Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

Portable PCM analyzer

Gheorghe Daniel Popa[1], Marius Oteşteanu[1]

Abstract – The purpose of this paper is to find a solution to implement a portable PCM analyzer composed of two blocks: a small PCM interface module and a usual PC, which runs a graphical user interface software. The PCM interface module, based on a digital signal processor (DSP), uses a USB interface to the host PC and on the PCM line side uses a single chip specialized circuit to interface multiple PCM flows.

Keywords: PCM, E1, digital, telephony, transmission, switching, analyzer, USB.

1. INTRODUCTION

PCM E1 is the first hierarchical level of multiplexing in the European digital telephony. The PCM flow is structured in frames, each frame carrying 32 *time slots* (TS) of 8 bits, with a bit rate of 2048 kb/s. 30 of the 32 time slots are used to carry voice channels for telephony, one is used for frame synchronization and another one for signaling, as shown in Fig. 1.

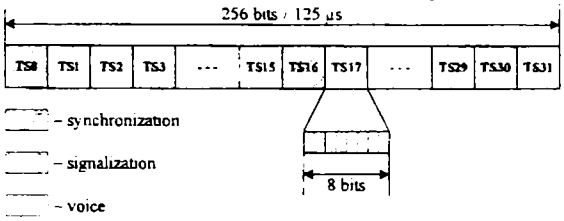

Fig. 1. The structure of a PCM E1 frame

The first time slot, TS0 carries the frame synchronization, which is realized using two signals: FAS (*Frame Alignment Signal*) and NFAS (*Not Frame Alignment Signal*) transmitted in consecutive frames, as required by [1]. Each of the 30 voice channels can transmit voice or other data at 64 kb/s. The voice signal, having a bandwidth of nearly 4 kHz (0.3–3.4 kHz) is sampled at 8 kHz and each sample, having a 12 bit precision, compressed to 8 bits according to the A-law, is transmitted in a voice channel TS each frame. The signaling information associated with the 30 voice channels is transmitted in TS16 as follows: in each frame the TS16 carries the signaling associated with two of the 30 voice channels. Therefore, it takes 15 frames to transmit the signaling associated with all the voice channels of the PCM flow. Based on this way of transmitting the channel associated signaling, the PCM frames are organized in signaling multiframes. Each multiframe is composed of 16 consecutive frames: the 15 frames carrying, in TS16, the signaling information for the 30 voice channels are preceded by a frame containing in TS16, the multiframe alignment signal (mutiframe synchronization).

PCM E1 format is used both in digital transmissions and switching, and therefore the PCM flows are the inputs and the outputs of a digital switching center. As recommended by ITU (CCITT) [2], in order to avoid possible long sequences of "1" or "0" on the transmission line, the PCM flows are encoded with the HDB-3 line code, which represents the 2 binary symbols on 3 voltage levels and limits to 3 the number of consecutive "0" transmitted. So, in a switching center, the outgoing PCM flows are HDB-3 encoded, while the incoming flows must be decoded.

The purpose of the paper is to find a solution for a small, portable analyzer which inserts, as shown in Fig. 2, on a PCM E1 link between two switching centers. The analyzer must be able to receive, extract, modify information and retransmit simultaneously the two flows of the PCM E1 link.

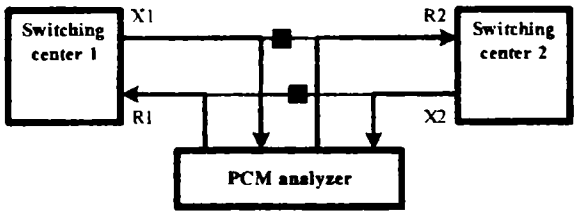

Fig. 2. External connection of the PCM analyzer

Also, the analyzer should be easily transported from one place to another, easy to install where it will be used and must be able to realize a large variety of functions such as:

- detailed analysis and interpretation of the information received from the two PCM flows (signaling),
- altering information transmitted on the two PCM flows (simulate long lines affected by perturbations),
- display the information contained in all the time slots of a frame/multiframe,
- modify the information transmitted in TS16,
- extract the information from a voice channel of a PCM flow and render it on a audio device.

