Realization of a Lost-Cost USB based ISDN Terminal Adapter using an 8-bit Microcontroller

06'98
Harald Lehmann
Siemens AG, Munich
Semiconductor Group
System Solution Center

With the latest generation of Personal Computers (PC) and the new operating system *Windows'98 (and later NT 5.0*) the Universal Serial Bus (USB) will become the standard connection for additional peripherals to the PC. All devices connected to a standard serial (COM ports) or PCI bus will utilize the USB. Therefore all telecommunication devices like Modems, Terminal Adapters for ISDN, Telephones, etc. can use an USB interface to connect to the PC. This will make it easy for everyone to hook-up their PC to the *Super-Information-Highway*.
This article will give an insight in the possibilities and technical details how to design an USB supported ISDN Terminal Adapter (TA). The application, based on a 8-bit microcontroller with an USB interface on-chip, will be explained. The main goal is to keep the application design as the same or even at lower cost than traditional solutions. Everything is based on state-of-the-art technology and an actual running solution.

Function of an ISDN Terminal Adapter

An ISDN Terminal Adapter (TA) or ISDN PC-card provides nothing else than a connection from the PC to the ISDN network. An ISDN TA is usually an **ACTIVE** device which is connected via a COM port to the PC. Most of the protocol is handled in the "box", thus there is only a little CPU load on the PC.

On the other hand, is the ISDN PC card the low cost ***passive*** solution where the main functionality is done in the PC. The PC uses the internal PCI bus to connect to the ISDN PC card. These cards are very often realized as an "one-chip-solution" like with the IPAC-chip (PSB2115) from Siemens. The IPAC will handle the lower layer of the ISDN protocol. The part has integrated S-transceivers (for the ISDN S_0 lines), B-channel and D-channel protocol-controller and a microcontroller interface.

The upper layers, depending on the User interface is managed by the PC software. Slower PC's in the past were only able to get connected with other digital sites like

an internet provider which offered an ISDN dial-in or other users with an ISDN PC card, too.

Today with the increasing microprocessor speeds, the PC software can even "emulate analog modem functions" to enable the "digital" user to get connected to all the standard analog telecommunication services like fax machines, voice, Bulletin Board Systems (BBS), etc.

This shows that less complex hardware is necessary due to higher CPU performance available on the PC.

The concept of a low-cost USB TA

Any low-cost USB TA is nothing else than a ISDN PC card which is outside the PC and connected via the USB.

Simplified, the USB replaces the PCI bus as a connection to the ISDN hardware, or the ISDN protocol is passed through the USB directly to the host (PC). The ISDN protocol is fully managed by the PC.

It might be a concern that the USB connection is more erroneous than the PCI connection, but because the error rate of the ISDN lines is much higher than the USB error rate (around factor 1000) it is not a countable disturbance for the transfers. Additionally, the several layers above this physical layer are designed to cover this issue easily.

The hardware solution is simply an ISDN protocol handler (IPAC, PSB2115) and the Siemens SAB-C541U microcontroller with on-chip USB peripheral.

USB transfer types and their usage

BULK transfers are for transferring large, non periodic, bursty data with the guaranty of error-free delivery. There is no bandwidth reserved for it, and only if the bus has unused bandwidth, data will be delivered. Bulk is used for printers, scanners, etc., it is not used in this application.

ISOCHRONOUS transfers are for data which need to be periodically sent with a predictable latency on data delivery. Here the delivery is more important than the error-free data, thus the bandwidth can be guaranteed up to 90% of the max. bandwidth. Isochronous is used for audio, video and telephony. Here the ISDN data (D-channel and B-channel frames) is fully transferred using this transfer type.

INTERRUPT transfers are for small bursty, low bandwidth data with the guaranty of error-free delivery. This transfer type doesn't interrupt anyhow the system. The word interrupt is only used for the kind of devices which were using interrupts before the implementation of USB - like mice, game-controls or keyboards. The host (PC) is polling these devices and is checking for "interrupt events". The polling interval is programmable from 1ms to 255ms at Full Speed(FS = 12Mbit/sec) and 10ms to 255ms at Low Speed(LS = 1.5Mbit/sec).

CONTROL transfers are for bursty, host initiated (bus management, configuration) data delivery with a max. bandwidth guaranty of 10% and is error corrected.

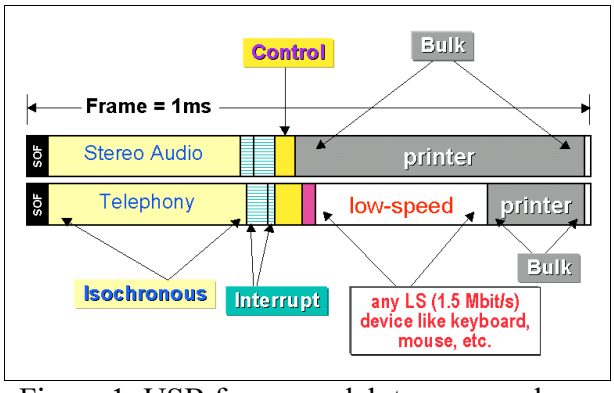

Figure 1: USB frame model, two examples

Details to the ISDN / USB data transfers

Bulk and isochronous are the only transfer types which could handle the load/bandwidth for the B-channels. The high overhead on control transfers with up to 45 Bytes per packet at FS are not suitable for this amount of data. Interrupt transfers are excluded because of their one-way, device to host (PC), data transmission.

On ISDN there is a continuous data stream which can not be guaranteed with transfer type bulk.

Isochronous is then the resulting choice for the B-channels. The lack of error correction with this transfer type will not affect the general function as already mentioned before.

It looks like, the low bandwidth of the D-channel would allow to use the control- and interrupt-transfer types. The guaranteed bandwidth of 10% for the control transfers and the polling interval up to 1ms, for the interrupt transfers, could be sufficient enough for the D-channel data stream. But the host(PC) buffer-handling for control- and interrupt- transfers will add unpredictable delays which will not fit into the constant data stream requirements for ISDN. This will exclude these transfer types, and the isochronous transfer type is used for the D-channel, too.

