# HPC16400 A Communication Microcontroller with HDLC Support

National Semiconductor
Application Note 593
Nick Burd
July 1989



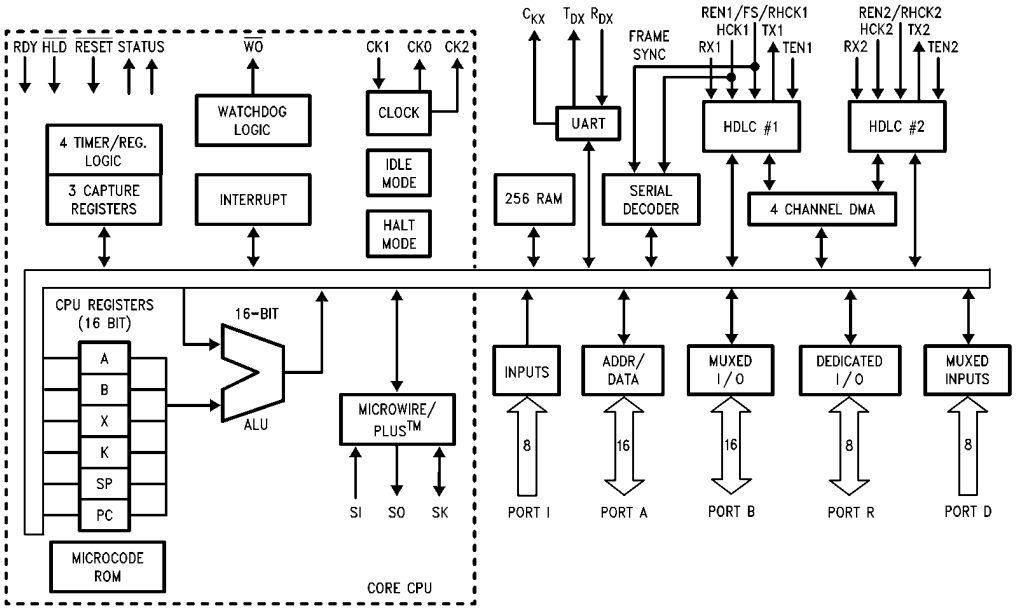


## INTRODUCTION

The HPC16400 is a communications microcontroller for HDLC based applications and is the latest in the range of High Performance microcontrollers (HPC™) from National Semiconductor Corporation. HPC is a family of 16-bit CMOS microcontrollers which feature a common core to which are added peripherals for a specific application area. In the case of the HPC16400, these include dual HDLC channels and a four channel DMA controller which make the HPC16400 ideally suited to embedded protocol processing, such as X.25/LAPB. In addition, the HPC16400 also contains an on-chip serial decoder which allows the HDLC channels to be time multiplexed onto common transmit and receive lines as used by the ISDN (Integrated Services Digital Network) Basic Rate interface. This means that together with Nationals' ISDN line interface and COMBO™ circuits, and a software package which implements the generic ISDN protocols (Q.921 and Q.931) a complete system solution for ISDN Basic Rate applications is possible.

The HPC16400 is capable of running at a maximum clock frequency of 20 MHz, and each of its HDLC channels can operate up to a maximum 4.65 Mbps data rate. A photograph of the HPC16400 chip is shown in *Figure 1.*

This article describes the features of the HPC16400, and in particular the operation of the HDLC/DMA channels and the serial decoder. As an example of how the HPC16400 would be used in an ISDN application, an ISDN terminal is described together with the features of the ISDN software package which can be used to minimize the time and effort in developing such equipment.




TL/DD/10361–2

**FIGURE 1. Block Diagram of the HPC16400**





RRD-B30M75/Printed in U. S. A.

## THE HPC CORE

*Figure 1* shows the block diagram of the HPC16400 in which the functions within the dotted line form the HPC core which is common to all HPC family members. It can be seen that the core contains the CPU as well as several peripherals. Those functions outside the dotted line are the peripherals specific to the HPC16400.

