PATENT SPECIFICATION (11) 1 515 163

1 515 163

(21) Application No. 37375/75 (22) Filed 11 Sep. 1975

(31) Convention Application No. 27258 (32) Filed 13 Sep. 1974 in

(33) Italy (IT)

(44) Complete Specification Published 21 Jun. 1978

(51) INT. CL.[2] G06F 3/00

(52) Index at Acceptance
G4A 12P 16EY 17P 3W2 EX

(72) Inventor: Angelo Bardotti

(19)

(54) COMPUTER INTERFACE SYSTEM WITH FAULT DETECTION

(71) We, HONEYWELL INFORMATION SYSTEMS ITALIA SpA, an Italian company of Caluso, Turin, Italy, do hereby declare the invention for which we pray that a patent may be granted to us, and the method by which it is to be performed, to be particularly described in and by the following statement:–

This invention relates to interfaces for connecting peripheral units to a central processor and in particular relates to interfaces of the kind described in our prior application no. 20774/75, (Serial Specification No. 1512263) and defined there as an "integrated" interface. More particularly the present invention relates to a mechanism for signalling and discriminating among different kind of errors which may occur in such interfaces.

It is known that in designing modern data processing systems, diagnostic ability and information protection criteria have taken a primary role. In designing a data processing system it is desirable for the system itself to be able to detect and locate with sufficient precision the causes of errors and be able to remedy them as far as possible, so minimizing the need for external manual intervention. At the same time it is desirable for the system to be able to discriminate those errors which do not hamper the execution of subsequent operation from those which may cause wrong operation in the execution of subsequent tasks or the damage or irreparable loss of information.

It is also known that modern data processing systems comprise a central unit connected to a plurality of peripheral units by means of interconnecting channels each composed of a set of wires on which electrical signals representing information are transferred following suitable procedures. The set of such wires from a physical point of view together with the set of logical signals transferred thereon is commonly termed an "interface".

In modern data processing systems, the central unit is often used, not only for detecting internal errors, but also for commanding diagnostic procedures which enable the identification of faults and errors inside the peripheral units. For this purpose it is essential to be sure the communication interface is operating in the right way. Unfortunately the interface too is a possible error source in the information exchange.

Those errors may be due to two fundamental causes:– electrical disturbances and electrical faults. Independently from the cause which has generated an error, the error may have two effects. If it occurs during the transfer of control information, it hampers in irreparable way the execution of the following operations and requires the interruption of the operations being executed and the launching of suitable diagnostic or allerting procedures. If it occurs during a data transfer it hampers only the transferred data and does not have any effect in general on the following operations, which may be completed provided no errors occur during the transfer of control information.

An object of the present invention is to provide a communication interface having mechanisms for signalling and discriminating between errors so that it can be determined whether the detected error must be attributed to the interface or is an internal error to be attributed to the central unit or the peripheral equipment and whether the error occurred during a data transfer or during a control information transfer, thus enabling the performance in the various cases of the most suitable procedures for diagnosing or recovering the errors.

Accordingly the invention provides a data processing system comprising a central processor and a peripheral unit connected by an interface which comprises an information bus and signal lines and on which information transfer is performed by the passage of control information from the central processor followed by the passage of command/service information or data information, the central

29 JUL 1981

processor and the peripheral unit each having a parity generating network for generating a parity bit for informating being placed on the information bus and a parity checking network for checking the parity of information being received from the information bus, the peripheral unit also having a logic checking circuit for identifying logical errors in at least the command information, error signalling means in the peripheral unit for signalling over the signal lines an error signal if an error is detected in information received over the information bus and for further signalling a parity fault if the error is a parity error, and error monitoring means in the central processor for determining, from the parity checking network in the central processor and the error signal and parity fault signal (if any) the occurrence of an error and whether the error is or is not one prejudicial to the operations which are to follow immediately.

In the preferred embodiment, any communication over the interface consists of the transfer of a single control word from the central processing unit, and which controls the interface itself, followed by either command/service information or data information. Command information is information which controls the peripheral unit, while service information is information sent from the peripheral unit to the central processing unit (in response to the preceding control word) to inform the central processing unit of the state of the peripheral unit. Logical checks are made on command information, and also in some cases on data information; these logical checks are made in general by fixed logic. A distinction can be made between errors in the control word, prejudicial to the interface itself, and errors in command/service information, prejudicial to the operation of the peripheral unit fed by the interface; both these errors however are prejudicial to subsequent operations and this distinction is not of great significance from an operational point of view.

A data processing system embodying the invention will now be described by way of example, with reference to the accompanying drawings in which:

Figure 1 is a block diagram of a computer system;

Figure 2 is a block diagram of the control unit of Figure 1;

Figure 3 is a block diagram of the logical unit of Figure 1;

Figure 4 is a block diagram of the main memory of Figure 1;

Figure 5 is a drawing of integrated interface of Figure 1;

Figure 6 is a timing diagram showing a write operation without error on the integrated interface;

Figure 7 is a timing diagram showing a write with error operation on the integrated interface;

Figure 8 is a timing diagram showing a read without error operation on the integrated interface;

Figure 9 is a timing diagram showing a read or write operation with error in the control information;

Figures 10 to 13 are block diagrams showing the integrated interface circuits in the central unit; and

Figures 14, 15 and 16 are block diagrams showing the interface circuits contained in a peripheral device adapter.

General Structure Of Computer System

Figure 1 illustrates a computer system which comprises a central processor 1, a main memory 2, and a plurality of peripheral units 4 to 13. The central processor 1 is functionally divided into a logical unit 3A, an input/output (I/O) control unit 3 and a control unit 3B. Peripheral units 4 and 5 are connected to the central processor by means of a peripheral control unit (PCU) 14 which communicates with the central processor through an information exchange channel or interface 15 and a group of interface circuits or interface module 16. Module 16, in turn, is connected to a port 15A of I/O control unit 3. Channel 15 is used as an exchange channel for both of peripheral units 4 and 5. Since channel 15 communicates with peripheral units 4 and 5 on a time multiplex basis, the channel may be thought of as two subchannels, each of which communicates separately with peripheral units 4 and 5 at different points in time. In this example, two subchannels correspond to channel 15 and connect two peripheral units 4 and 5.

Central processor 1 has a plurality of interfaces including plugs, channels and subchannels. For instance, channels 15, 17, 18 and 19 are connected to peripheral control units (PCU's) 14, 20, 21 and 22, respectively, through interface modules 16, 23, 24 and 25 and ports 15A, 17A, 18A and 19A, respectively. Peripheral units 4 and 5 are connected to PCU 14 by channels 26 and 27 respectively, and peripheral unit 6 is connected to the peripheral control unit 20 by a channel 28. Peripheral units 7 to 9 are connected to peripheral control unit 21 by a common channel 29, and peripheral units 10 to 13 are connected to PCU 22 by channels 30 to 33, respectively.

During a transfer of data between a peripheral unit and main memory 2 or central processor 1, a communication path must be established within the central processor and within the corresponding PCU or device adapter. The communication path is provided on one side by the input/output control unit 3 and on the other side by the PCU involved. An example of an I/O control unit is more fully described in our prior application no. 55286/73 (Specification No. 1,423,674) and will not be described here since it is not relevant for present purposes.

Figure 2 schematically illustrates control unit

3B of central processor 1. Control unit 3B basically comprises a control assembly 40 and a timing unit 42.

Timing unit 42 comprises a timing circuit 44 and a timing network 45. The timing circuit may consist of a delay line having a plurality of intermediate taps fed by a pulse generator, such as a one shot, controlled by an external START signal or controlled by a feedback signal from the delay line itself, so as to cyclically generate a sequence of timing pulses. Alternatively, an oscillator may be used to produce a base frequency which is frequency divided to obtain a sequence of timing pulses. The operation of the timing circuit may be arrested by means of a STOP command. Some of the timing pulses are transferred through outputs $T_1 \ldots\ldots T_N$ to several points of the computer in order to control the opening of certain AND gates. Other timing pulses are applied to timing network 45 over conductors CT1 to CTN.

Timing network 45 receives from the control assembly 40 a plurality of microcommands over conductors C1 ..... CN. By means of a plurality of AND gates (and, if required, flip-flop or other gates such as NOR-NAND. gates) controlled by timing pulses timing network 45 releases a plurality of timed microcommands on conductors TC1 to TCN. The microcommands have a suitable duration and time relation to each other, and are distributed into the computer, including the control unit.

In the drawings, a circle represents a gate for performing a logical switching function and an arrow drawn to a circle, if not otherwise specified, represents a gate input for receiving a suitable timing signal from conductors $T_1 \ldots\ldots T_N$ or a suitable microcommand from conductors TC1 to TCN.

Control unit 40 comprises a read only storage (ROS) 46, an output register (ROR) 47, a decoding network 48, an address register (ROSAR) 49 for ROS 46, an auxiliary address register (AROSAR) 50 and a counting network 51. ROS 46 contains 20 bit microwords organized into microprograms which control the operation of the computer and are called out by means of program instructions. Circles 59-66 represent several sets of AND gates which control the transfer of signals through the communication channels associated with each set. Each set receives at its input and transfers to its output (when enabled) a plurality of suitable signals on a group of wires represented by a single line and defined as a channel.

On channel 58, which is not conditioned by any AND gate, a set of signals representative of a pre-established ROS address may be loaded in a binary form. This address may be obtained by an initialization forcing through preset of suitable computer console keys (not shown) and by pushing a start button may start timing circuit 44 at the same time. The address on channel 58 is then loaded in ROSAR 49 and, as soon as gate set 62 is enabled by a timing pulse, ROS 46 is addressed and a microword is read out. The microword is transferred through gate set 66 and is loaded in ROR 47. The same microword now present at the outputs of ROR 47 is transferred through gates set 66A, enabled by $\overline{NOP}$ signal (which be discussed later) and decoded by decoding network 48. The decoded signals are transferred to timing network 45. Timing network 45 converts the signals to timed microcommands.