Layer model
ISDN-PC-card ↔ USB ISDN TA

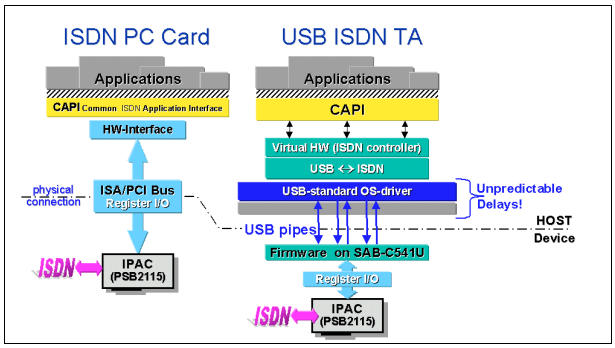

Figure 2: Layer model PC-card ↔ USB TA

Figure 2 shows the basic layer models for both ISDN applications in their structure. The ***C**ommon ISDN **A**pplication **I**nterface* (CAPI) is the standard interface for the PC application software.

Through the hardware interface the ISDN-PC-card has a direct access to the ISDN hardware. The PCI-bus establishes the physical connection. The data stream (B-channel) and ISDN controls (D-channel) can be controlled with virtual no delays via this connection.

The USB ISDN TA needs to have in comparison several layers between the CAPI and the physical access to the ISDN controls. In this application example there is a virtual ISDN controller implemented which looks to the CAPI like the HW device. At this point there is no USB visible to the Application software through the CAPI. An additional layer will do the connection to the USB adaptation of the operating system (OS). The OS has buffers , stacks, etc. and will schedule the messages to be sent through the USB pipes. On device site the firmware (running on the USB microcontroller) has implemented a buffer structure, too. Later in this article the details on this will be explained.

USB buffer delay

Due to the USB buffers delays on device- and host-site, the whole concept of a TA will be different to traditional TA's.

On an ISDN PC-card the interrupt latency is deterministic, a couple of milliseconds. On time critical tasks a direct interaction of the PC-CPU is possible.

This is not possible on USB. This Terminal Adapter (TA) have to have a certain amount of *intelligence* to handle the USB specific delays. But this intelligence should just be right enough to handle the delays, so the TA is still in the "low-cost" arena, in comparison to active TA's.

An active TA (an external "box" with additional external power supply) handles, besides the B-channel data-transfers, a large amount of the ISDN protocol (layer 2 and 3). This frees more CPU resources from the PC, which is very helpful for lower performance machines (<100 MHz).

This application shows a cost efficient USB solution comparable to an ISDN PC-card but as an external "box" with no external power supply. This is typical for USB devices!

Partitioning the ISDN into USB data

Every logical point to point connection in USB is called pipe. The ISDN B-channel data stream is called the B-channel pipe. On ISDN there are 6 data pipes. (B1_IN, B1_OUT, B2_IN, B2_OUT, D_IN, D_OUT). The data sink and source of every pipe is called endpoint (EP). As a result it looks like we need 6 USB endpoints. Additionally every device needs to have the endpoint 0, the control EP which is used by the host for bus management.

Because of the fact that the B1 and B2 pipes are equal in the function they can share endpoints. Instead of 4 independent EP's we need only 2 *shared EP's*, one for each direction. A shared EP is physically the same as a standard EP. The realization is done in software.

Used endpoints (EP):

EP0, Control (IN / OUT)
device configuration (IN / OUT), device control

EP1, Isochronous (IN)
Received B1- and B2-channel data

EP2, Isochronous (OUT)
B1- and B2-channel data transmission

EP3, Isochronous (IN)
D-channel control and D-channel data transmission and
Sync. feedback information for EP2

EP4, Isochronous (OUT)
D-channel control and D-channel data reception

Realization with the SAB-C541U and the IPAC (PSB2115) from Siemens

This Example outlines a concept for ISDN D-channel handling and B1-/ B2-channel data transfer, as well as the resulting performance requirements for an ISDN Terminal Adapter (ISDN TA) attached to the Universal Serial Bus (USB).

The USB ISDN TA does not implement higher layer ISDN protocols but provides the means to access the ISDN D- and the two B-channels. The IPAC and the SAB-C541U USB Microcontroller will be used.

The IPAC implements the four-wire S/T interface used to link voice/data terminals to an ISDN.

The C541U controller is a member of the Siemens C500 family of 8-bit Microcontrollers and provides an on-chip USB module with on-chip transceivers (compliant to the USB specification) which enables the direct connection to the USB, Full or Low Speed possible, with four EP's + control EP0.