The CPU contains a 16-bit ALU and a 16-bit accumulator which acts as the source and destination for most operations. Two 16-bit address pointer registers, B and X, are intended to be used for indirect addressing of data with auto increment and decrement of the register. The K register is used to set a limit for the B register when it is either incremented or decremented with successive execution within program loops. A specific feature of the instruction set of the HPC CPU is that conditional execution of an instruction is based on a skip structure instead of the traditional conditional branch or jump. This is best illustrated through an example using the B, K and X registers described above. The example listed in *Figure 2* swaps the contents of two areas of memory in the ranges 0x4000 to 0x4FFF and 0x5000 to 0x5FFF. A single instruction is used to load the B and K registers which define the boundaries of the lower memory area, and the X register is loaded to point to the beginning of the upper memory area. The first instruction within the loop loads the accumulator with the memory word pointed to by the X register, and the X register is then incremented. The fact that a word value has been specified here means that the X register will automatically be incremented by two. If a byte value had been specified, it would be incremented by one. The second instruction in the loop is an exchange with a conditional skip which exchanges the contents of the 16-bit accumulator with the memory word pointed to by the B register, and the B register is then incremented by two. If the new value of the B register now exceeds the value in the K register, the following jump instruction will be skipped and program execution will exit the loop. If the value of the B register is less than the K register, then the next instruction is executed and the loop is continued. Judicious encoding of the opcodes for the HPC instruction set has resulted in a very efficient implementation of common constructs such as the loop just described. The register indirect instructions are encoded as single-byte instructions as well as the short jump instruction where a six bit offset is included within the opcode. The loop described above therefore generates only three bytes of program code. In total, the HPC has 54 instructions and nine addressing modes.

```
       LD BK, #4000, #4FEE   ;load B and K with start and end of lst. memory block.
       LD X, #5000           ;load X with start of second memory block.
LOOP:  LD A, [X+].W          ;get word from second block, increment X.
       XS A, [B+].W          ;exchange with word from first block, increment pointer,
                             ;skip if B>K.
       JP LOOP               ;do loop again

EXIT:   ↓   Continue program
```

**FIGURE 2. An Example HPC Program to Swap Two Memory Areas**

The HPC core contains several peripheral features. The MICROWIRE/PLUS™ is an inter-chip serial communication port which consists of an 8-bit shift register and a clock. Writing data to the microwire port when configured as a master causes the data to be loaded into the shift register and eight clock pulses generated to shift the data out. At the same time, these clock pulses can be used to clock data in from a microwire slave device such as the ADC0834 A/D converter or the NMC93C46 EEPROM.

The HPC core also contains a number of timers. A purpose of one of these timers, T0, is to provide a means for accurate time interval measurements, and when configured in this mode, it is associated with up to three capture registers which can be triggered by external interrupt inputs. Timer T1 provides a dual function as it can operate as a normal timer, or its registers can be used as two of the capture registers for T0. The timer T0 also drives the Watchdog™ logic which causes the Watchdog output to trigger whenever it is not serviced before a timeout of T0. The remaining two timers can be used to generate a variety of timing outputs.

Interrupt logic provides enabling circuitry for the numerous sources of interrupt on the HPC, and an interrupt pending register eases the processing of multiple interrupts. The HPC can be placed into one of two power saving modes by programming the Processor Status Word (PSW) register and the Halt Enable register. In the Halt mode, all processor activities, including the clock and timers, are stopped thereby reducing the power requirements of the HPC to a minimum. Recovery from the Halt Mode can either be from a Reset or from the NMI. In Idle mode, all processor activity apart from the on-board oscillator and timer T0 is stopped so that recovery from the Idle mode can be achieved with the timer T0 overflow as well as the reset or NMI functions as in the Halt mode (except that in the Halt mode recovery is not immediate as the oscillator will take tome to stabilize).

**HPC MEMORY**

All functions on the HPC chip are memory mapped. The on-chip peripherals, core registers, and on-chip user RAM (16-bit) occupy an address area between 0 and 0x1FF as shown in the memory map of *Figure 3a*. The area of user on-chip RAM in the range 0–0xBF is in the BASEPAGE (0–0xFF) of the address space, and in addition to being used as general purpose storage locations for variables, the indirect addressing mode of the HPC allows memory words in this area to be used as pointers containing the effective address of the operand. This allows many additional pointers to be created in addition to the B and X register and significantly eases the programming of many tasks.

The memory requirements in telecom applications are generally large for both program and data areas, and so the HPC16400 does not have a single-chip configuration with on-chip ROM. Instead, a 16-bit multiplexed address and data bus is brought external to the chip and is used to add program memory and additional data memory to the system in the address range 0x200 to 0xFFFF.