The ROS address contained in ROSAR 49 may be updated in several ways. It may be loaded through gate set 63 into AROSAR 50 and through gate set 64 into the input of counting network 51 where it may be incremented by 1, the updated address then being reloaded into ROSAR 49 through a channel 67 and gate set 61. Alternatively, the address may be incremented by a suitably quantity K obtained from the microword contained in ROR 47 and applied to network 51 through a channel 68 and gate set 65. The new address may be obtained from the microword contained in ROR 47; for example, a certain number of bits may be transferred through channel 68 and gate set 60 to ROSAR 49. A further addressing alternative is through a channel 69, controlled by gate set 59, which allows the loading of ROSAR 49 with the contents of an A register, one of the many working registers within the operating section of the computer. Since the contents of the A register may come from other registers of the operating section, as well as from peripheral or external units, or from main memory 2, ROS 46 ultimately may be addressed by using any data source within the computer or any data source connected to the computer.

It is useful to note that address loading of ROS and AROSAR may be performed also through channel 58A. When suitable error detecting networks, some of which will be considered below, recognize the occurrence of an error, a signal MCK (machine check) may be generated together with a signal NOP. The inverted signal $\overline{NOP}$ inhibits gate set 66A so that the generation of most timed microcommands is inhibited. At the same time, signal MCK generates a fixed ROS address which is applied to channel 58A. The fixed address produced by MCK may point to a ROS microprogram having diagnostic function so as to identify the nature of the error and, in some cases, the causes of the same. The first part of the microprogram checks that ROS is operating correctly, and once this is verified it removes signal NOP thus enabling the microprogram to proceed so as to identify errors external to ROS. This point will be considered again, when discussing in particular the errors detected through the interface.

The parallelism of control assembly 40 follows conventional techniques. The length of the ROS addresses, expressed in bits, is related to the number of memory positions which have

to be addressed, and is independent of the length of the microwords read out of ROS 46. For instance, if ROS 46 has 64,000 memory positions, the addresses have a length of 16 bits so that 2^{16} or 64,000 addresses can be formed. This means that the channels used to load the ROS address will consist of 16 wires and registers 49 and 50 will have 16 memory cells. ROR 47 will have the same number of cells as the number of bits in a microword, e.g., 20 bits. When the ROS address is obtained from a microword contained in ROR 47, only 16 out of 20 bits are used as an address and transferred on channel 68.

Referring to Figure 3, logic unit 3A of the computer basically comprises register banks 71 and 72, a working register A and a logic network 73. Register bank 71 comprises 16 registers, each having 16 bits and being addressed by a microcommand code through inputs 77. Following the addressing, information coming from a channel 75 may be loaded in the addressed register under the control of a gate set 74. Alternatively or jointly, the information contained or loaded in the addressed register may also be stored in output register A under the control of a gate set 74A. The contents of register A is fed through a distributor channel 70 and selectively controlled AND gate sets 91 to logic network 73.

The contents of register A may also be forwarded through AND gate set 82 to register bank 72. Register bank 72 comprises 16 registers each having 16 bits and being addressed by a microcommand code through inputs 83. The content of the addressed register is always present on a distributor channel 80 so that it is seen from the exterior as a single register B whose content may be selectively fed to operative network 73 through AND gate sets 95 to 98.

The register addressed within bank register 72 may be loaded by information coming from many sources. As described previously, data contained in register A may be transferred to bank register 72 through channel 81 and gate set 82. Another source is channel 78 coming from a peripheral unit and controlled by a gate set 76. Data coming from main memory 2 on a channel 118 (see Figure 4) is transferred to bank register 72 through a gate set 86. Data from the microprogramming memory (ROS) 46 may be forwarded to bank register 72 on a channel 84 (Figure 2) that is controlled by a gate set 88. It is also possible to load desired information into register bank 72 through a channel 89 controlled by a gate set 90. Channel 89 may be fed by information obtained through console keys actuated by an operator of the computer. In addition, particular conditions representative of the status of I/O channels and received from channel registers (not shown) may be loaded into register bank 72 through channel 90B controlled by gate set 90A.

Register Banks 71 and 72 may be controlled simultaneously so as to address in each of them one and only one register at each time. They provide the contemporary physical support for data which are fed to logic network 73.

Logic network 73 is connected to register A and bank 72 through distributor channels 70 and 80 by means of a certain number of AND gates. The AND gates are arranged in groups depending on: (1) the "format" of the data to be handled, (binary, decimal packed or unpacked), (2) the operation to be performed, and (3) the parallelism of the registers and logic network. In the arrangement shown, the registers and logic network 73 each have a parallelism of 16 bits. This allows parallel execution of operations on two "bytes" as well as a single byte at a time or on a zone of a byte. For this purpose, the AND gates which connect the input of operative network 73 to the distributor channels 70 and 80 are divided into groups 91 to 98. For 2 bytes (16 bits), groups 91 and 95 handle bits 0 to 3, groups 92 and 96 handle bits 4 to 7, groups 93 and 97 handle bits 8-10 and groups 94 and 98 handle bits 12 to 15.

The outputs of logic network 73 are connected to collector channel 75 through AND gate sets 100, 101, 102 and 103. In addition, the outputs of logic network 73 are connected through gate sets 104, 105, 106 and 107 to a decoding network 108 which identifies particular output configurations (e.g. all zeroes, or all zeroes with respect to bit groups corresponding to the enabled gates) by generating signal NZ2. Signal NZ2 at the output of the decoding network verifies the occurrence of a pre-established condition.

Logic network 73 may consist of conventional logic circuits arranged to perform operations of the following kinds:

(1) arithmetical operations of addition/subtraction (binary or decimal) and multiplication/division;

(2) logical operations, such as AND/OR/Exclusive OR, comparison and SHIFT: and

(3) incrementing operations (count up/down).

The operation to be performed is defined by a set of microcommands forwarded by the control unit to inputs 110.

It should be noted that data present on collector channel 75 following an operation performed by network 73 may be fed and loaded into a suitable register of bank 71 and from there into register A. The data can then be loaded into bank 72 for eventual storing in the main memory through channel 80A or for transfer to a peripheral unit through channel 80B and a gate set 111. Channel 80A communicates with Main Memory 122 (Figure 4), and channel 116 at the output of gate set 111 provides an output path toward peripheral units through input/output control unit 3. In

addition, "by-pass" channels 112 and 113, controlled by gate sets 114 and 115, connect distributor channels 70 and 80 to collector channel 75 directly. Certain flip-flops not shown are also provided to store specific conditions arising in the logic network, such as carry in-carry out and results of check operations performed on data by means of specialized decoding networks.

Figure 4 shows main memory 2 in block diagram form. The main memory comprises: a read/write memory 122 having 18 bit parallelism; an address register 121; a DATA register 120 and a suitable number of communication channels. The address register 121 receives information from register A through channel 70 (Figure 3) and AND gate set 124. Register 121 has a length related to the maximum desired memory capacity. For instance, a bit register enables the addressing of about 64,000 memory cells. Data to be written in the memory is transferred from register bank 72 to DATA register 120 through channel 80A controlled by AND gate set 125. Data contained in DATA register 120 is then written in the addressed memory portion through a channel 124. Data read out from the memory is forwarded on a channel 127 and, through an AND gate set 126, is loaded in DATA register 120. The data is then loaded through channel 118 into register bank 72. At the output of register 120 the data is further subjected to a parity check by means of parity control network 120A. If an error is detected, the network generates at the output an error signal ERR MM which is used to activate suitable diagnostic and error signalling procedures.

Interface

Referring to Figure 5, the interface comprises channel 17, a port 17A and an interface module 23 which contains a circuit set dedicated to the transfer of information and its timing on channel 17. Channel 17 comprises a number of data and control lines, each of which will be described.

Data Bus (Data Bus)

The Data Bus is a set of nine bidirectional lines (eight information bit lines plus an odd parity bit line) which provide a one byte wide path over the interface. The information transferred over the Data Bus basically includes:

* Data-out information (write data. I/O commands)
* Data-in information (read data. subsystem status);
* Control-out information (I/O instructions to the peripheral control unit PCU, which from now on will be called the device adapter or DA, as in the above-informed application 20774/75, Serial No. 1512263);
* Destination of the data sent from the I/O control unit (IOC) to the DA; and
* Control-in information (interface error reporting from DA). The meaning of the byte present on the Data Bus is defined by timing signals which may be concurrently present on the interface as will be seen later.

Data Strobe In (DSI): The DSI line is used with two functions:

1°) To forward from the peripheral Device Adapter 20 to the interface module 23 a timing signal which characterizes byte concurrently present on the Data Path as data when data are transferred from the peripheral unit to the central unit.

2°) As an answer signal to a timing signal DSO received from the interface module during transfer of information from the central unit to the peripheral unit.

Data Strobe Out (DSO): The DSO line is used with two functions:

1°) To forward from interface module 23 to peripheral DA a timing signal which characterizes the "byte" concurrently present on the Data Path as data when data are transferred from the central processor to the peripheral unit.

2°) As an answer signal to a timing signal DSI received from the device adapter during transfer of information from peripheral unit to the central processor.

Control Strobe In (CSI): The CSI line is used with two functions:

1°) To forward from the peripheral DA to the interface module 23 a timing signal which characterizes the "byte" concurrently present on the Data Path as control information, when control or service information is transferred from the peripheral DA to the central processor.

2°) As an answer signal to a timing signal CSO received from interface module 24 during transfer of control information from central processor to peripheral unit.