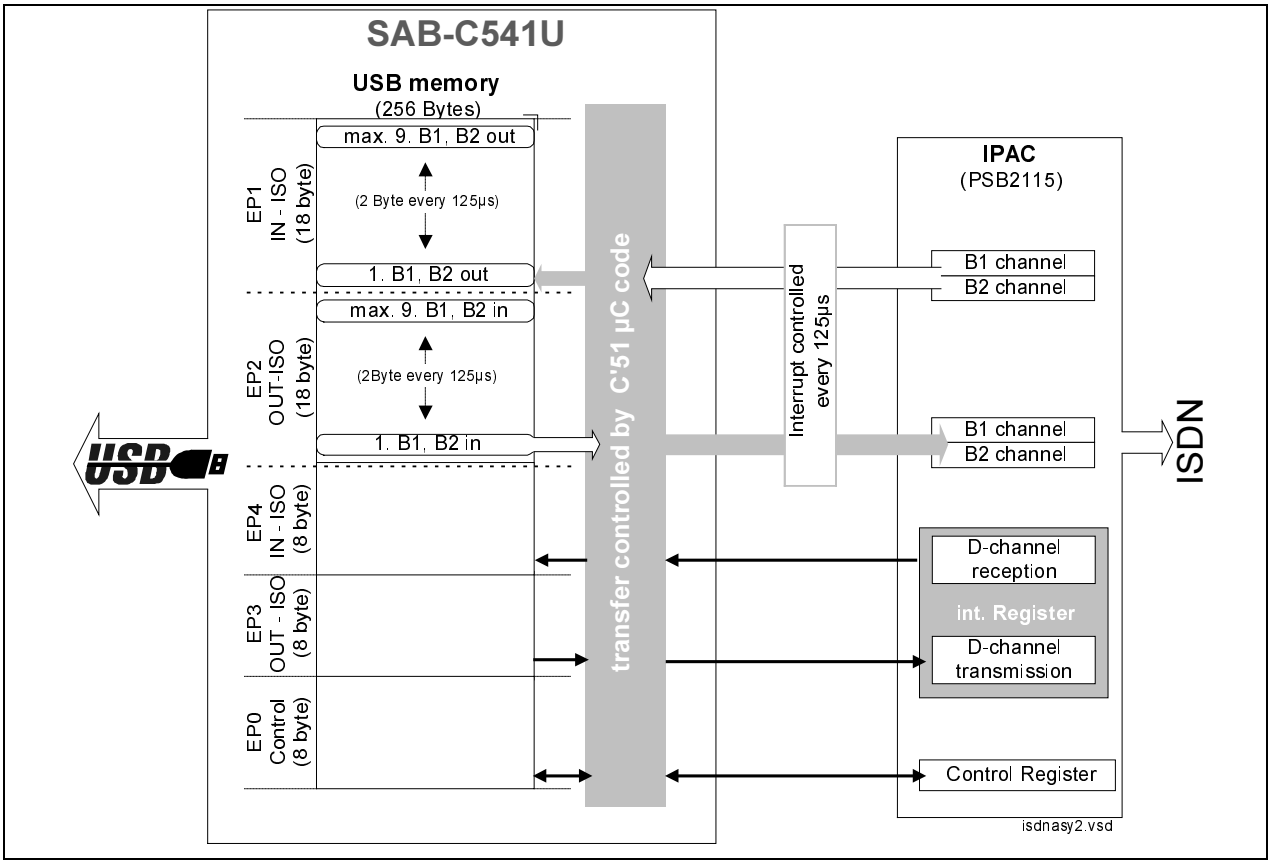

Figure 3: Block diagram SAB-C541U ↔ IPAC

Assumption:

The B-channel HDLC en-/decoding and protocol is completely handled by the host(PC).

The IPAC is equipped with the IOM-2 interface for high-speed interconnection of multiple ISDN peripherals. The data are in packet format with a frame rate of 125µs. B-channels, D-channel and the monitor-channel (control function) are transferred. With the frame strobe generated by the IPAC, it is possible to trigger an external interrupt of the C541U and read-out the internal registers via the data/address bus which is connected to the microcontroller.

The ISDN timing is asynchronous to the USB timing. Every 125µs is an IOM-2 frame interrupt generated. External interrupt 1 is used to keep track of the IOM-2 data frame timing.

The error rate of USB is smaller than the error rate of ISDN. Additional protocol layers above the USB (device ↔ PC) are protecting the isochronous data transfers.

Dual buffers for continues data stream

To realize the isochronous data stream, a 256 bytes buffer is implemented on the USB interface. For every EP a memory block can be assigned to it. In this example 8 byte blocks are reserved for the control endpoint (EP0) and the D-channel IN- and OUT- endpoints (EP3 / EP4). The size of the B-channel *shared endpoints* (EP0 / EP1) is 32 bytes. To allow the USB to have continues data transfer it should be possible to read or write into the buffer while on the other side, the opposite is happening.

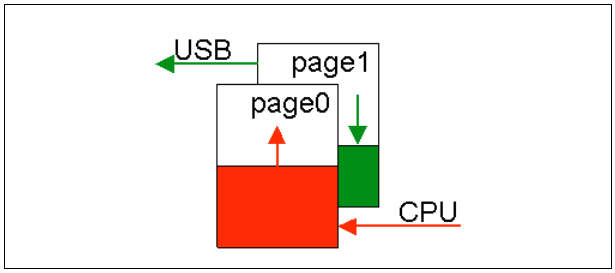

Figure 4: USB buffers access read / write

The 32 Bytes per B-channel are configured in this mode. 64 Bytes are therefore allocated to support the dual buffer mode.

To protect the data for overflow errors the buffers are swapped automatically by the Interface.

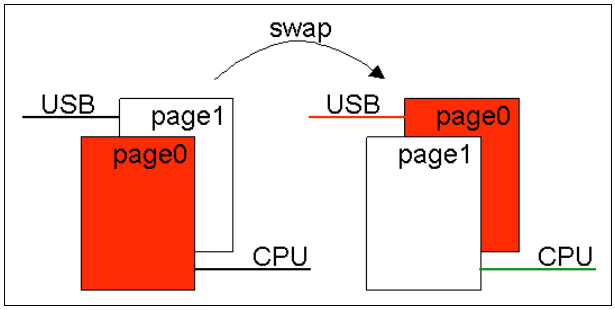

Figure 6: USB buffers swap on overflow

The firmware will be notified via a flag. On read / write attempt to an empty / full page the interface will automatically issue a NAK (**N**ot Acknowledge) to the host (PC).

Synchronization of USB and IOM-2 bus

The IOM-2 bus clock is controlled by the ISDN-switching-system and the USB clock is generated by the PC internal oscillator. Little differences between both clocks will result in fitting 7, 8 or 9 IOM-2 (125µs)-frames in one USB (1ms)-frame.

On one hand, the ideal solution would be to have both synchronized. This is possible within the USB specification, one device could be the "Clock-Master-Device". This device could tune the USB frame timing by ± 0.5% (± 63 bits out of the 12000 bit per frame). The obvious disadvantage is that only one device can be the *clock-master* in the system. It can not be guaranteed that the *clock-master* function is already used by another device.

On the other hand, it is not really necessary to synchronize both protocols. The D-channel IN-pipe will additionally be used to send a feedback-byte (figure 7, ***SYNC***) to the PC software. This is a very easy way to keep both sides synchronized.