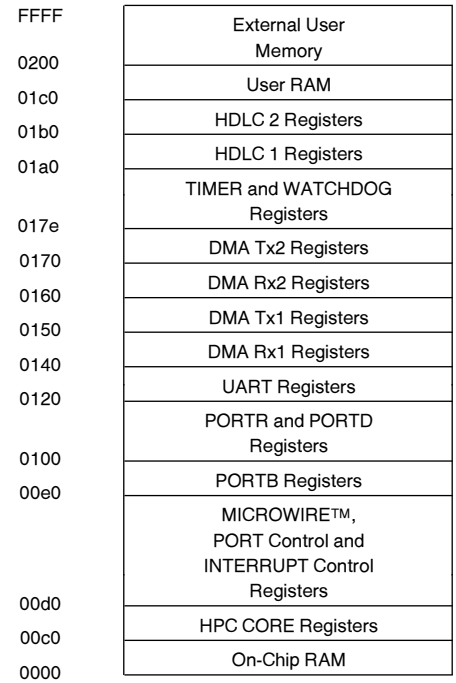



**FIGURE 3a. The Basic 64k Memory Map of the HPC16400**

The 64 kbyte address space can be expanded further by the use of bank switching. Four lines from Port B may be used to select one of sixteen banks of 32 kbytes in the address range 0–0x7FFF as shown in *Figure 3b*. In this way, the upper 32 kbytes of memory are common to all of the banks and allows a program in one bank to jump or call subroutines in other banks via this common area where the banks can be safely switched (see reference 1 for a more in-depth discussion of bank switching on the HPC). The common memory also provides storage area for global variables and stack locations when operating in a bank-switched environment. The total memory addressing capability of the HPC16400 amounts to just over 500 kbytes as the section of on-chip RAM in the range 0–0x1FF is common to all banks.

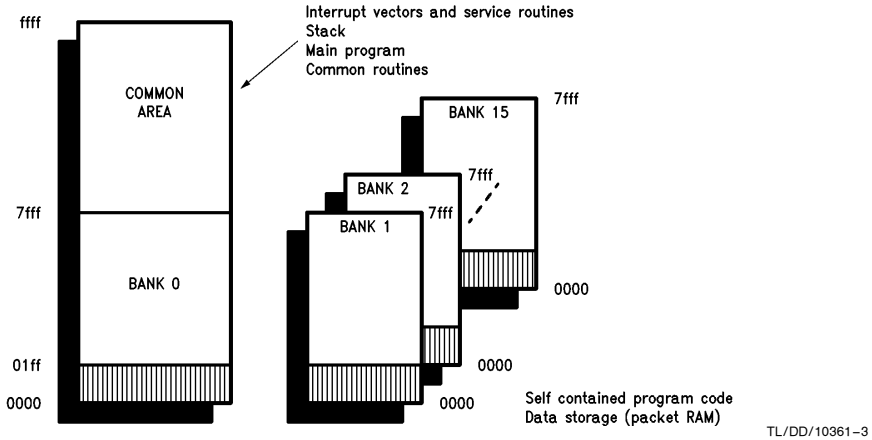



**FIGURE 3b. HPC Extended Memory Addressing by Bank Switching**

When the HPC fetches program from memory, it does so one byte at a time because the opcode encoding is byte oriented. This allows the HPC to be configured with either 16-bit external memory, 8-bit external memory for more cost sensitive applications, or a mixture of both. When operating with 16-bit memory, the HPC can access both odd and even bytes, and words on an even boundary. In 8-bit mode the HPC makes only byte accesses to external memory, and although the registers in the BASEPAGE memory of the HPC are 16 bits, they may also be addressed individually as high and low bytes thereby enabling the 16-bit architecture of the CPU to be used.

Selection of 8- or 16-bit bus mode for the HPC is achieved on reset of the processor when the ''high byte enable'' control line is sampled by the CPU. If this line is detected in a high state, the HPC enters 8-bit mode. However, if the line is detected as high impedance, as a result of it being used as a control output to select low and high 8-bit memory banks, then the HPC enters 16-bit bus mode.

### THE HDLC AND DMA CHANNELS

The HPC16400 contains two identical on-chip HDLC channels, each capable of transmitting and receiving HDLC frames transparently to the operation of the CPU. The format of an HDLC frame is shown in *Figure 4*. The frame is delimited by an identical opening and closing flag which is a unique bit pattern consisting of a zero followed by six consecutive ones and then a final zero. This pattern must not occur anywhere else within the frame and is guaranteed by a zero insertion mechanism which, after the transmission of five consecutive ones in the data stream between flags, will insert a zero before continuing to transmit data. A reverse procedure is adopted at the receiver to delete the additional zeroes. Immediately following the opening flag is an address field which identifies which equipment on the network is to receive the frame. The control field contains information, such as handshake control, which is used to control the flow of frames between communicating devices. This is followed by the application specific data, a frame check sequence which validates the integrity of the frame with a cyclic redundancy check (CRC) code, and the closing flag. The HPC HDLC channels provide automatic framing functions such as opening and closing flag insertion and deletion, zero bit insertion and deletion (also known as bit-stuffing), CRC16 or CCITT implementations of CRC checking, and abort sequence transmission and recognition. The abort sequence in this case is a modified flag consisting of a zero followed by seven ones. In addition, the transmitters can be programmed to generate flags, abort sequences, or just idle (transmitting consecutive ones) between the transmission of consecutive frames.