Control Strobe Out (CSO): The CSO line is used with two functions:

1°) To forward from interface module 23 to DA a timing signal which characterizes the "byte" concurrently present on the Data Path as control information, when control information is transferred from the central processor to the peripheral DA.

2°) As an answer signal to a timing signal CSI received from DA during transfer of control information from DA to central processor.

Operational Out (OPO):

The OPO line is used for signalling from module 23 to device adapter the status of the module 23 and the central processor. When OPO signal (in the following the same name will be used for signals and the lines which convey them) is at logical level "1", it indicates that the central processor and the interface module are in condition to work and to communicate with the DA. When OPO is at logical level "O" it indicates that the central processor and the interface module are not fed by the power supply or for some other reason are not in condition to work.

Operational In (OPI):

The OPI line is used for signalling from device adapter to module 23 the status of the device adapter. When OPI is at logical level "1" it indicates that the device adapter is in condition to work and to communicate with the central processor. When OPO is at logical level "O" it indicates that the device adapter is not fed by the power supply or is not in condition to work.

Reset Out (RSO):

This signal of impulsive type is forwarded by the central processor to the DA in order to reset the DA in an initial state. When not used this signal is at logical level "O".

Enable In (ENI):

This line is a control line used to provide a control signal from device adapter to interface module. The signal ENI has two functions:

1) When ENI is present, that is at logical level "1" it indicates that the direction of transferred information (data or control information) in the data Path is from peripheral device adapter towards central processor.

2) When is rises, that is when it goes from logical level "O" to logical level "1", it has the power to acquire control of the interface and to put it at the service of the device adapter so as to let it forward information to the central processor. When the DA is in reset status ENI must be at logical level "O". In addition, since (except for signal RSO already seen) it is not planned that the central processor could reset such signal ENI must be reset by the initiative of the DA in order to return the control responsibility over the interface to the central processor.

Enable Out (ENO)

The Enable Out line is a control signal line from module 23, to the DA. When module 23 recognizes the fall of ENI it causes ENO to be set. Thus when ENO is high and ENI is low, these signals indicate that the interface is either ready for or is currently performing an information transfer from module 23 to the DA. Similarly, when module 23 recognizes the rise of ENI, it causes ENO to be reset. Thus, when ENI is high and ENO is low, these signals indicate that the interface is either ready for or is currently performing an information transfer from the DA to module 23. Thus, the main function of these two signals ENO and ENI, is to provide a directional indication of the flow of data or control information over the data path lines. The other function is to provide an indication of good transfer of data on the interface.

As it will be seen later all the dialog sequences start with a control out transfer followed by a data transfer. Therefore, ENO must be high at the beginning of each dialog sequence. The data information transfer which follows in initiated by either module 23 or the DA as specified by the control-out information. The data-out information transfer initiated by module 23 is performed with ENO high. This data-out transfer can be aborted immediately by the DA and replaced by a control In Information transfer when the error (parity or illegal code) is detected by the DA during the control out transfer. The data–out information transfer is always followed by a double data path reversal to indicate the good finish of the dialog. If an error is detected by the DA on the data sequence, the DA may (but need not) initiate a control-in information transfer instead of this double data path reversal. At the end of each dialog sequence, ENI must be reset by the DA, which in turn sets ENO. The Data In information transfer initiated by the DA, when the control out information transfer gives DA the order, begins by raising ENI which in turn resets ENO. At the end of the data–in information transfer, the DA resets ENI which in turn sets ENO. When initialized ENO is set a logical ONE.

Event No. 1 (EV1)

Event No. 1 is a signal line from the DA which indicates the occurrence of a first group of asynchronous events at the DA. It is reset by the DA, when module 23 makes the appropriate response. When initialized, EV1 is reset to logical ZERO.

The typical function of EV1 is to signal that the DA is ready to execute a data transfer. EV1 is interpreted by the interface module as a request for character exchange. EV1 may be either present on the interface for the whole transfer period of more data or characters, or may be raised to logical level "1" and reset to logical level "O" at each character transfer depending on the particular type of peripheral unit involved in the data transfer. Indirectly EV1 causes the generation of a signal internal to the interface module, which signal will be designated by INTB (the same name has been used in the above cited patent application) and will be interpreted by the central processor as an interrupt request (for character exchange) to which a preestablished priority level will be attributed.

Event No 2 (EV2)

Event No. 2 is a signal line from the DA to module 23 which indicates the occurrence of a second group of asynchronous events at the DA. It is reset by the DA, when module 23 makes the appropriate response. When initialized, EV2 is reset to logical ZERO. The group of events which will cause the state change of this line are:

*Asynchronous events like device state change, operator panel caused changes.

*Events which are not time dependent.

*DA or device incident.

*End of order, end of off-line phase.

Module 23 responds to this signal line by initiating a dialog sequence. Therefore EV2 is interpreted by the central processor as an interrupt request of a second type. Such an interrupt request receives a suitable priority

level.

Event No. 3 (EV3)

EV3 is a signal line from device adapter which is used to indicate the detection of a parity error in the control information. This line reflects the interface integrity during the dialogue. EV3 signal is raised with SCI and lowered when the device adapter detects the fall of CSO. In rest condition this line is at logical level "0".

It must be remarked that when an error other than a parity error (for instance an illegal code error) is detected by the device adapter, this does not raise EV3 and start a control sequence in input to the central unit. The combined use of such control sequence and EV3 signal at logical level "0" or "1" depending on the circumstances, is the key aspect of the present system, as it will be seen later.

Integrated Interface Dialog Sequences

There are four basic dialogue sequences which may take place on the integrated interface:

*WRITE without error dialog sequence; that is, information transfer from central processor

*WRITE with error on data information dialog sequence

*READ without error dialog sequence; that is, information transfer from peripheral unit.

*READ or WRITE with error on control information dialog sequence.

As will be shown in the following, all these sequences, be they activated by central processor or by peripheral unit through an interrupt request EV1 or EV2, are always characterized by the preliminary forwarding to the DA of a control information or "control character". After this follows, in sequence, if no error is found in the "control character", the effective transmission or reception by the central processor of an information byte (data, command or peripheral unit status). The control character is composed of 9 bits one of which indicates the transfer direction (read or write in the sense above explained). Seven bits express (generally but not always) in coded form the address of a register within the peripheral subsystem, for instance a register for storing commands, or a status register where information pertaining to the status of the peripheral subsystem (that is DA plus peripheral units connected to the DA) may be read out. The ninth bit is used as a parity control bit to assure the integrity of the transferred information.

Some registers, used exclusively to store data to be read out or to be written in (for instance a write data register and a read data register) are conveniently addressed by a 6 bit code. In such a case, as will be shown in the following, the seventh remaining bit is used to provide additional information of different type.

A communication between central processor and DA is composed of a plurality of sequences of the indicated type, which will be described more in detail in the following and may be considered as consisting in three phases:

1° phase: Activating the communication.

In this phase the dialog sequences are activated by the central processor through its direct initiative. The DA does not forward any interrupt request of the EV1 or EV2 type. Through such sequences the control unit sends to the DA information pertaining to the peripheral unit to be selected, commands and other control information. Each information "byte" is always preceded and accompanied by the "control character" already mentioned.

2° phase: Data transfer phase.

As a consequence of the first phase, when so requested a data transfer phase may occur. In such a phase the dialog sequences are activated by signal EV1 for write sequences as well as for read sequences. EV1 may be raised and lowered by the DA at each character exchange (that is at each dialogue sequence) or may be kept at logical level "1" for the whole duration of the data transfer phase. Even in this case each information byte is preceded by the forwarding of the control character from interface module.

3° phase: Such phase is activated by the DA at the end of the second phase (or the first) by means of EV2 signal and is used to signal to the central processor the execution of the previously received commands as well as peripheral unit status. Even in this case the transfer of the control precedes the exchange of each information "byte".

Since the dialogue sequences, be they activated by the central processor or by interrupt requests, do not change in the following description and in figures 6, 7, 8 and 9, signals EV1 and EV2 which may be present during the dialog sequences are not mentioned or shown.

Write Without Error

Figure 6 illustrates in timing diagram forum the write without error dialog sequence. The timing relationship between signals is indicated by chaining arrows. Each signal transition is shown as seen at the source and will be received at the other end of the interface line with a delay which depends on the line length.

If ENO is high, module 23 simultaneously raises CSO and loads the control information. Upon detecting the rise of CSO, the DA raises CSI. Module 23 detects the rise of CSI and resets CSO. When the DA detects the fall of CSO, it can take the control information off the data path. This operation is indicated by an asterisk in Figure 6. The DA holds up the dialog in order to check the parity and the validity of the control information byte by delaying the reset of CSI. The reset of CSI indicates to module 23 that it can alter the control information off the data path; that is, the control information may be replaced by other information.

The control information transferred in this

dialog will have to be stored by the DA, because the information indicates the destination or the source of the data information that will subsequently be transmitted over the interface. This information is then viewed by the DA as an instruction. It is decided by the DA to determine the next dialog to occur. In that case, the next dialog is a data-out information transfer initiated by Module 23. Therefore module 23 simultaneously raises DSO and loads data information on the data path. Upon detecting the rise of DSO, the DA, if ready to receive the information, raises DSI. Module 23 detects the rise of DSI and resets DSO. When the DA detects the fall of DSO, it can reset DSI and take the data information off the data path. Such operation is indicated in figure 6 by two asterisks. The DA can hold up the dialog in order to check the parity by delaying the reset of DSI. The data information can be altered when module 23 detects the fall of DSI. Then the DA raises ENI which in turn causes ENO to be reset by module 23. Upon detection of the fall of ENO, the DA resets ENI. Upon detection of the fall of ENI, module 23 raises ENO. The interface is then ready to receive another dialog sequence.

Write With Error on Data Information

Figures 7 illustrates the write with error on data information dialog sequence. Each signal transition is shown as seen at the source.

