brazil
Siemens S.A.
Sede Central
Caixa Postal 1375,
01051 São Paulo-SP
☎ (011) 833-2211
Tx 11-23641

Canada
Siemens Electric Limited
7300 Trans-Canada Highway
P.O.B. 7300, Pointe Claire,
Québec H9R 4R6
☎ (514) 6957300,
Tx 05822778

U.S.A.
Power semiconductors:
Siemens Components, Inc.
Colorado Components Division
800 Hoyt Street
Broomfield, Colorado 80020
☎ (303) 469-2161
Tx 454357 sie colo

Intelligent displays:
Siemens Components, Inc.
Optoelectronic Division
19000 Homestead Road
Cupertino, California 95014
☎ (408) 257-7910
Tx 352084 sie lit opto

All other products:
Siemens Components, Inc.
Special Electronics Division
186 Wood Avenue South
Iselin, New Jersey 08830
☎ (201) 321-3400
Tx 844491

ASIA

Hongkong
Jebsen & Co., Ltd.
Siemens Division
P.O.B. 97
Hongkong
☎ (05) 8233777, Tx 73221

India
Siemens India Ltd.
Head Office
134-A, Dr. Annie Besant Road,
Worli
P.O.B. 6597
Bombay 400018
☎ 4938786, Tx 75142

Japan
Fuji Electronic Components Ltd.
New Yurakucho Bldg., 8F
12-1, Yurakucho 1-Chome,
Chiyoda-ku
Tokyo 100
☎ (03) 201-2401, Tx 32182

Korea
Siemens Electrical
Engineering Co., Ltd.
C.P.O.B. 3001
Seoul
☎ (02) 275-6111, Tx 23229

Singapore
Siemens Components Pte. Ltd.
Promotion Office
10-15 E, 5th floor
47 Ayer Rajah Crescent No. 06-12
Singapore 0513
☎ 7760044, Tx RS 21000

Taiwan
TAI Engineering Co. Ltd.
6th Floor Central Building
108, Chung Shan N. Rd. Sec. 2
P.O.Box 68-1882
Taipei
☎ 5363171, Tx 27860

AUSTRALIA

Siemens Ltd.
544 Church Street, Richmond
Melbourne, Vic. 3121
☎ (03) 4207111, Tx 30425