**first byte** **last byte**

| FLAG | ADDRESS | CONTROL | DATA | CRC | FLAG |
|---|---|---|---|---|---|
| 1 | 1 or 2 | 1 | Typically < 1024 | 2 | 1 |

**BYTES**

**FIGURE 4. The HDLC Frame Format**

A feature which helps to reduce the CPU overhead in protocol processing is the address recognition logic. Each channel has two address recognition registers that can be programmed with a byte which can be compared in a number of different ways with the first two bytes received by an HDLC channel. The different comparison modes are intended to cope with a range of different communication network addressing modes. *Figure 5* shows the logical operation of three of the four possible modes. In mode one, the second byte received after the opening flag of the frame is compared with both address registers and a seven bit broadcast address pattern (0x7F). If any of the registers match the incoming address, then the HDLC channel will continue to receive the complete frame. If no match is detected, the HDLC channel will stop receiving the frame, discard the address already received, and start to look for the opening flag of the next frame. This particular address recognition mode is useful in ISDN communications because the second byte received will be the address of the terminal equipment, such as a telephone or perhaps a PC, on the ISDN network.

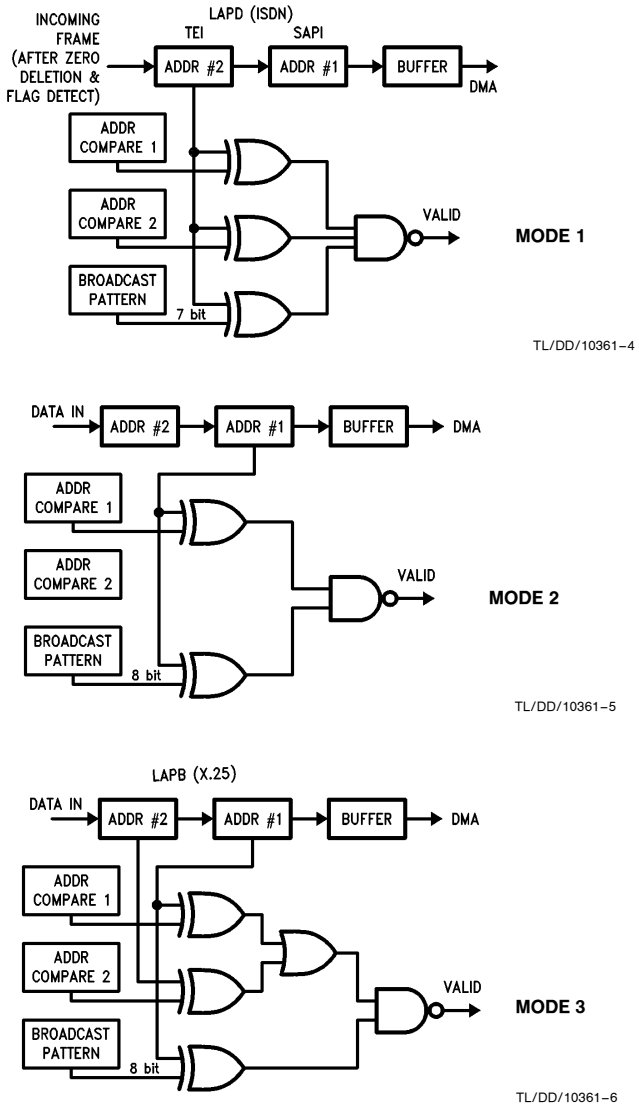



**FIGURE 5. The Address Recognition Logic for the HDLC Receivers**

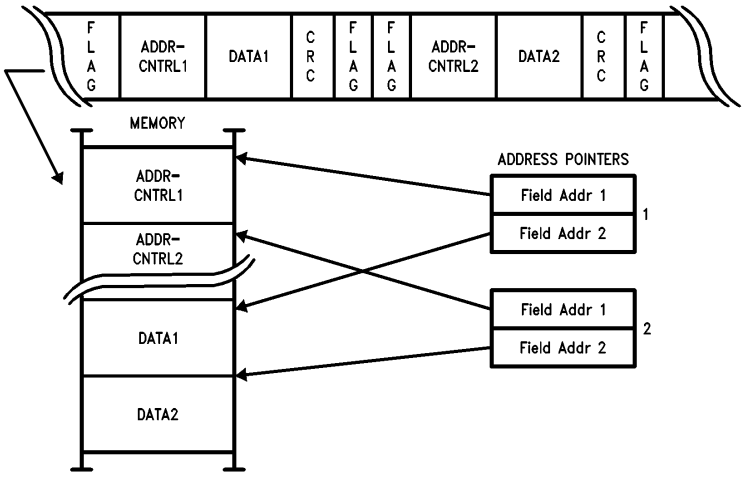
Mode two matches the first byte received with the first address register and an 8-bit broadcast address which could be used in X.25/LAPB applications. Mode three compares a 16-bit address field so that the contents of the first comparison register must match the first address byte received, and the contents of the second comparison register must match the second address byte received. Or, if the first byte corresponds to the 8-bit broadcast pattern, an address match will also be signalled to the CPU. The last mode, Mode zero, is the ''transparent mode'' in which all frames are received by the HDLC controller regardless of the address field contents. This mode would be used, for example, in a device which had to gather all information from the communications network and compute statistics about its communications loading.

Both HDLC channels are capable of implementing bit oriented protocols, such as IBMs SDLC, by programming the number of bits to be transmitted in the last byte of the information field. Further flexibility is achieved with a bypass mode which disables all of the HDLC framing functions allowing designers to implement their own byte-oriented synchronous protocols.