This dialog begins when module 23 simultaneously raises CSO and loads the control information after checking that ENO and ENI are high and low respectively. Upon detecting the rise of CSO, the DA raises CSI. Module 23 detects the rise of CSI and resets CSO. When the DA detects the fall of CSO, it can take the control information off the data path and reset CSI. The DA holds up the dialog in order to check the parity and the validity of the control information byte by delaying the reset of CSI. The reset of CSI indicates to module 23 that it can alter the control information off the data path.

The control information transferred in this dialog will have to be stored by the DA, because the information indicates the destination or the source of the data information that will be subsequently transmitted over the interface. This information is then viewed by the DA as an instruction, and is decoded by the DA to determine the next dialog to occur. In this example, the next dialog is a data out information transfer initiated by module 23 and it assumed that no error is detected in the control information. Module 23 simultaneously raises DSO and loads data information on the data path. Upon detection of the rise of DSO, the DA, if ready, raises DSI. Module 23 detects the rise of DSI and resets DSO. When the DA detects the fall of DSO, it can take the data information off the data path and reset DSI. The DA can hold up the dialog in order to check the parity by delaying the reset of DSI. The data information can be altered by module 23 when it detects the fall of DSI.

After checking for a parity error, the DA raises ENI. Upon detecting the rise of ENI, module 23 resets ENO, and when the DA detects the fall of ENO it raises CSI. Upon detecting the rise of CSI, module 23 raises CSO. The DA detects the rise of CSO and drops CSI. When module 23 detects the fall of CSI, it resets CSO. This signal exchange with ENO low and ENI high informs module 23 that a parity error has been detected by DA. When the DA detects the reset of CSO, it drops ENI. This will cause setting of ENO, thereby allowing module 23 to initiate a new dialogue sequence. Error signalling by DA is not compulsory: if error signalling is not performed the DA simply initiates a double data path reversal as for the Write without error.

At this point in order to understand the principles underlying the invention, some clarification is needed. The information present on Data Bus may be basically of two kinds:

1) simple data

2) command or control information, for instance the address of a zone in the peripheral unit where information must be recorded

In the first case an error detected in the information does not influence the following operation, nor does it hamper or damage other information. For instance, if the peripheral unit is a printer and the transferred information is a character to be printed, an error in the character coding will produce the printing of a character other than the desired one but will not have any other effect. In the second case an error detected in the information may have serious consequences on the following operations and may damage other information. For instance in case of a disk memory unit, control information for addressing a track in a disk, which for some reason may be wrong, can lead to the subsequent erasing or recording of information in the wrong zone of the disk, with consequent loss of information already contained in that zone.

Therefore in one case, error signalling is not compulsory and the DA may defer such signalling to the end of the operation in course. In the other case, on the contrary, it is essential that the error be signalled immediately.

From another point of view, it is necessary to distinguish among different kind of errors. As will be seen later, a parity error in the information is essentially due to wrong interface operation. In fact, in the system here described the parity bit for each exchanged information is generated immediately upstream of the interface and checked immediately downstream. Therefore if a parity error is detected, this must be attributed to wrong operation of the interface, by reason of a fault or some electrical noise.

Another kind of error which may be

detected is of logical nature: certain binary codes with the meaning of commands or control information are allowed, and others are not. The allowed codes may be recognized and distinguished from those which are not allowed by means of decoding networks. Simple data will often have a specific format: for instance numerical data in unpacked form is represented by a bit-quartet accompanied by another bit-quartet or zone, which characterizes it as an unpacked decimal digit. The zone may be coded and used to perform suitable controls: if it is preestablished that transferred data is of a certain kind, for instance unpacked decimal and data is detected which is not an unpacked decimal digit, a logical error is detected.

The logical errors cannot be attributed to interface failure, but rather to some faulty operation within the central unit. It is irrelevant whether the errors can be attributed to wrong programming or to some hardware failure: the essential fact is that the interface can be excluded as possible error source.

After what has been said, it is clear that parity errors may be also defined as external errors and the other errors as internal errors. The distinction between internal errors and external errors, in the case of error detection, is made by the use of signal EV3. It has been mentioned that the exchange of signals CSI-CSO in the phase of dialogue inversion is used to signal the detection of an error (internal or external). If during the exchange of signals CSI-CSO signal EV3 is at logical level "1" the detected error is further characterized as external or interface error. In contrast, if EV3 is at logical level "0" the detected error is characterized as internal error.

Read Without Error

Figure 8 illustrates this dialog sequence, which begins when module 23 simultaneously raises CSO and loads the control information after checking that ENO and ENI are high and low respectively. Upon detecting the rise of CSO, the DA raises CSI. Module 23 detects the rise of CSI and resets CSO. When the DA detects the fall of CSO, it can take the control information off the data path and reset CSI. The reset of CSI indicates to module 23 that it can alter the control information off the data path.

The control information transferred in this dialog will have to be stored in the DA, because the information indicates the destination or the source of the data information that will subsequently be transmitted over the interface. This information is thus viewed by the DA as an instruction. It is decoded by the DA to determine the next dialog to occur. In this example, the next dialog is a data-in information transfer initiated by the DA. For this, the DA raises ENI. Upon detecting the rise of ENI, module 23 resets ENO. When the DA detects the fall of ENO, it places the data information on the data and raises Data Strobe in (DSI). Upon detecting the rise of DSI, module 23 raises DSO. The DA detects the rise of DSO and resets DSI. When module 23 detects the fall of DSI, it can take the data information off the data path and reset DSO. The DA on detecting the fall of DSO, resets ENI. The reset of ENI causes module 23 to raise ENO, so that module 23 can initiate another dialog sequence.

Write Or Read With Error on Control Information

Figure 9 illustrates this dialog sequence, which begins when module 23 simultaneously raises CSO and loads the control information after checking that ENO and ENI are high and low respectively. Upon detecting the rise of CSO, the DA raises CSO. Module 23 detects the rise of CSI and resets CSO. When the DA detects the fall of CSO, it can take the control information off the data path and reset CSI. The DA holds up the dialog in order to check the parity and the validity of the control information byte by delaying the reset of CSI. The reset of CSI indicates to module 23 that it can alter the control information off the data path. This information is viewed by the DA as an instruction. It is stored in the DA and decoded by the DA to determine the next dialog to occur.

In this example, the DA detects an error on the control information byte. It initiates a Control In information transfer, and so raises ENI. Upon detecting the rise of ENI, module 23 resets ENO (causing DSO to be reset if it was raised). When the DA detects the fall of ENO it raises CSI and, if appropriate, EV3, which means that the error is external. Upon detecting the rise of CSI, module 23 raises CSO. The DA detects the rise of CSO and drops CSI, and when module 23 detects the fall of CSI it resets CSO. When the DA detects the reset of CSO it drops ENI. This will cause the setting of ENO, thereby allowing module 23 to initiate a new dialog sequence.

Interface Hardware.

The preceding functional description shows that both in the central unit and in the device adapter there must be some circuit which supervises in a specific way the information transfer. Such circuits which may be grouped for convenience on modular boards, and interface module 23 will be now described in detail, with reference to Figures 10 to 13. In the figures, an open circle represents a gate or a gate set which performs a logical AND function. A rising edge waveform (upward pointing arrow) symbol on the input of an AND gate set indicates that the gate set produces a high output signal on the leading edge of the signal conducted to the input (assuming other inputs are properly switched); a falling edge waveform symbol (downward pointing arrow) indicates that the gate set produces a high output signal on the trailing edge of the signal conducted to the input. A line over the name of a signal indicates an

inverted signal obtained from the direct one through an inverting circuit (generally now shown). Similarly, for clarity of representation, when two or more signals are collected in a single line, the OR circuit generally required for such operation is not represented. For other circuit elements such as flip-flops and one-shots, conventional representation is adopted.

In general terms it will be seen that the interface circuitry receives from the central processor microcommands and signals timed by the interval timing of the central processor and forwards to the central processor a set of information which, apart from few exceptions, are timed according to the internal timing of the central processor. For clarity of representation the interface circuitry is divided in blocks which are in fact interconnected, signals passing from one block to another being given the same name in both. Also, when not otherwise specified the same name will be used for a generating logical element of the signal generated by it.

The circuitry for generating signal CSO comprises (Figure 10) a flip-flop FCSO with set input 134 and reset input 135 fed by AND gates 136 to 139 as shown. The direct output of flip-flop FCSO is fed to gates 140 and 141 which in turn feed line driver 143.

FCSO is set by a microcommand TC3 when an interface channel selecting signal PSEL (PORT SELECT) is present)PSEL is generated by the central processor) and enables AND gate 136. FCSO is also set by the rising front of signal CSI when $\overline{ENO}$ signal enables and gate 137. As previously seen, this happens when a dialogue with error sequence is preformed. FCSO is reset by a "CLEAR" signal issued by the central processor during an initialization procedure. In addition it is reset when gate 138 is enabled by the rising front of CSI if ENO is high, that is on the occurrence of information transfer from central processor towards peripheral unit. In the case of a dialogue sequence with error detection it is reset byy the falling front of CSI when $\overline{ENO}$ is high and gate 139 is enabled.

Signal FCSO at the output of flip-flop FCSO is transferred to the driver 143 through two alternative paths. In a dialogue sequence performed in a data transfer phase, a signal, DTP ("Data Transfer Phase") which comes from the central processor enables gate 140. Signal DTP and its inverted signal $\overline{DTP}$ are generated by flip-flop FDTP (figure 13) which is set and reset by microcommands TC7 and TC8 respectively. A second enabling signal EV1 controls and subjects the transfer of FCSO to the presence of an interrupt request for character exchange sent by the DA through a signal EV1. During dialogue sequences which are performed in phases other than data transfer phases, signal $\overline{DTP}$ is present in lieu of DTP and signal FCSO is transferred without delay through gate 141, enabled by DTP.