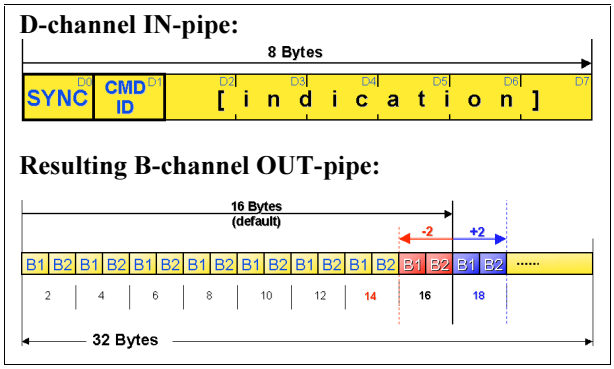

Figure 7: D- / B- channel pipes

This implies that the device is equipped with a circular buffer structure to support all the delays which are resulting in the incoming data out of the ISDN, transferred via USB into the PC´s buffer-system, the buffer management, the reaction of the PC´s drivers back through the USB into the device back to the ISDN.

Details to the circular buffers

Direction ISDN→USB (OUT-pipe)

This buffer stores the B-channel data out of the IOM-2 frames. The buffer size relates to the amount of data which can be received within one USB frame (1ms). Additional bytes are added for possible software delays which can be fine tuned on the running prototype. For simplification the IN-pipe has the same size (32 Bytes) as the OUT-pipe. Every 125µs the IPAC is generating an interrupt which triggers the microcontroller on a new IOM-2 frame.

The 2 Bytes (B1 and B2) are fetched from the IOM-2 register inside the IPAC and transferred to the circular buffer, realized in the internal microcontroller-RAM. The pointer (OUT-write_ptr) will be incremented.

Every millisecond, the USB interface generates a start-of-frame (SOF) interrupt. All the stored data from the buffer will be transmitted to the USB unit. It could be 14, 16 or 18 bytes transferred , depending on how asynchronous the frame clocks are to each other.

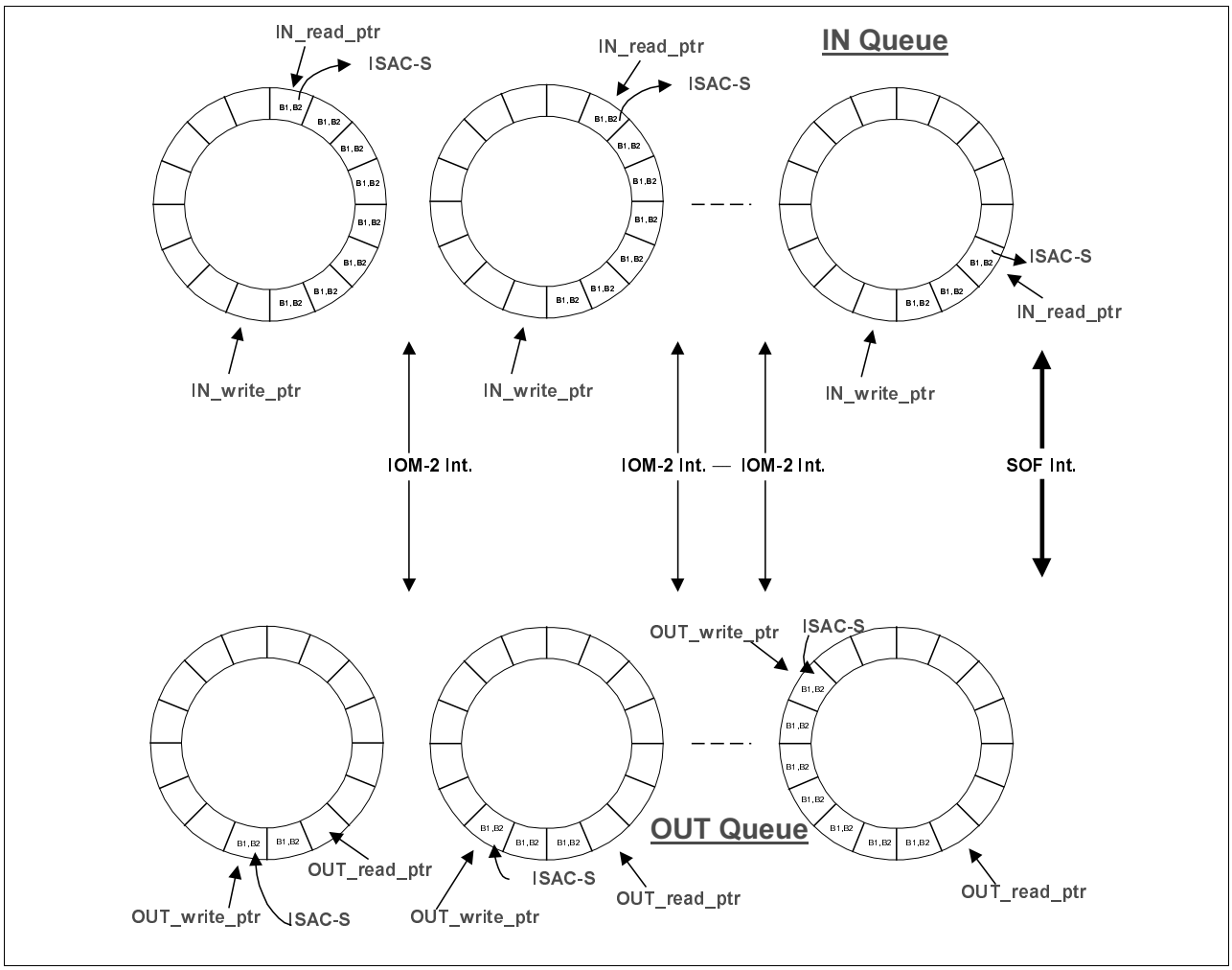

Figure 8: Circular buffers (part 1)

Direction USB→ISDN (IN-pipe)

At least half of the buffer must be filled will dummy data during the phase of initialization. With every IOM-2 interrupt, data will be transferred to the IPAC.

The pointer (IN-read_ptr) will be incremented.

Every SOF-interrupt the buffer will be filled with new Data out of the USB unit. It could be 14, 16 or 18 bytes transferred, too. But this time the PC controls the number of bytes.