[1] Facultatea de Electronică şi Telecomunicaţii, Bd. V. Pârvan Nr. 2, 300223 Timişoara, e-mail: gheorghe.popa@etc.utt.ro, marius.otesteanu@etc.utt.ro.

2. PCM ANALYZERS

Various constructive solutions, implemented in a large variety of PCM analyzers can be found on the market. Concerning the number of physical blocks used in the PCM analyzer, two main solutions can be distinguished:

- single block PCM analyzers and
- [illegible] analyzers compose [illegible] two [illegible] oc s: a usual P [illegible] (or laptop) and a module which realizes the interface between the PCM flows and the computer.

Single block analyzers, appeared the first. The main disadvantage of single block analyzers is that they are large, so inconvenient to trans ort from one lace to another. The first analyzers used a front panel with a display and many buttons, which allowed the user to select the desired function of the analyzer and to see the result. As microprocessors developed, the single block analyzers developed too, and they began to integrate IBM computers, with an operating system (Windows), while the front panel was looking as a usual laptop. Even with these enhancements, single bloc analyzers continued to have relatively large dimensions and considerable weight.

The analyzers composed of two blocks appeared later, only after the development of computers. In the case of the analyzers composed of two blocks, supposing that a PC or laptop can be found anywhere, the interface module is the only block which must be transported, and its dimensions are much smaller than those of a complete single block analyzer. In most cases, the two block analyzers use a parallel interface between the PC and the module which realizes the interface with the PCM flows.

A third solution for a PCM analyzer would be a hybrid between the single block and two blocks analyzers, considering that the PCM interface module of a two blocks analyzer can be implemented on a PCI or ISA card. Both, PCI and ISA interfaces can provide transfer rates higher than those needed by the PCM analyzers. From the transportability point of view, this solution is better than a normal single block system, because, considering that a PC with free PCI or ISA slots[2] is not hard to find at destination, it would be enough to transport the PCI or ISA card containing the PCM interface module. As compared with the two blocks analyzers, the analyzers with PCI or ISA interface present a higher difficulty degree in attaching/detaching the PCM interface module to/from the PC. In this case, in order to attach/detach the PCM interface module, the PC must be turned off and also it is necessary to open the PC chassis for the insertion/extraction of the PCI or ISA card.

The proposed solution is an analyzer composed of two blocks, having a DSP based interface module, which is also able to process the data from the two PCM flows and communicates with the host PC. A grafical user interface, which runs on a Windows operating system, is implemented on the host PC.

[2] ISA slots are less present on the PC produced in the last years

3. PC INTERFACE

The PCM interface module and the host PC comunicate through an USB interface, as shown in Fig. 3.

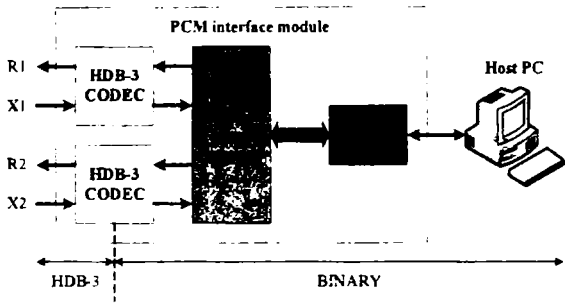

Fig. 3. Block diagram of the PCM analyzer

The USB interface between the PC and the PCM interface module was preferred, because in the last years the USB interface is used by more and more systems and is about to replace the parallel and serial RS232 interfaces in most applications. Table 1 compares the USB, standard serial port and parallel interfaces, with obvious advantages for the USB, which explains why USB is preferred instead of the other two older interfaces [3].