The circuitry for generating signal DSO (Figure 10) comprises flip-flop FDSO provided with set input 144 and reset input 145, fed by gates 146 to 149 as shown. The output of flip-flop FDSO is connected to line driver 153 and provides signal DSO.

FDSO is set by the falling front of CSI when gate 146 is enabled by a suitable signal A/C_{00}. Generation of signal A/C_{00} will be described later; it corresponds to bit 00 of the already mentioned "control information" and specifies the transfer direction. FDSO is also set by the rising front of DSI when gate 147 is enabled by signal $\overline{A/C_{00}}$.

FDSO is reset in three ways:
- by a "clear" signal during initialization
- by the falling front of DSI when gate 148 is enabled by signal $\overline{A/C_{00}}$
- by the raising front of DSI when gate 149 is enabled by signal $\overline{A/C_{00}}$.

The alternative set/reset modes correspond to the different dialogue modes already indicated, that is to information transfer towards the DA, (indicated by signal A/C_{00}) or towards the central unit (indicated by signal $\overline{A/C_{00}}$).

The circuitry for generating signal ENO (Figure 11) comprises flip-flop FENO which is provided with set input 154 and reset input 155. The output of flip-flop FENO is connected to a line driver 157. According to the interdependence already mentioned between signals ENO and ENI, FENO is set by a "clear" signal or by signal RSO or by the falling front of signal ENI. It is reset by the rising front of signal ENI.

With reference to Figure 12 the circuitry for controlling the flow of data on the data bus from a B register in the central processor comprises a control register A/C 162, which is used to store a "control character" read out from B register through channel 116, when enabled by gate 164 register A/C stores this character in bit positions 00 to 07. Bit A/C_{00} read out from register A/C is used, as already seen to control the set and reset of flip-flop FDSO. A register DATA OUT 165 having 9 bits, is used to store information read out from register B when the gate set 167 is enabled.

The data flow from outputs of register A/C and DATA OUT is controlled by gate sets 174 and 175, gate 175A, and flip-flop ABDO which in turn is provided with set input 169, reset input 170 and complementary outputs 171 and 172 connected to enable alternatively gates 174 or 175 and 175A. Flip-flop ABDO is set by the falling front of signal CSI and is reset either by the falling front of ENI or by an initialization signal "CLEAR" Bits 00 to 06 read out from register A/C and DATA OUT are alternatively conveyed on the same channel 199 to seven line drivers 190 to 196 and from there to the DATA BUS.

Bit 07 read out from DATA OUT register is conveyed to line driver 197 through line 199A and gate 175 enabled by output 172 of flip-flop

ABDO. Bit A/C_{07} of the control information, in contrast, is subjected to a treatment which will be explained in the following. Flip-flop ACUC, provided with set input 177, reset input 178 and complementary outputs 180 and 181 is used to control gates 183 and 184 which receive from A/C register to one of their inputs, signals $\overline{A/C_{07}}$ and A/C_{07} respectively. According to the set or reset status of ACUC and through gate 185 enabled by signal $\overline{ABDO}$, signal $\overline{A/C_{07}}$ or signal A/C_{07} is conveyed to the line driver 197.

Flip-flop ACUC is reset by the rising front of ENI and is set by the presence of more signals according to a mechanism which will be described in the following. Circuitry for controlling the output data flow from the central unit is completed by a parity bit generation network 186. This network may consist for instance of a plurality of Exclusive OR's suitable cascaded. It receives as inputs from channel 199 and from line 199A the eight information bits (in the case of control information as well as data information) and provides an output to the line driver 198 of the required parity bit.

Since the DATA BUS is bidirectional, the line drfivers 190 to 198 are provided with an enabling input 187 connected to the output of flip-flop FENO so that they are enabled only when information has to be transferred from module 23 to DA.

Still with reference to figure 12, module 23 receives data from DATA BUS through line receivers 200 to 208 and channel 210. When one-shot 212 is activated by the falling front of signal DSI the gate set 213 enables thedata to be loaded in register DATA IN 215. The data flow between register DATA IN and register B within the central unit is controlled by gate set 218 which is enabled by signal PSEL ("Port Select") and by microcommand TC4, both received from the central processor. The output of register DATA IN is additionally connected to a parity check network 216, which in case of parity error detection generates at the output an error signal ERDI.

With reference to figure 11 there are also shown circuits used by some signals which are simply transferred by module 23. They are: a line receiver 219 which transfers signal EV2 from the integrated interface to the central unit; a linedriver 159 which receives from the central unit a reset microcommand RSO1 and transfers it to the interface as signal RSO; a line receiver 220 which receives from the interface the OPI signal and transfers it to the central unit as OP11 signal; and a line driver 221 which receives from the central unit the OPO signal and transfers it on the interface. Similarly receivers 225, 226, 227, 228 and 228A are used to receive signals CSI, DSI, ENI, EV1 and EV3.

With reference to EV1, it has been said that in an indirect way it generates an interrupt request for character exchange. Since EV1 is not transferred to the central unit, it is useful to explain how this is done.

It has been seen in describing the generation of signal CSO that in the data transfer phase, the generation of signal CSO and the dialogue sequence, the fall of EN1 is interpreted in the data transfer phase, as an interrupt request for character exchange INT B. The generation of INT B is provided by flip-flop FINT (Figure 11). FINT is set by microcommand TC3 and is reset by the fall of ENI. The direct output of FINT is connected to an input of the AND gates 223 and 224, gate 224 being enabled by signals PSEL and $\overline{DTP}$. Therefore the output of gate 224 is a signal, named BUSY, at logical level "1" for the whole duration of the dialogue sequence, if not in data transfer phase. Such a signal indicates to the central unit that the interface module is busy in the execution of a dialogue sequence.

Gate 223 is enabled by signal DTP. The falling front of a signal INT B appears at the output of gate 223 only if is enabled by DTP and when flip-flop FINT is reset by the fall of ENI. The fall of INT B constitutes an interrupt for character exchange, in the data transfer phase, which is presented to the priority network and is there stored (see the cited patent application).

According to the present system the interface module 23 is further provided with circuits for detecting and signalling errors (figure 13).

Signal ERDI, already seen is indicative of parity error regarding the received information and is applied through AND gate 228 B to flip-flop ERCE when gate 228 B is enabled by microcommand TC4 (Data IN B). ERCE may be set also by MM ERROR signal from the parity control network 120A, connected in the memory unit (Figure 4), to DATA register 120 through the AND gate 223 enabled by PSEL signal and by a TC9 microcommand. The microcommand TC9 is provided by the control unit when the information read out from the memory is directed to a peripheral unit and therefore is transferred from register 120 to register bank 72 through the channel 118 and from there to the selected peripheral unit through the channel 80, the set of AND gates 111 and the channel 116. Hence ERCE has the function of storing either a parity error in the information to be transferred to the peripheral unit or a parity error in the information received from the peripheral unit. The output of ERCE is connected to the follecting wire 243 through AND gate 240A, enabled by DTP and PSEL signals; further it is connected to the wire 237 through the AND gate 240, enabled by PSEL signal and TC6 microcommand. ERCE is reset by a microcommand TC5, through the AND gate 229, enabled by PSEL signal.

Flip-flop ERTO provides a "TIME OUT" error indication. In command sending phases or in status reading phases from the DA, each dialogue sequence has to be executed within a

predetermined time interval: if this does not happen a "Time Out" error occurs. For this purpose microcommand TC3 (which also generates CSO through FCSO and activates the dialogue sequence) triggers one-shot 230 through AND gate 231, when gate 231 is enabled by signals $\overline{DTP}$ and PSEL. At the same time TC3 sets flip-flop 233A. The fall of the pulse generated by one-shot 230, which has a preset delay time, sets ERTO through AND gate 232 if that gate is enabled by the signal on the direct output of flip-flop 233A. Flip-flop 233A is reset either by the fall of ENI or by the raising of signal ERTO at the output of flip-flop ERTO. Clearly ERTO is set only if the dialogue sequence is not completed within the delay time of the one-shot. Therefore the output of ERTO provides a "Time Out" error signal, which is applied on collecting wire 243, and on wire 238 through AND gate 241, enabled by PSEL signal and by microcommand TC6. ERTO is reset by microcommand TC5 when AND gate 229 is enabled by PSEL signal.

Flip-Flop EIDI provides an error signal for "dialogue inversion". It has been seen that, if the DA detects a parity error or a logical error, it performs a dialogue inversion, that is, it raises ENI and then CSI. Such a circumstance is recognized by flip-flop EIDI which is set by CSI through AND gate 234, enabled by signal ENI. The output of EIDI is conveyed on collecting wire 243 through AND gate 236, enabled by $\overline{DTP}$ and PSEL, as well as on wire 239 through AND gate 242, enabled by PSEL and TC6. EIDI too is reset by microcommand TC5 when gate 229 is enabled by signal PORT SELECT.

Flip-flop EOPI provides an error signal for a fall of OPI; that is in case of peripheral unit deenergization. The signal OPI 1 at the output of line receiver 220 is applied to the input of univibrator 235, which is triggered by the falling front of this signal and provides at its output a pulse which sets flip-flop EOPI through AND gate 244 enabled by signal OPO. The output of EOPI is connected to the collecting wire 243 through AND gate 245 enabled by signals $\overline{DTP}$ and PSEL, as well as to wire 246 through AND gate 247 enabled by PSEL and by microcommand TC6. EOPI too is reset by microcommand TC5 when gate 229 is enabled by PSEL.

Lastly flip-flop FEV 3, set by signal EV3, stores such a signal. The output of FEV 3 is connected to the wire 248 through the AND gate 249 enabled by PSEL and by the microcommand TC6. FEV 3 too is reset by microcommand TC5 when gate 229 is enabled by PSEL.