The PC has to make sure that the number of receive bytes is on average the number of bytes which has to be sent to the ISDN.

The resulting delays are covered with the circular buffers. Buffer over- and under-run will cause a fatal error by stopping the continuos data stream.

This can be excluded by selecting the right buffer-size:

$$\mathbf{B}_{size} = \mathbf{2} * \Delta\mathbf{f}_{ISDN/USB} * \mathbf{f}_{byterate} * \mathbf{t}_{PCdelay}$$

$\mathbf{B}_{size}$	Buffer-size in bytes	$\Delta\mathbf{f}_{ISDN/USB}$
$\mathbf{f}_{byterate}$	Byte-rate, both B-channels (16 Kbytes / s)	max. difference between ISDN and USB frame
$\mathbf{t}_{PCdelay}$	max. PC-buffer delay the	clock in percent / 100

The max. difference between the ISDN and the USB frame clock is according the specifications, with ISDN of ±0.01% and USB of ±0.25% additional the ±0.5% if one device is doing the max. *master-clock* shift. Together this results in ±0.76% difference between both frame clocks.

The max. delay on Host-site (PC) is not accurately known yet. But it will be properly in the area of 100ms.

$$\mathbf{B}_{size} = 2 * 0.76\% /_{100} * 16\text{Kbytes} * 100\text{ms}$$
$$\approx \underline{\mathbf{26\ Bytes}}$$

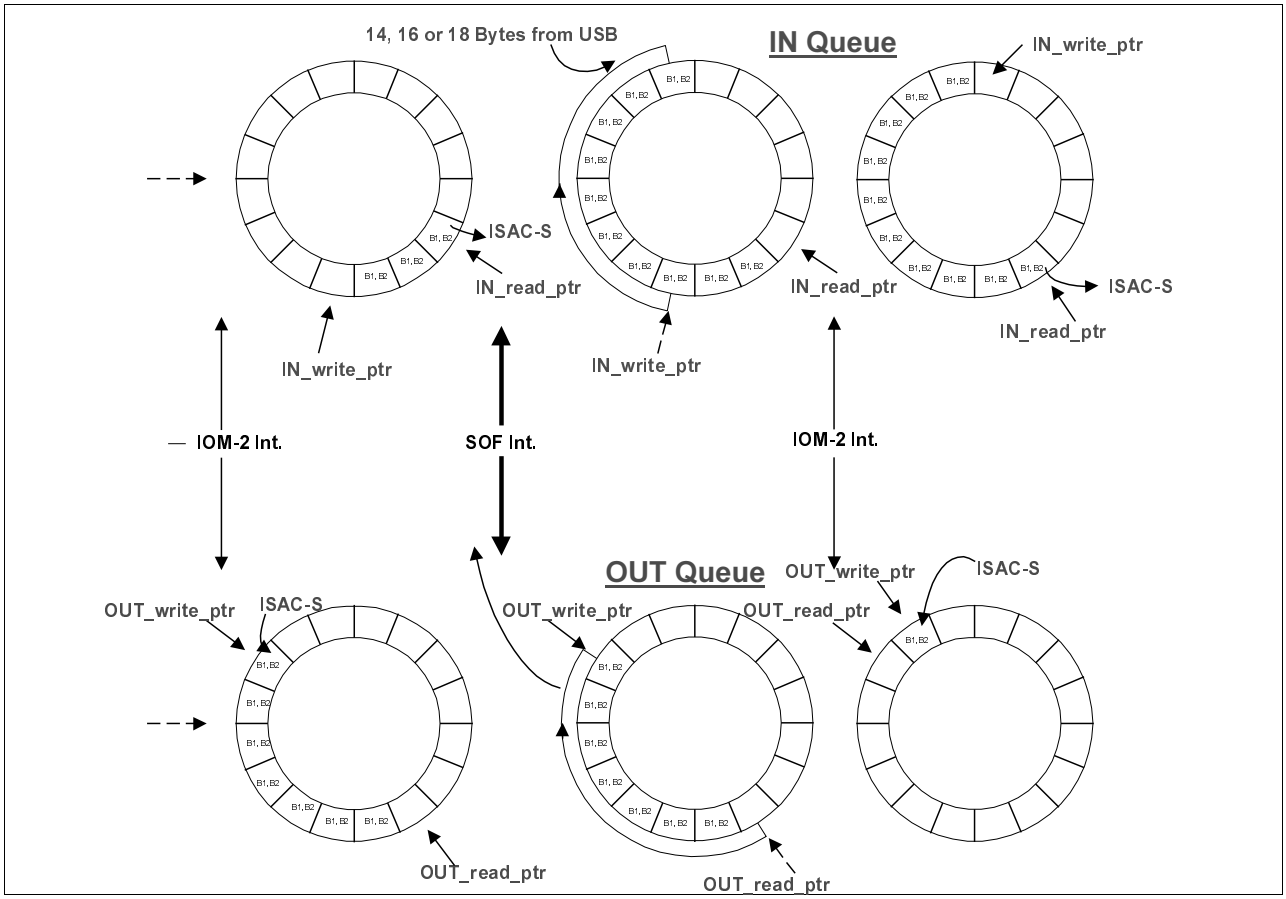

Figure 9: Circular Buffers (part 2)

This buffer size is easy to implement in the internal microcontroller RAM(256Bytes) of the SAB-C541U. This solution makes it possible to synchronize USB and IOM-2 to each other without tempering with the USB clock.

The additional software overhead on device- and host-site is a necessary trade-off to keep this solution in the low-cost range.

Performance estimation for the SAB-C541U

B-channels

Today's external TA's (usually connected via RS-232) are active units with at least a 16-bit microcontroller. But this article shows a low-cost USB solution with the use of the SAB-C541U 8-bit microcontroller. Is this part suitable for a TA? This chapter will explain in details the implementation with the C541U.

The software to the buffer management is partitioned into two interrupt service routine (ISR). One is triggered every 125µs by the IPAC writing and reading the circular buffers with IOM-2 data. The other is the SOF(USB) triggered (1ms) ISR which transfers data between the USB and the ISDN (IOM-2). The IOM-2 triggered (125µs) ISR is faster and has the highest priority to even interrupt the SOF ISR.