Table 1. Comparison between standard serial, parallel and USB interfaces

	Serial	Parallel	USB
Transfer speed	115 kb/s	115 kB/s (SPP), 3MB/s (ECP/EPP)	1,5 Mb/s (v1.0) 12 Mb/s (v1.1), 480 Mb/s (v2.0)
Plug & Play	No	No	Yes
Cable length	3 m	1.8 m	5 m
Number of devices	Up to the number of ports available on the PC	Up to the number of ports available on the PC	Up to 127 devices on a single USB bus
Bus power	No	No	Can provide up to 500 mA at 5V

The USB interface provides a transfer rate of up to 12 Mb/s, for version 1.1 and up to 480 Mb/s for version 2.0 [4], while the standard serial port only provides 115 kb/s and the parallel port can support at most 3 MB/s[3] in the EPP/ECP mode. Another important advantage of the USB over the standard serial and parallel interfaces is that the USB interface is *Plug & Play*, while the other two interfaces are not.

[3] The transfer rates for the parallel interfaces are expressed in Bytes per second, while the transfer rates for serial interfaces are expressed in bits per second.

4. GRAPHICAL USER INTERFACE

On the host PC runs a software application which implements the graphical user interface. The application is realized in Visual C++ and, as shown in Fig. 4, presents 5 menu entries: File, View, Help and one menu entry for each of the two PCM flows.

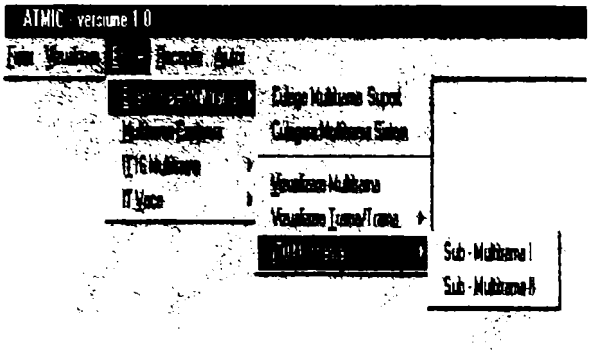

Fig. 4. The graphical user interface

The menu entries for the PCM flows present multiple submenus to allow the user to select the desired function of the PCM analyzer.

5. PCM INTREFACE MODULE

The main tasks that must be performed by the PCM interface module are:

- at the receiving flow, to recover the clock and data from the received signal and to transform the HDB-3 signal from the E1 line in a binary signal, which can be handled by a processor;
- to search for the frame/multiframe synchronization, to extract from the received PCM flows the data required by the host PC and to modify the data in the PCM flows as required by the host PC;
- at the transmitting flow, to perform the HDB-3 encoding of the binary signal that must be transmitted to the E1 line.

A possible example of modifying data in the PCM flow is long transmission lines simulation, when some bits of the PCM flow must be altered periodically in order to produce CRC errors, or frame synchronization errors.

Extracting data from the PCM flows is sometimes a very simple process, data being transmitted to the host PC 'as is' (for example when the analyzer is used to render a voice channel on an audio device, or to display a frame/multiframe). But when the analyzer is used for the interpretation of signaling associated to a specific channel, the PCM interface module must provide to the host PC, processed data, because the user must find what frequencies are transmitted on the voice channel, while specific codes are transmitted on the signaling channel. These frequencies can be determined by computing fast Fourier transforms on the data transmitted in the voice channel.

The interface between the PCM lines and the DSP may be implemented in two ways, depending on where the frame/multiframe alignment of the data is performed.

A solution is to use an E1 *Line Interface Unit* (LIU) specialized circuit which performs the HDB-3/binary and binary/HDB-3 conversions, and let the DSP to perform the frame/multiframe alignment of the data.

The other solution is to use a *Single Chip Transceiver* (SCT), another specialized circuit which is composed of a LIU and a framer. In this case the DSP does not need to search for frame/multiframe synchronization and the remaining execution time may be used by other operations running on the DSP.