Therefore the various signals at the output of flip-flopf ERCE, ERTO, EIDI, and EOPI are conveyed on a single wire 243, which communicates to the central unit the generical occurrence of an error with a signal "MCK". The operating conditions in which such a signal is generated will be seen later. Additionally individual output wires 237, 238, 239, 246 and 247 enable the various error conditions to be loaded onto channel 210A and from there into register B of central unit when AND gates 240, 241, 242, 247, and 249 are enabled by signal PSEL and by microcommand TC6 (ERROR B).

The operating characteristics of error lines will be seen later. For the moment the operating characteristics of control network of information exchange, and first of all a command phase will be considered. By forwarding signal PSEL and microcommands TC1 and TC2, the central unit loads registers A/C and DATA OUT respectively with control information which specifies destination and direction of the information transfer, and with a command. Since the central unit does not forward the microcommand TC7 to module 23, flip-flop FDTP is in reset status and signal $\overline{DSP}$ is present, which indicates that the channel is not in the data transfer phase, but is in command or status transfer phase. When register A/C and DATA OUT are located, microcommand TC3 sets flip-flop FCSO through gate 136. Since $\overline{DTP}$ enables gate 141, as soon as FCSO is set, signal CSO is generated. Output 171 of flip-flop ABDO which is initially in resent status (by reason of a "clear" signal) enables gates set 174, and the control information contained in A/C is therefore present on Data Bus.

The dialogue sequence develops in the way already seen. The DA answers to CSO raising CSI and therefore CSO is lowered (FCSO is reset). The DA receives the control information and lowers CSI. At this point FDSO and ABDO are set so that signal DSO is generated and gates set 175 and gate 175A are enabled. On the DATA BUS the information contained in register DATA OUT is now present and the sequence is concluded in the way already seen, that is with the exchange of signals DSO, DSI, ENI and ENO and, in the case of error, with the additional exchange of CSI and CSO when ENI is high. During the dialogue sequence, interval flip-flop FINT, which has been set by microcommand TC3, sends a "BUSY" signal to the central unit. Such a signal is used to prevent the central processor from proceeding in the execution of the microprogram which has started the command transfer phase. Stopping of the microprogram is obtained, for instance by inhibiting the microprogram memory output by signal NOP, which inhibits gates set 66A, thus inhibiting the generation of new microinstructions until the dialogue sequence is completed. At this point a new microinstruction sequence may start a new dialogue sequence and transfer a new command.

In the case where the dialogue sequence requests the reading of statuses from DA the behaviour is similar. By microcommand TC1 register A/C is loaded with control information, then microcommand TC3 starts the dialogue. When signals CSO and CSI are exchanged and

the control information is forwarded, the DA generates ENI and the data transfer path is reversed. The DA places on the data bus the required information, accompanied by the signal PSI. When DSI is lowered univibrator 212 is triggered and the information is loaded in register DATA IN. The dialog sequence is concluded as for the already seen model. In this case, the microprogram which has started the dialogue sequences is arrested by the signal BUSY. At the end of the sequence the microprogram is again enabled to proceed and a new microinstruction generates microcommand TC4, which enables gates set 218 to transfer the received information in register B of central processor.

In the case where the initiative for the information exchange is taken by the DA through signal EV2, the dialogue sequence is not changed. Once EV2 is acknowledged and the interrupt request is accepted, it is always the central processor which performs the information exchange through the dialogue sequence already seen.

At the end of the command phase the address of the register in which the data is to be written (or from which data is to be read out) is loaded into register A/C, and in the case of data to be written the first data character is loaded into register DATA OUT. At the end of such operations, flip-flop F DTP is set by microcommand TC7 and thereafter microcommand TC3 is generated. Therefore flip-flops FINT and FCSO are set. However, signal CSO is issued only at the reception of signal EV1, if that signal was not already present, and the dialogue sequence starts. At the end of the dialogue sequence, the fall of ENI resets flip-flop FINT, and causes the fall of signal INT B at the output of gate 223. The fall of INT B is detected by the priority network which, once interrupt requests of higher priority have been honoured, presents it to the central processor. A new data character is then loaded in register DATA OUT and microcommand TC3 is generated again. Then a new dialogue sequence starts as soon as a new EV1 signal is received, if it was not already present.

In the case of data read out from DA the mechanism is similar. At the beginning of the data transfer, suitable commands are forwarded and register A/C is loaded. Immediately after, signal DTP is set and microcommand TC3 is forwarded. When signal EV1 is received the dialogue sequence begins and the first character read out from DA is loaded in register DATA IN. By the fall of ENI, flip-flop FINT is reset and signal INT B is lowered. The fall of INT B is detected by the priority network, which upon execution of services having higher priority, presents it to the central processor. The central processor transfers the content of DATA IN in the working register B, and from there to a suitable main memory position. Once this operation is performed the central processor generates a new microcommand TC3 so that flip-flops FINT and FCSO are again set. When a new EV1 signal is received, if not already present, a new dialogue sequence is started.

Now it remains to be considered how the central processor informs the DA that the data transfer operation must terminate. For this purpose the control information which is sent with the last character to be read out is suitably modified by setting flip-flop ACUC. It has been said that the name or address of the two DA registers, where the data are written or read out, is expressed by a 6 bit code rather than a 7 bit code contained in AC register. Bit 07 of the same register is not significant for the address definition and may be set to logical level "1" or "0" at will. Therefore a specific control function may be attributed to such a bit: for instance if it is set to "0" it may indicate that the character to be transferred is not the last one; if it is set to "1" it may indicate that the character to be transferred is the last one. At the beginning of a transfer operation it is set to "0".

When the central processor recognizes that the transfer must conclude by the transfer of a last character, it forwards to module 23 a signal ΔLL. ΔLL may be obtained in the central processor by updating the transfer length expressed as a number of characters. Such length may be contained in a suitable working register and is decremented by 1 in the operative network of the central processor at each data character transfer. At the output of the operative network a decoding network detects when such a length is reduced to a preestablished value (one or zero) and generates signal ΔLL. Referring to figure 12, signal ΔLL is applied to an input of AND gate 188, which, when enabled by microcommand TC3, by signal PSEL and by signal DTP transfers it to the set input of flip-flop ACUC thus setting it. Hence the output 181 of flip-flop ACUC enables AND gate 183 which transfers signal $\overline{A/C_{07}}$ that is the inverse of bit 07 contained in register A/C, on wire 199A through gate 185 enabled by signal $\overline{ABDC}$. Therefore control information is forwarded to the DA indicating through the inversion of the last bit, that the data transfer must be concluded. In such a way the central unit forwards an end of transfer signal to the DA by using the Data Bus, without the need of an additional wire.

Now we have to consider the behaviour of the error detecting and signalling lines.

ERCE

Flip-flop ERCE has the double function of storing interface or channel errors on transfers towards the central processor, and detecting memory errors on data transfers towards a peripheral unit.

1st case: As will be seen later, in the case of information from the output of DA directly to the central processor, the parity bit is

immediately computed before entering the communication or interface lines, in the same way as the parity bit for the information from the central processor. As shown in figure 12, the parity check is performed immediately at the output of line receivers and interface registers, that is before the information is transferred into the central processor. Therefore if a parity error is detected by network 216 (figure 12), such error necessarily arose in the interface wires or line transmitters/receivers. It is clear that the detected error may be caused either by a failure or by an electrical noise. The parity error is stored in ERCE through AND gate 228 and signalled to the central processor.

However the behaviour is different according to whether the error is detected during a data transfer phase or a service information exchange phase. If the error is detected during the service information transfer (for instance: status of peripheral unit) the signal $\overline{DTP}$ enables AND gate 240A (together with the signal PSEL) and the error signal is forwarded on collector wire 243. From here through AND gate 250 enabled by signal MCE (MACHINE CHECK ENABLE), normally present, the error signal is forwarded to the central processor, where suitable circuits, not shown, generate the following commands:

a) blocking of input/output control unit in the present state;

b) forcing on channel 58A of control unit the ROS address, where a microprogram for managing error interrupts is located.

c) resetting of signal MCE in order to mask signal MCK.

The central processor starts the called microprogram and, after checking that the error concerns neither ROS nor the basic operating network (the description of such microprogram being beyond the scope of the present system), it tries to identify the causes of the error, For this purpose the central processor generates the microcommand TC6 (PSEL is already present in consequence of the I/) control unit blocking) and the contents of all the error flip-flops ERCE, ERTC etc., are transferred through the wires 237, 238, 239, 246 and 248, into register B of the central processor where they are analysed through comparison with suitable masks. After having recognized that it is an interface error or an external error the microprogram for managing errors resets error flip-flops by generating the microcommand TC5 disconnects the channel by lowering OPO (limitedly to the channel), restores the condition MCE, to enable the I/O control unit, and signals to the software the information relating to the error.

The Supervisor program decides what actions have to be taken in relation to the type of detected error. For instance if it is an external error, which may be caused by electrical noise, the Supervisor program may decide to try again the information transfer operation on which the error occurred. If this operation too fails to be completed the Supervisor program may decide to start functional tests in order to determine the cause of the failure with better precision and then ask for the intervention of the operator. If the error is detected during data transfer operation, AND gate 240A is blocked because signal $\overline{DTP}$ is low. Therefore the data, though wrong, are transferred to the central processor. In fact in such a case, during all data transfer operations there is no danger that the reception of wrong information may jeopardize the development of the following operations or may damage other information, which are in memory. However at the end of data transfer operation, before the central processor resets flip-flop FDTP by microcommand TC8, the state of flip-flop ERCE may be checked by generation of microcommand TC6 and consequent transfer of the interface information on the data channel. After that, the different flip-flops ERCE, ERTO etc. are reset by microcommand TC5 and at the end FDTP is reset by TC8.