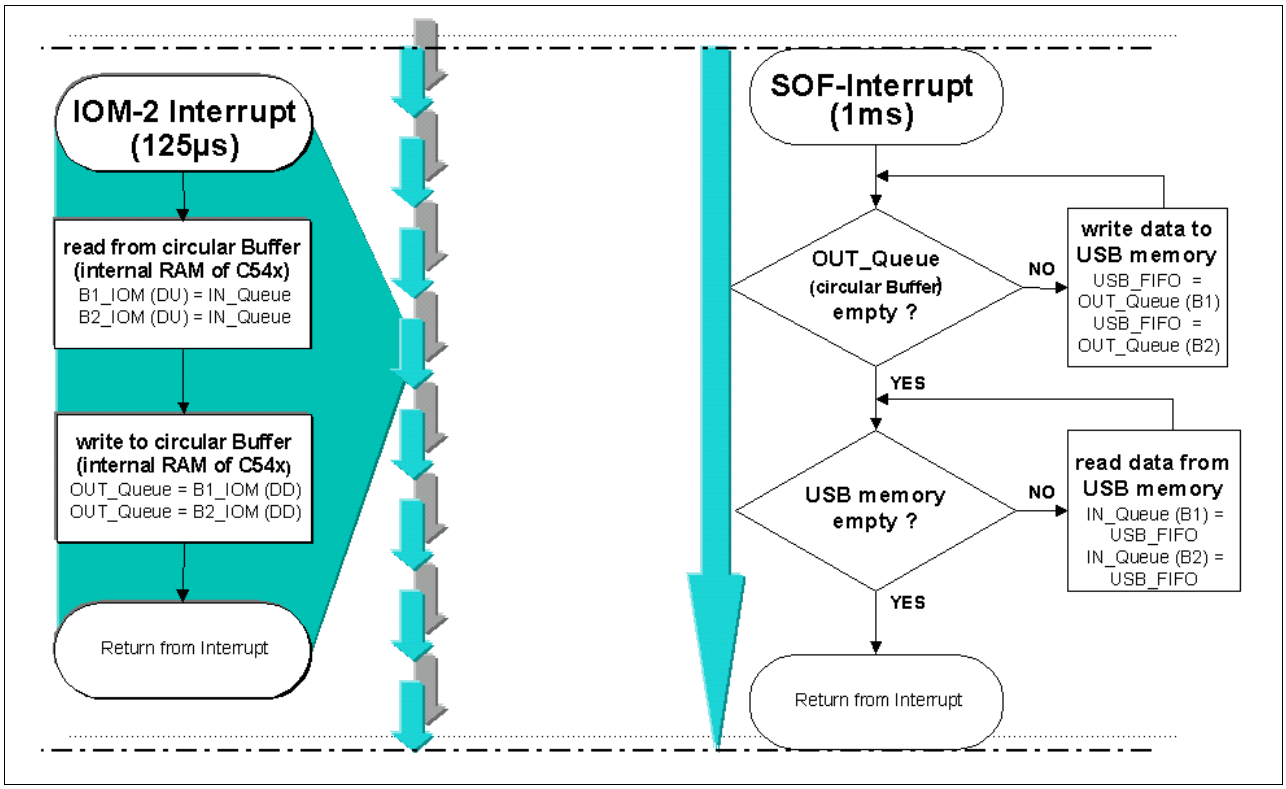

Figure 10: Flow-charts, interrupt service routines (ISR)

Both interrupt service routines handle the two B-channels (in both directions) simultaneously. The maximum runtime of the IOM-2 interrupt service routine is 64 cycles (32µs) that is ~25% of the C541U computing performance.
The USB interrupt service routine is called every 1000µs the maximum runtime of the USB int. service is ~460 cycles (230µs).

The USB interrupt service routine will be interrupted
(max. twice) by the IOM-2 interrupt service routine that is 23% of the C541U computing performance.
As a result, the C541U will spend only 49% of its computing performance on maintaining the two B-channels through the isochronous pipes.

D-channel

The following assumptions have been made to accomplish D-channel control and D-channel frame transfer:

The host software provides vendor specific commands to be issued via the USB control pipe. The isochronous pipe provides at least the capacity of 8 bytes per transfer.
The USB control pipe is used to issue commands to the USB ISDN TA

The USB isochronous pipe is used to transfer status information and received D-channel frames from the USB ISDN TA to the host. The interrupt pipe provides a transmission capacity of 8 bytes per USB frame.
The USB ISDN TA provides buffers for temporary storage of D-channel data which has been received or shall be transmitted until the data can be processed.

The D-channel frame reception and transmission is completely handled by the firmware in the USB ISDN TA

The firmware supports the following D-channel related functions via specific isochronous commands:

— D-channel controller and S-transceiver initialization
— D-channel frame transmission
— layer-1 status control
— HDLC controller reset

The USB ISDN TA firmware reports the following D-channel related information via the isochronous pipe to the host:

— received D-channel frames
— layer-1 status information
— HDLC controller status information, e.g. the completion of a D-channel frame