On the market there are single chip integrated circuits, containing up to 32 LIUs. Complete single chip transceivers are also available, with up to 8 SCTs. The operation of the existing LIU circuits can be configured by software, by programming some control registers, or by hardware, by setting some configuration pins of the circuit to logical "0" or "1". All the LIU circuits on the market support the software control, while the hardware configuration is supported only by a few circuits, which integrate less than 4 LIUs on the same chip. The SCTs, being more complex than the LIUs, are only software configurable.

5.1. PCM INTERFACE MODULE IMPLEMENTED WITH LIU

When a simple LIU circuit, which only recovers the clock and data from the received signal and performs the HDB-3 encoding/decoding, is used between the PCM line and the DSP, the last one must take care of the frame/multiframe alignment of the data. In this case the PCM flows will always need to pass through the DSP. The block diagram in Fig. 5 shows the main blocks of the DSP implied in the transit of the PCM flows.

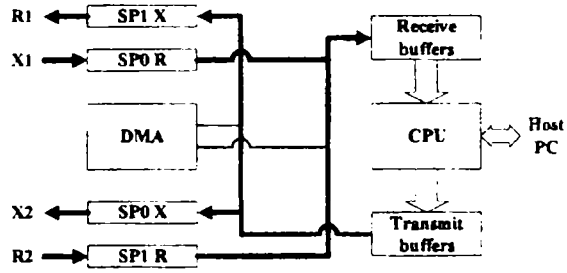

Fig. 5. The main blocks of the DSP used in the transit of the PCM flows

The transit of a PCM flow through the DSP is illustrated in Fig.6. The method used to pass the PCM flow through the DSP is a double "ping-pong" buffering method, which assumes, for each PCM flow the existence of 2 receive and 2 transmit buffers. Each of the 4 buffers has a width of 512 bytes.

The DMA controller transfers the data from the receive serial port (SP R) to a receive buffer until it fills this buffer. Then it continues to transfer the received data to the second receive buffer, while the CPU may start to process the data in the first receive buffer and transfer it to a transmit buffer.

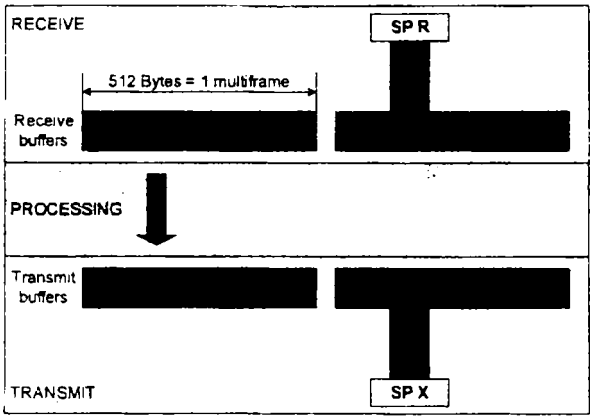

Fig. 6. The transit of the PCM flow through the DSP

While the DMA controller fills the second receive buffer, the CPU finishes to process and transfer the data from the first receive buffer, and when the second receive buffer is filled, the CPU and DMA switch the receive buffers (i.e. the DMA controller will continue to transfer data to the first receive buffer and the CPU will start processing the data in the second receive buffer).

On the transmit side, when the CPU has filled a transmit buffer, it will switch to the other transmit buffer, and the DMA controller may start transferring the data to the transmit serial port (SP X). The buffers continue to be switched between CPU and the DMA controller until the application is stopped. In this way it is ensured that the CPU and the DMA controller will never access simultaneously the same buffer, neither on the receive side, nor on the transmit side. Therefore it will not be possible to have memory access conflicts between the CPU and the DMA controller.

5.2. PCM INTERFACE MODULE IMPLEMENTED WITH SCT

The SCT circuits offer the possibility to extract information from the PCM flows, or to modify the information in certain time slots, without passing the PCM flows through the DSP. In this case the DSP will only receive the two PCM flows, and in certain situations it will not even be necessary to receive them, because some information extracted from the PCM flows can be accessed by reading some status registers of the transceivers. Some examples of information obtained from the transceiver's status registers would be: the signaling associated to a specific channel, the information contained in a specific time slot, alarms and so on. Also when some information in the PCM flow, such as the signaling associated with one or more channels, or the information contained in a specific time slot, must be modified, it can be done by the use of control and status registers of the transceiver. The only situation when the PCM flow really needs to pass through the DSP would be in the case of the simulation of long transmission lines, when the information in the PCM flow must be altered in order to simulate different levels of transmission quality.