2nd case: For information from the central processor and more precisely from memory, the parity bit relating to the information read from memory and to be transferred to the peripheral unit is tested immediately at the output of memory DATA register (Figure 4), by means of network 120A. In the presence of a microcommand TC9 which indicates that data transfer operation from memory to a peripheral unit is in course, an error (MM ERROR), detected by the network 120A, is transferred through AND gate 233 to the set input of flip-flop ERCE, which is set. Since the data transfer takes place typically during a phase in which $\overline{DTP}$ is absent, the signalling of the error is delayed until the end of the data transfer phase, as we have already seen for the received data. In fact, if the transmitted data is wrong, it does not compromise the execution of the data transfer operation which is being carried out.

ERTO

As already seen, flip-flop ERTO provides an error signal for "Time Out". The error signal present at the output of ERTO is forwarded to the collecting wire 243 directly. In fact a "time out" error is an interface error which may jeopardize the computer operation: if an interface dialogue is not completed within a preset time, there is a risk it will not be completed at all, and in such case the computer would remain perennially awaiting the end of the dialogue before it proceeds to other operations. Therefore the "time out" error, stored in ERTO, enables immediately, through AND gate 250 the generation of signal MCK, with the consequences already seen.

EIDI

As already indicated flip-flop EIDI provides an error signal for "dialogue inversion". A dialogue inversion may assume two

significances according to whether it is accompanied or not by signal EV3. When it is not accompanied by signal EV3 the dialogue inversion points out that the DA recognized an information which, through formally correct with regard to the parity, is unacceptable for other reasons. For instance, in case of commands, the command code is not recognized among the allowable command codes and therefore cannot be interpreted. Also in case of control information, which is in substance an addressing code, such code is not recognized among the allowable command codes and therefore it does not provide a valid address. If the transferred information is simple data, it is generally more difficult for the DA to perform logical checks on the information. Sometimes it is possible to check logically the information, for example, when the information is provided with a "zone". In any event, when a dialogue inversion not accompanied by EV3 occurs the error which is recognized is a logical error and imputable to software errors or to internal operations of central processor which are not correct from the logical point of view. However they are not errors imputable to failures of or static on the interface, since the information received by DA is formally correct. In contrast when the dialogue inversion is accompanied by EV3 that means a parity error has been recognized and therefore an interface failure or interference occurred during the transmission from central processor and is pointed out to DA.

The signalling of the errors for dialogue inversion is performed in this way. The output of flip-flop EIDI is applied to collector wire 243 through AND gate 236. When this gate is enabled by signals DTP and PSEL, the signal MCK is generated, as soon as the dialogue inversion is recognized. The microprogram for managing errors operates as in the preceding cases by enabling, by means of microcommand TC6, the loading of register B with the contents of various flip-flops for storing errors, including flip-flop EV3. Therefore the nature of the error is recognized and the most suitable actions are taken.

This always happens in the control phases; that is for the control character, the commands forwarded by central processor, and the control character forwarded by central processor to request DA to send the peripheral unit status. In contrast, during the data transfer phase, the signalling of dialogue inversion is postponed. In particular the output of EIDI is connected to an error wire 251. The logical level present on this wire may be tested by the central processor at the end of an operation and before it starts to interpret a new program instruction. In other words in a computer able to execute contemporaneously several processes (multiprogramming), as soon as an operation of a process ends the interpretation phase of the following instruction of the same process verifies also the conditions of execution of the concurrent operations which are being performed relating to other processes, and recognizes any occurrence of an error in these processes too. In the case of a computer able to execute only one process at a time, any error will be recognized at the end of the data transfer operation, by inspecting of the contents of interface flip-flops ERCE, ERTO etc. as already seen. Similar behaviour takes place if the DA and associated peripheral unit go out of order and therefore the signal OPI falls.

Interface circuits of the peripheral unit or DA

It is now useful, for completeness of the description, to mention the interface circuits on the DA side. As shown in figure 14 such circuits include essentially a register A/C1 for storing the received control character, a register WDR for storing the received information, and a register RDR for storing the information to be forwarded to the central processor. The control characters received on DATA BUS are transferred through line receivers 300 to channel 301 and from here are transferred to register AC1 through AND gate 302 enabled by command C1. The information contained in register AC1 is verified for parity by the control network 303, which provides an address error signal ADE on such error, and the information is then decoded by decoder 304 which provides at its output: signals for selecting registers of peripheral units (P1 . . . P2 . . . PN); a "last character" signal when that is the case; and if necessary, an illegal address signal ADI. The first flip-flop of register A/C1 further provides information about the direction of the following transfer operation.

The information received from the central processor is transferred to register WDR through gates set 305 enabled by command C2. The information contained in WDR is verified for parity by the control network 306 and for the presence of errors and logical incompatibility by decoder 307, which provide at their outputs respectively the signals DPE (Data Parity Error) and IDE (Illegal Data Error). Then through the channel 308, the information is register WDR is transferred to the register selected by decoder 304.

In contrast, the information to be forwarded to the central processor is taken out from registers selected by a control character and loaded into register RDR. From here the information is transferred on channel 310 through gate set 309 enabled by a command C3, and finally is fed onto DATA BUS through line transmitters 311. At the output of register RDR the parity bit, which accompanies the transmitted information, is calculated by means of check generation network 312. The transfer of information from and to DATA BUS is controlled by clock circuits and command generation circuits, like those considered in the interface module which will be described shortly.

Figure 15 represents the signal generation circuits CSI, DSI and ENI essentially composed by flip-flop FCSI, FDSI, FENI and by several AND gates and one-shots.

Flip-flop FENI is set by the falling front of signal DSI through AND gate 316 when this gate is enabled by signal W (WRITE) which specifies that the operation defined by the control information is a write operation. Alternatively flip-flop FENI is set by the falling front of signal CSI through AND gate 317 when this gate is enabled by signal R (READ) which specifies that the operation defined by the control information is a reading operation. These are the normal ways by which the flip-flop ENI is set, and it generates at its output the signal ENI, then forwarded on the interface. However, in the case of error detected on the control information, FENI is set by the fall of signal CSI independently of the direction of the transfer defined by signals R or W. This is achieved through the gate 318 enabled by signals EV3 and IDI, indicative, as we'll see of parity errors or logical errors. FENI is reset by the falling front of signal ENO, applied to its reset input, through AND gate 319, if such gate is enabled by signals $\overline{EV3}$, $\overline{IDI}$ and W, that is during write operations without error. Alternatively FENI is reset by the falling front of signal DSO applied to its reset input, through AND gate 320, if such gate is enabled by signal R, that is during reading operations. Finally FENI may be reset by the falling fron of signal CSO applied to its reset input, through AND gate 321, if such gate is enabled by signals EV3 and IDI, that is in case of error.

Flip-flop FCSO is set by the rise of signal CSO through AND gate 260 when such this is enabled by signal $\overline{DAV}$, $\overline{DAV}$ is generated by a flip-flop of DA, which describes the status of interface registers of DA. $\overline{DAV}$ points out that registers are empty and therefore the interface dialogue may go on. Further FCSI may be set by the falling front of signal ENO, when the gate 261 is enabled by signals EV3 or IDI. FCSI is reset through the chain of one-shots 262 and 263 if the dialogue is without error.

The falling front of CSO starts the one-shot 262, which generates a pulse at its output. When AND gate 264 is enabled by signal ENO, this pulse is transferred through this gate and generates the command C1 for loading the register AC1. The fall of this pulse starts the one-shot 263, at whose output a pulse C4 appears whose falling front resets FCSI. Further the pulse C4 is transferred through AND gate 265 when this gate is enabled by signals R, $\overline{EV3}$ and $\overline{IDI}$, and generates a signal START 1 which is used by the peripheral unit for enabling suitable clockings and other operations. The conditions R, $\overline{EVB}$ and $\overline{IDI}$ indicate that the operation to be performed is a reading operation (R is the output of the first cell of register A/C1) and that parity errors or logical errors have not been detected in the control character. (EV3 and IDI are the outputs of two flip-flops which will be considered later). Following the command START 1, the peripheral unit takes the information from register selected by the control character and loads it into register RDR (Figure 14). At this point the peripheral unit, by a SET command sets a suitable flip-flop FDAV, which signals that the information is available on the interface. The output DAV of the flip-flop FDAV enables (figure 14) gate set 309 to transfer the information on DATA BUS. The pulse C4, generated by univibrator 263, further enables (figure 16) gates 312 and 314 and so, when a parity error (ADE) or a logic error (ADI) are present in the control character flip-flops FEV3 and FIDI are respectively set. A further way for resetting flip-flop FCSI uses AND gate 266. In case of error, and therefore of dialogue inversion, AND gate is enabled by signal ENI and the rising front of CSO resets FCSI.

The function of the univibrators 262 and 263 is to provide a suitable clocking sequence (pulses C_1, C_4 and respective rising and falling fronts) in order to allow thesequential execution of the operations already seen, that is:

- loading of the control character into AC1;
- character checking,
- if required, triggering of the transfer operation to central processor.