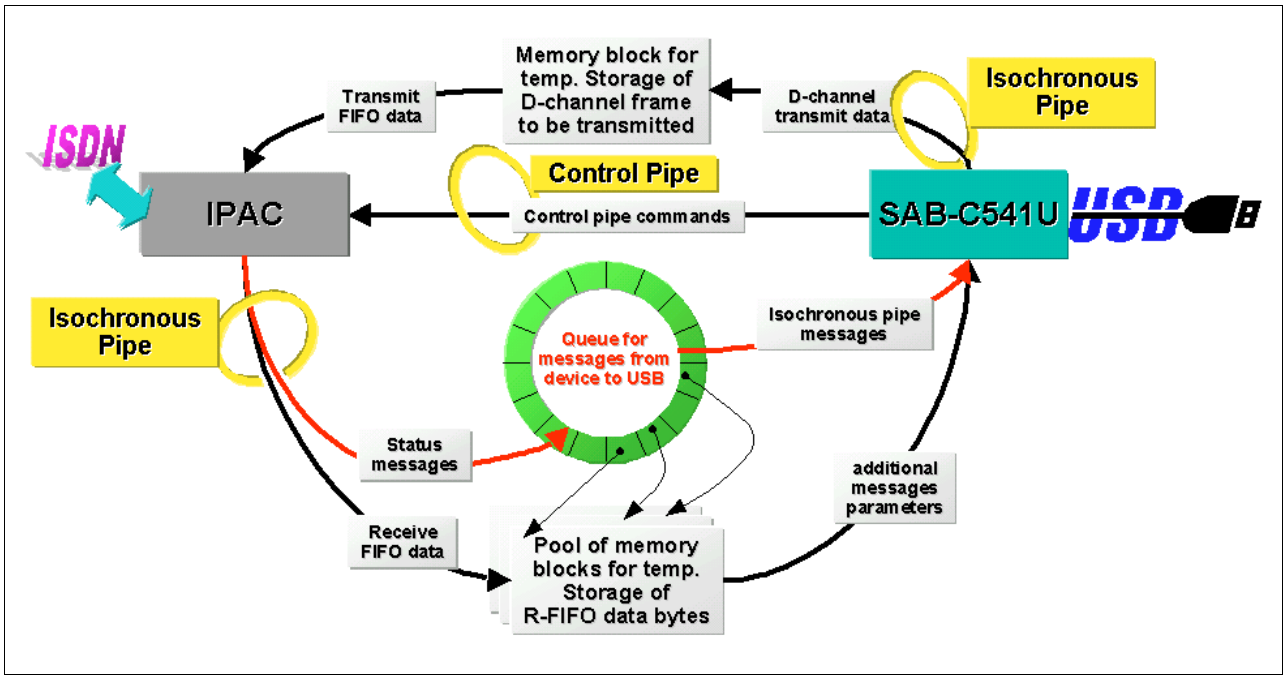

Figure 11: Data-flow for the D-channel

Data partitioning on the D-channel

For an example; 49 bytes D-channel data have to be transferred to the ISDN. The D-channel isochronous pipes are the size of 8 bytes:

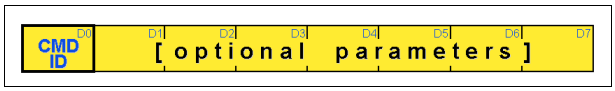

Figure 12: D-channel OUT-package

The command identification byte (CMD ID) is used to determine the kind / function of the *optional parameters* (7 bytes). Assume the CMD ID is "*send data*" then the followed 7 bytes will be the length (here: 49 bytes) and the first fragment of the whole data block. Then the next package will indicate (via CMD ID) that standard data is transferred.

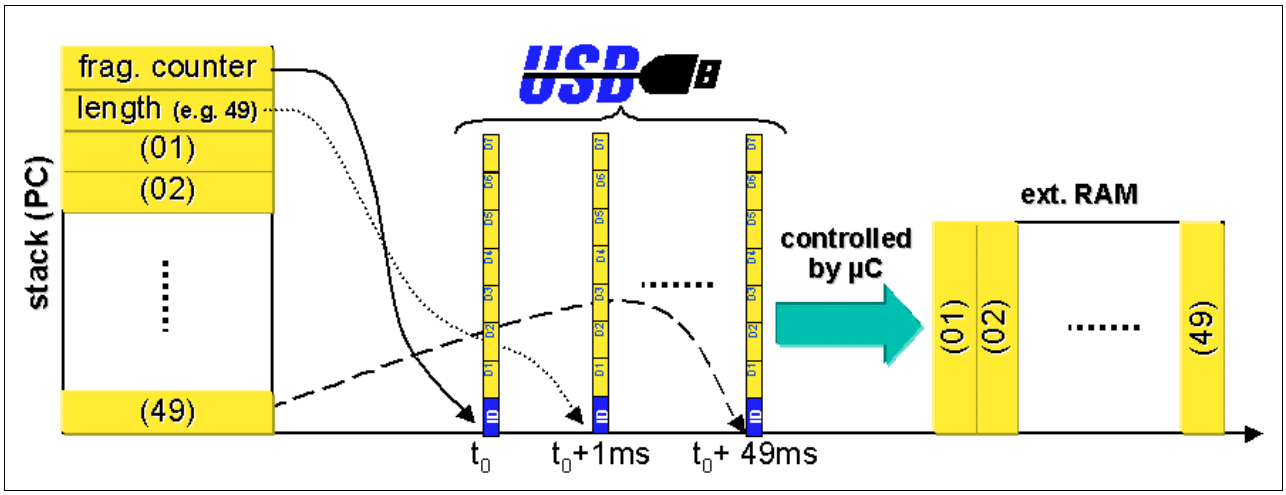

Figure 13: D-channel OUT-package , data fragmentation

The frames are received by the USB interface. The microcontroller Firmware will reassemble the fragments together again and transfers them interrupt controlled, via the register access, to the IPAC (ISDN).

Incoming data from the IPAC (ISDN) are using the following D-channel IN-pipes:

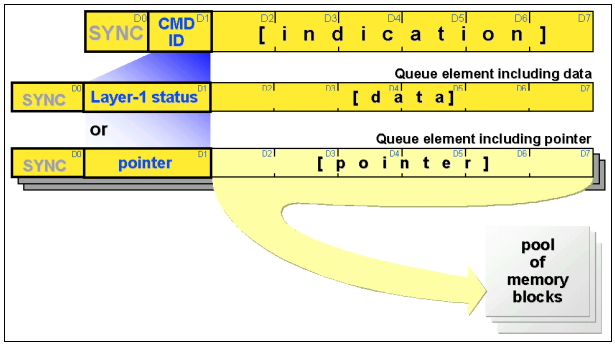

Figure 13: D-channel IN-package, example

Besides of the ***SYNC***-byte which was already explained, are the IN-packages similar to the structure of the OUT-packages.

Some layer-1 controls are handled on device site. The resulting data will be place in a "*pool of memory blocks*" which are accessed via the pointers indicated through the D-channel packages.