Concerning the software control, most of the SCTs support both, serial and parallel control. The parallel control can be realized either through Intel or Motorola multiplexed/non-multiplexed bus. While in certain situations the serial ports of the DSP must be used for the real-time receiving and/or transmission of the two PCM flows, it would be better to choose the parallel bus for the software control of the transceivers. Therefore the block diagram of a PCM interface module should look as in Fig. 7.

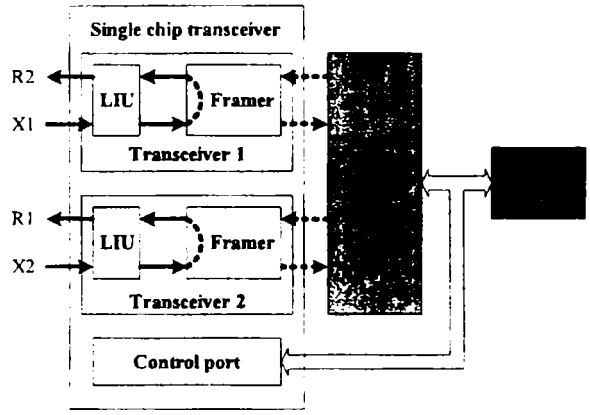

Fig. 7. Block diagram of the PCM interface module with SCT

5.3. SOFTWARE IMPLEMENTATION OF THE FRAME ALIGNMENT OF THE DATA

The frame/multiframe alignment of the data in the DSP is realized, as shown in Fig. 8 through some steps for each PCM flow. The frame synchronization search procedure is realized according to ITU (CCITT) recommendation [5]. First the receive buffer is filled by the DMA controller with 512 bytes (1 multiframe) obtained, one byte at a time, from the serial port, which receives the PCM flow. The problem at this point is that when the serial port starts to receive data from the PCM flow, there is no frame or TS alignment signal available, and no one can guarantee that the first bit received by the serial port is the first bit of a frame/TS. Therefore the bytes received by the serial port will have in the most significant positions the last n ($0<n\leq 8$) bits of a TS, and following them in the less significant positions the first $8-n$ bits of the next TS.

The DSP will load then the first byte of a receive buffer in a register and check if it matches with the FAS (i.e. the 7 LSBs should be 0011011). If they do not match, the register is left shifted by 1 bit and the LSB will be loaded with the first bit of the next byte in the receive buffer. The register will be compared again with the FAS and if they still do not match the 1 byte window will continue to slide bit by bit through the receive buffer until the frame alignment signal is found.

When a byte containing the FAS is found the window's sliding stops, and the DSP advances 32 bytes from the position where the FAS was found, checks if the NFAS (i.e. a x1xxxxxx word) is present in this position, then, if it is, the DSP will advance

another 32 bytes from this position and if the FAS is found again, the frame synchronization is assumed found. If either the NFAS or the second FAS are not found at the expected positions, the process of sliding window continues from where it stopped, ignoring the false FAS.