The flip-flop FDSI, which generates the signal DSI, may be set in two ways. If the transfer direction imposed by the control character is a write (towards DA), the signal W (WRITE) coming from the first cell of register AC1, enables AND gate 267, so that the rising front of signal DSO sets FDSI. If the transfer direction imposed by the control character is a read operation (towards central processor), the signal R (READ) coming from the first cell of register AC1, is applied to AND gate 268. When DAV and $\overline{ENO}$ conditions are verified that is when the reversal of the direction of DATA BUS ($\overline{ENO}$ is present) has been carried out and when the information is available in register RDR (DAV is present), FDSI is set and it generates DSI. FDSI may also be reset in two ways. During read operations (R present), AND gate 269 is enabled and the rising front of DSI resets FDSI. During write operations (W present), AND gate 270 is enabled and the falling front of DSO resets FDSI through the chain of one-shots 271 and 272. One-shot 271 is triggered by falling front of DSO through AND gate 270. This one-shot generates at its output a command pulse C2 which enables AND gates set 305 (figure 14) to load register WDR. The falling front of this command pulse triggers the one-shot 272 which generates at its output a pulse C5, which enables AND gates 313 and 315 (figure 16), so that the presence of a parity error (DPE) or the presence of a logical error IDE detected in the received information

sets respectively flip-flop FEV 3 or FIDI. The falling front of C5 sets flip-flop FDAN, which, for writing operations, indicates that the data to be transferred is available in register WDR. Further the falling front of C5 resets flip-flop FDSI.

Figure 16 shows the flip-flops FEV 3, FIDI and FDAV, already mentioned.

FEV 3 stores the parity error signal and generates the signal EV3 forwarded to the central processor through the interface. FEV 3 is set by signal ADE (parity error found in the control character) applied to the set input through AND gate 312 enabled by command C4, and by signal DPE (parity error found in the datum) applied through AND gate 313 enabled by command C5. FEV 3 is reset by falling front of signal ENI at the end of the dialogue.

FIDI is used to store logic error signals and to generate the signal IDI, which triggers the exchange of signals CSI and CSO during the dialogue inversion; as already seen, FIDI is set by signal ADI (illegal control character) applied to the set input through AND gate 314 enabled by command C4, and is set by signal IDE (illegal received data) applied through AND gate 315 enabled by command C5. Besides triggering the exchange of signals CSI and CSO on the interface the signal IDI, generated by FIDI, may be used to perform other functions with DA, as, for instance, the sending of signal EV2 and others. Also the flip-flop FIDI, like FEV 3 is reset by the falling front of signal ENI.

FDAV is a flip-flop which is used to signal that, at interface level, a data to be read or written is available or, in other words that the interface register (reading or writing register as the case may be) is full. In fact FDAV is set by the falling front of command C5, that is when, during writing operations (from central processor to peripheral unit), the register WDR has been loaded. Conveniently the set command C5 is conditioned by AND gate 316 enabled by signals $\overline{EV3}$ and $\overline{IDI}$, so that the flip-flop FDAV is set only if the received datum is correct both with regard to the parity and from the logical point of view. When the data present in register WDR may effectively be accepted by the peripheral unit (it depends generally on its internal clock rate), the peripheral unit forwards a signal RES for resetting the flip-flop FDAV. Conversely, when the operation is a reading operation (from peripheral unit to central processor), the peripheral unit sets the flip-flop FDAV when the requested information has been fetched from one of registers $P_1 \ldots$ to $\ldots P_N$ and loaded into register RDR. In fact, at this point, the datum is available to be transferred to the interface. In this case, flip-flop DAV is reset by the falling front of signal DSO, that is after the transfer has been performed. It is clear that the DA contains other circuits such as line receivers and transmitters and specific circuits of the peripheral unit, which are not essential for the understanding of the present system.

Concluding observations

From the above description the basic characteristics of the interface of the present system are seen to be the following.

1) For both directions of transfer a parity bit is generated immediately at the input of the interface channel and it is checked immediately at the output of the interface channe. Therefore a parity error is indicative of interface failures or disturbances.

2) In addition to parity control networks, on the side of peripheral unit, control networks which check the logical correctness of the received information are provided. A logical error is indicative of a failure upstream of the input of the interface or of a programming error.

3) Two distinct means of error signalling are provided: an exchange of signals CSI and CSO for signalling logical errors, and the same exchange, accompanied by signal EV3 for signalling parity errors.

4) The mechanisms for recognizing error signals are organized in such a way as to suspend the transfer sequences when an error is signalled to the central processor, in case of both logical and parity errors, detected on the character control or on the service information (commands, addresses or status), provided that they are not simple data to be read or written. On the other hand, when an error is present on data, the recognition of the error signal is delayed until the end of data transfer.

WHAT WE CLAIM IS:–

1. A data processing system comprising a central processor and a peripheral unit connected by an interface which comprises an information bus and signal lines and on which information transfer is performed by the passage of control information from the central processor followed by the passage of command/service information or data information, the central processor and the peripheral unit each having a parity generating network for generating a parity bit for information being placed on the information bus and a parity checking network for checking the parity of information being received from the information bus, the peripheral unit also having a logic checking circuit for identifying logical errors in at least the command information, error signalling means in the peripheral unit for signalling over the signal lines an error signal if an error is detected in information received over the information bus and for further signalling a parity fault if the error is a parity error and error monitoring means if the central processor for determining, from the parity checking network in the central processor and the error signal and parity fault signal (if any), the occurrence of an error and whether the error is or is not one prejudicial to the operations which are to follow immediately.

* C L M S *

sets respectively flip-flop FEV 3 or FIDI. The falling front of C5 sets flip-flop FDAN, which, for writing operations, indicates that the data to be transferred is available in register WDR. Further the falling front of C5 resets flip-flop FDSI.

Figure 16 shows the flip-flops FEV 3, FIDI and FDAV, already mentioned.

FEV 3 stores the parity error signal and generates the signal EV3 forwarded to the central processor through the interface. FEV 3 is set by signal ADE (parity error found in the control character) applied to the set input through AND gate 312 enabled by command C4, and by signal DPE (parity error found in the datum) applied through AND gate 313 enabled by command C5. FEV 3 is reset by falling front of signal ENI at the end of the dialogue..

FIDI is used to store logic error signals and to generate the signal IDI, which triggers the exchange of signals CSI and CSO during the dialogue inversion; as already seen, FIDI is set by signal ADI (illegal control character) applied to the set input through AND gate 314 enabled by command C4, and is set by signal IDE (illegal received data) applied through AND gate 315 enabled by command C5. Besides triggering the exchange of signals CSI and CSO on the interface the signal IDI, generated by FIDI, may be used to perform other functions with DA, as, for instance, the sending of signal EV2 and others. Also the flip-flop FIDI, like FEV 3 is reset by the falling front of signal ENI.

FDAV is a flip-flop which is used to signal that, at interface level, a data to be read or written is available or, in other words that the interface register (reading or writing register as the case may be) is full. In fact FDAV is set by the falling front of command C5, that is when, during writing operations (from central processor to peripheral unit), the register WDR has been loaded. Conveniently the set command C5 is conditioned by AND gate 316 enabled by signals $\overline{EV3}$ and IDI, so that the flip-flop FDAV is set only if the received datum is correct both with regard to the parity and from the logical point of view. When the data present in register WDR may effectively be accepted by the peripheral unit (it depends generally on its internal clock rate), the peripheral unit forwards a signal RES for resetting the flip-flop FDAV. Conversely, when the operation is a reading operation (from peripheral unit to central processor), the peripheral unit sets the flip-flop FDAV when the requested information has been fetched from one of registers $P_1 \ldots$ to $\ldots P_N$ and loaded into register RDR. In fact, at this point, the datum is available to be transferred to the interface. In this case, flip-flop DAV is reset by the falling front of signal DSO, that is after the transfer has been performed. It is clear that the DA contains other circuits such as line receivers and transmitters and specific circuits of the peripheral unit, which are not essential for the understanding of the present system.

Concluding observations

From the above description the basic characteristics of the interface of the present system are seen to be the following.

1) For both directions of transfer a parity bit is generated immediately at the input of the interface channel and it is checked immediately at the output of the interface channe. Therefore a parity error is indicative of interface failures or disturbances.

2) In addition to parity control networks, on the side of peripheral unit, control networks which check the logical correctness of the received information are provided. A logical error is indicative of a failure upstream of the input of the interface or of a programming error.

3) Two distinct means of error signalling are provided: an exchange of signals CSI and CSO for signalling logical errors, and the same exchange, accompanied by signal EV3 for signalling parity errors.

4) The mechanisms for recognizing error signals are organized in such a way as to suspend the transfer sequences when an error is signalled to the central processor, in case of both logical and parity errors, detected on the character control or on the service information (commands, addresses or status), provided that they are not simple data to be read or written. On the other hand, when an error is present on data, the recognition of the error signal is delayed until the end of data transfer.

WHAT WE CLAIM IS:–

1. A data processing system comprising a central processor and a peripheral unit connected by an interface which comprises an information bus and signal lines and on which information transfer is performed by the passage of control information from the central processor followed by the passage of command/service information or data information, the central processor and the peripheral unit each having a parity generating network for generating a parity bit for information being placed on the information bus and a parity checking network for checking the parity of information being received from the information bus, the peripheral unit also having a logic checking circuit for identifying logical errors in at least the command information, error signalling means in the peripheral unit for signalling over the signal lines an error signal if an error is detected in information received over the information bus and for further signalling a parity fault if the error is a parity error and error monitoring means if the central processor for determining, from the parity checking network in the central processor and the error signal and parity fault signal (if any), the occurrence of an error and whether the error is or is not one prejudicial to the operations which are to follow immediately.

2. A data processing system according to claim 1 wherein the error monitoring means distinguishes between interface parity errors in prejudicial information, interface parity errors in non-prejudicial information, and logical errors in command information.

3. A data processing system according to either previous claim wherein the peripheral unit includes first and second registers for receiving control information and other information respectively the parity checking network in the peripheral unit comprises first and second decoders for checking the parity of information in the first and second registers respectively, and the logic checking circuit identifies logical errors in the information in the second register.

4. A data processing system substantially as herein described with reference to the accompanying drawings.

M.G. HARMAN,
Chartered Patent Agent.

Printed for Her Majesty's Stationery Office, by Croydon Printing Company Limited, Croydon, Surrey, 1978.
Published by The Patent Office, 25 Southampton Buildings, London, WC2A 1AY, from which copies may be obtained.