For the performance estimation the maximum basic rate D-channel load of 16 kbit/sec is assumed. The following functions are carried out by the firmware:

— decoding of commands received via the isochronous pipe
— copying of D-channel transmit data (fragments) from the USB controller FIFOs to the RAM buffer
— initiation of a D-channel frame transfer

— handling of IPAC interrupts, reading and temporary storage of received D- channel frames
— copying of received D-channel frames (or frame fragments) from the local buffers to the isochronous pipe
— transmission of device status notifications to the host via the isochronous pipe.

The 16 kbit/sec data-rate requires packets of 32 bytes (32 bytes = size of the D-channel transmit and receive FIFOs). The data is transferred from USB to the IPAC and vice versa in 16ms periods. In addition, the CPU-load for temporary storage of a D-channel frame has to be considered. One additional notification from the USB ISDN TA to the host shall be generated in each 16ms period, too. Instruction cycle is 500ns (@ 12MHz external) max. 5000 cycles are needed to execute the sample routines.

The C541U will spend only ~20% of its computing performance on maintaining the D-channel.

Result

Together, the transfer of D-channel messages from USB to the IPAC or and vice versa and the B-channel data transfers from USB to IPAC and vice versa, the C541U will spend approx. **70%** of its computing performance for doing the basic functionality of an ISDN Terminal Adapter.

Conclusion:

The USB will be the PC connection for the future. Every device-manufacturer in this arena is looking into or already developing devices. The big boost for this bus system will happen with the release of *Window '98.*

These kind of low-price ISDN terminal adapter with the easy to use interface as they are:

— "hot" plug 'n play
— no extra power supply
— automatic detection
— full bandwidth (128Kbyte/sec)

will make high-speed communication for everyone desirable and possible.

Remark:

This ISDN TA solution is focused on "low-cost". It is not intended to fulfill the USB *class definition, communication (CDC)*, which can be found at: http://www.teleport.com/~usb/devclass.htm.

This means that the SW driver for this solution would not be included in the OS, but is shipped separately with the ISDN TA device.

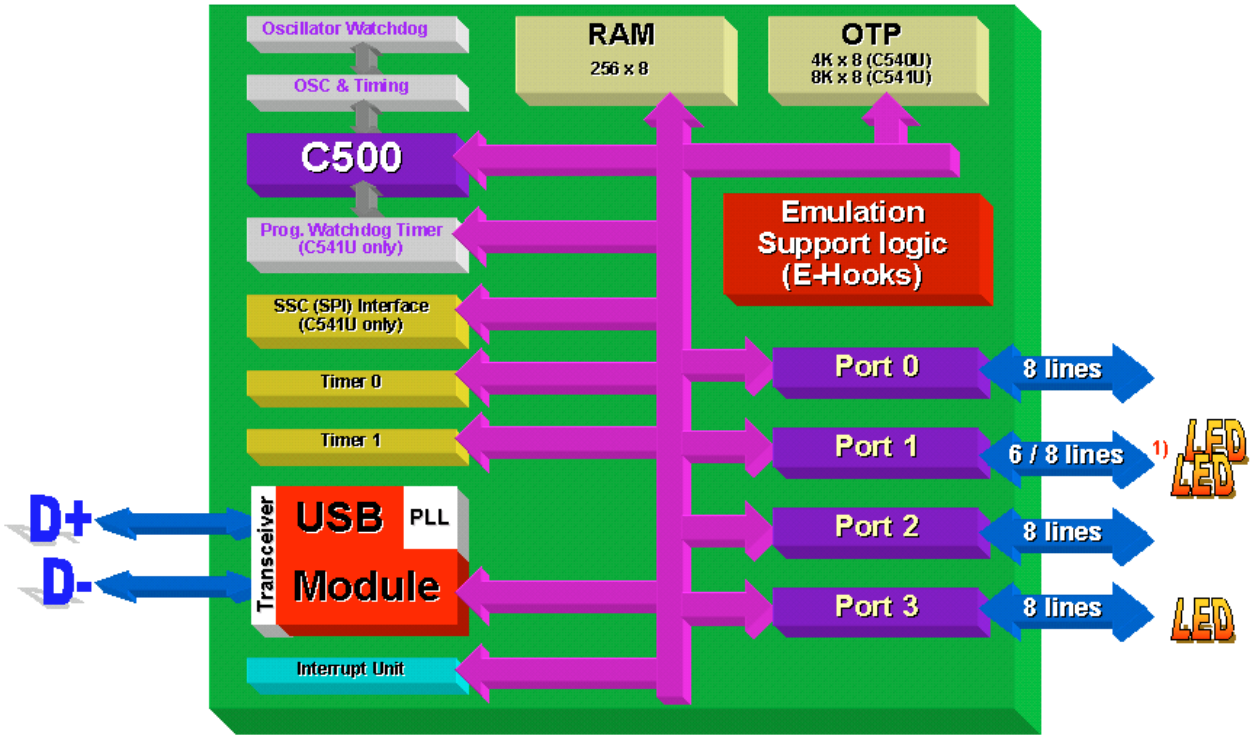

Figure 6: Blockdiagram SAB-C541U / -C540U

List of feature of the SAB-C541U /C540U

- Enhanced 8-Bit C500-CPU
- 500 ns Instruction Cycle Time @ 12 MHz CPU Clock
- Two 16-bit Timer/Counters (C501 compatible T0/1)
- [4kbytes in the C540U] 8 Kbytes OTP / 256 bytes RAM
- USB Device Core (FS/LS fully speed supported)
- Synchronous Serial Channel, Watchdog Timer [C541U only]
- LED Driver capability on 3 dedicated pins, 10 mA / 4,5 to 5,5 V
- Power supply voltage range: according to USB spec.
- Enhanced Hooks Technology for easy emulation
- 48 MHz PLL on-chip for FS
- D+ D-, on-chip transceiver
- P-LCC-44 and S-DIP-52 package
- 30 digit. I/O Ports, 32 in SDIP52

Contact:

Harald Lehmann
Siemens AG
HL CE M
P.O. Box 80 17 60
D-81617 Munich (Germany)

mc.support@hl.siemens.de