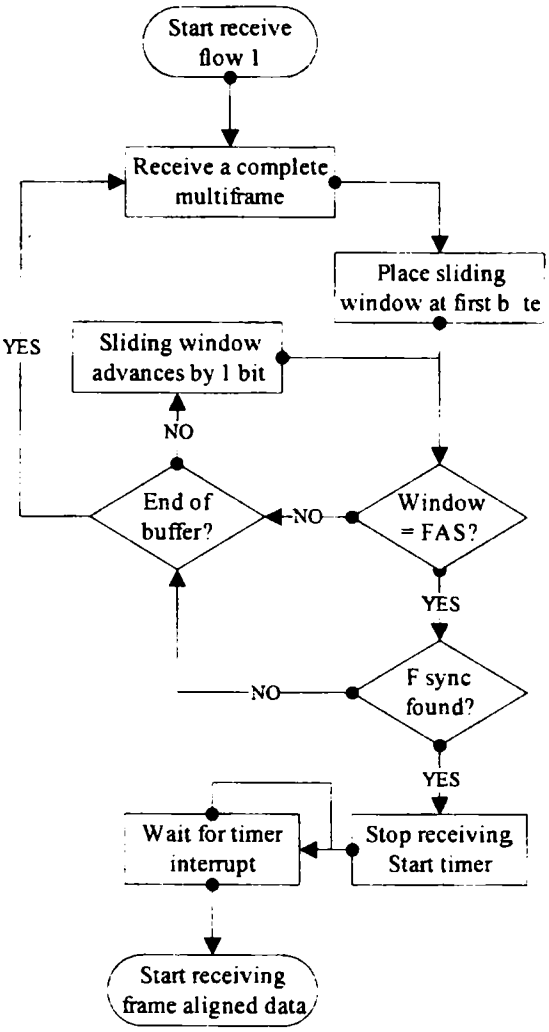

Fig. 8. The frame alignment of the data in the DSP

When the frame synchronization is assumed found the DSP computes how many bits at the beginning of the receive buffer are belonging to an incomplete frame. While the CPU performs the operations described above, the serial port continues to receive data from the PCM flow and the DMA controller moves each received byte to the second receive buffer. The CPU finishes the synchronization search in the first buffer long before the DMA fills the second receive buffer. If the frame synchronization is found in the first buffer, when the DMA controller signalizes to the CPU that the second buffer is full, the last one will suspend the activity of the serial port and of the DMA controller, in order to ignore the receiving of a number of bits equal to the number of unframed bits detected at the beginning of the first receive buffer plus one complete frame. The count of this period is realized by the use of an internal timer[4] of the DSP. When the timer finishes to count the bits which must be ignored, the CPU enables again the serial port and the DMA controller and the reception restarts, this time with frame aligned data.

The multiframe alignment can be realized in a similar way, but in this case the procedure simplifies a lot. The 1-bit-step sliding window is not necessary in this case, the CPU having only to search for the multiframe alignment word in the 16-th TS of each frame in the receive buffer.

After the frame and multiframe alignment of received data is done, the CPU will continue to verify the frame/multiframe synchronization in the received data and after extracting/modifying the data as required by the host PC, it will transfer the received multiframe to a transmit buffer.

In order to perform the frame/multiframe alignment of the data in both the PCM flows, after the frame and multiframe alignment of the first PCM flow are performed, this PCM flow is placed in a passive receiving state (i.e. the data continues to be received normally, but is not processed and transmitted), then the frame/multiframe alignment of the data is performed for the second PCM flow. When the frame and multiframe alignment is achieved for the second PCM flow too, the processing and the transmission of the data is started for both the PCM flows. The passive receiving step for a PCM flow is necessary in order to avoid perturbing the frame/multiframe alignment of the data on the other PCM flow.

6. RESULTS

The DSP development board used in experiments was a TMS320C6711 *DSP Starter Kit* (DSK) produced by Texas Instruments [6]. The main characteristics of the DSK used are:

- TMS320C6711 floating point DSP with 8 highly independent functional units (including 6 ALUs and 2 multipliers). 32 32-bit general-purpose registers, 4 kB program cache, 4 kB data cache, 64 kB internal RAM/cache, running at 150MHz and able to execute 900 million floating point operations per second;
- the main peripherals of the DSP are: 16 cannel EDMA (*Enhanced Direct Memory Access*) controller, parallel port for interface to a host PC, external memory interface, 2 bidirectional serial ports and 2 32-bit timers;
- 4 MB external 100 MHz SDRAM;
- 128 kB external flash EPROM .

Code Composer Studio (CCS) is a software provided by Texas Instruments with the DSK, to help the users to create and to debug applications for the DSK.

The size of the program loaded to the DSK, in the case of the PCM analyzer, is lower than 50 kB,

[4] When programming the timer, the execution times of the interrupt routines executed after the second receive buffer was filled by the DMA controller, should be taken into account, since they are not negligible with respect to the duration of a bit in the PCM flow (<500 ns).

therefore it can be easily loaded to the external flash of the DSK. While implementing the software frame alignment of the data, some problems occurred concerning the execution time, which became unacceptable when 2 PCM flows were passed though the DSK. The execution time of the data processing interrupt routines exceeded with very little the maximum execution time, which would allow a real time transit of the data in both the PCM flows. It was obvious that the application was far from being optimized at this moment. The main reasons which caused such an undesirable execution time were:

- the default parameters for the C compiler used by CCS were non-optimized,
- the data transfers were realized one-byte-at-a-time and
- the buffers were placed in the external memory.

By optimizing the C compiler's parameters, an approximately 6 times shorter execution time was achieved, and when the data transfers were realized on 32-bits, the execution time improved further almost 4 times. When the buffers were placed in the internal memory, the execution time improved 5 times more. Therefore, by operating the above optimizations the execution time in the data processing interrupt routines improved overall by more than 100 times. Besides, these execution times were obtained with a fully C written program, and the execution times can be even further improved by writing some interrupt routines in assembler. After performing all the above described optimizations, it is obvious that a very little percent of the DSP's execution time is used for the transit of the PCM flows, and the most part of the execution time will be available for more complex processing operations, like the computation of fast Fourier transforms, when analyzing the channel associated signaling.

The parallel to USB interface circuit used was a FT8U245 produced by FTDI [7]. The interface circuit, reached a transfer speed of up to 4 Mb/s when a *Virtual COM Port* (VCP) driver was used on the host PC, and up to 8 Mb/s with a direct USB driver on the host PC.

Before using the USB interface, the parallel interface was tried, taking advantage of the parallel interface available on the DSK, but it was not possible to reach transfer rates higher than 10 kB/s, which is far from being enough for this application. This severe speed limitation was caused by the interface circuit used on the DSK, and besides the parallel interface was also used by the DSK to change information with the CCS.

For the implementation of the SCT solution a DS21Q59 quad E1 SCT circuit, produced by Dallas Semiconductor [8] was used. This circuit contains 4 independently configurable, fully bidirectional E1 transceivers. Each transceiver can perform the frame/ multiframe alignment of the data, and can be configured to make available for the DSP, in some status registers, various data from the PCM flow. The SCT is controlled by the DSP through a parallel Intel non-multiplexed bus.

7. CONCLUSIONS

The proposed solution offers a high portability degree, using the modern USB interface between the PC and the PCM interface module. Concerning the solution for the implementation of the PCM interface module, the use of the SCT on the interface between the PCM line and the DSP is recommended. This solution offers a much easier way for the DSP to access various data in the PCM flow, even without passing the PCM flow through the DSP. Therefore the DSP will certainly have enough execution time to perform the fast Fourier transforms, required to analyze the channel associated signaling.

REFERENCES

[1] ITU-T Recommendation G.704 – Digital transmission systems - Terminal equipments - General; Synchronous frame structures used at 1544, 6312, 2048, 8 448 and 44 736 kbit/s hierarchical levels. October 1998.

[2] ITU (CCITT) Recommendation G.703 – General aspects of digital transmission systems; Terminal equipments; Physical/electrical characteristics of hierarchical digital interfaces. Geneva, 1991.

[3] http://www.totalphase.com

[4] http://www.usb.org

[5]ITU (CCITT) Recommendation G.706 – General aspects of digital transmission systems; Terminal equipments; Frame alignment and Cyclic Redundancy Check (CRC) procedures relating to basic frame structures defined in recommendation G.704. Geneva 1991.

[6] http://www.ti.com

[7] http://www.ftdichip.com

[8] http://www.maxim-ic.com