As mentioned earlier, all programmable features of the HPC are memory mapped, so the HDLC registers are mapped to an area of on-chip RAM above the BASEPAGE section in the range 0x1A0 and 0x1B8. Each HDLC channel has an identical set of registers, and each set contains receiver status, control, address comparison, and error status registers. In addition, there are two global registers which handle the enabling and servicing of interrupts from the HDLC channels. An interrupt can be generated whenever an HDLC channel signals an ''End of Message'' (EOM) which indicates that an HDLC frame has just been received or an HDLC frame has just finished being transmitted. Should a transmitter or receiver generate an EOM before the previous EOM has been serviced, then an overrun interrupt may be generated. All of these interrupt sources have a single interrupt service vector, and so the global registers contain bits which allow the source of the interrupt to be uniquely identified. Additional error conditions, such as reception of a bad CRC, reception of an abort sequence, or a framing error, cause bits to be set in the error status register which may also generate an interrupt, although this may lead to the generation of multiple interrupts. A more straight-forward approach would be to test the condition of the error status register once an EOM interrupt has been received.

The HPC16400 contains an on-chip four channel DMA controller. The operation of the DMA controller is closely linked to the HDLC channels because they are responsible for interfacing them to the memory. Hence, as each byte is received by an HDLC channel, it signals the DMA controller which requests and gains control of the processor bus and writes the received byte to a predetermined area of memory. Similarly, when an HDLC channel is transmitting a frame, it requests data from the DMA controller which transfers a byte from an area in memory to the HDLC channel. During DMA accesses the CPU loses control of the memory bus. However, for the HPC16400 running at 20 MHz, the CPU bus occupancy 1s only expected to decrease by 10% for an aggregate HDLC data rate of 2 Mbps. For typical Basic Rate ISDN applications the decrease is expected to be less than 2%.

The DMA channels contain several addressing features which allow convenient transmit and receive buffers to be created in memory. Each DMA channel supports a split-frame mode which allows the transmitted or received frame to be split into two sections with each section being stored in a different area of memory. In HDLC, it may be convenient to have all the address and control fields in one area of memory, and all the information fields in another. (The CRC and flag fields are stripped off or appended by the HDLC channels, and so are not present in the memory area.) In the DMA receiver, there are two pairs of address pointers, each pair pointing to the two sections of the same frame as shown in *Figure 6*. As the HDLC controller starts to receive data, the DMA channel places the first received byte in the memory pointed to by the first address pointer, and the pointer is then incremented. This continues until the number of bytes for the first segment, which can be programmed up to a maximum of 7 bytes in the DMA receiver control-status register, has been reached, at which point the contents of the second address pointer becomes the destination for the remainder of the received frame.




TL/DD/10361–7

**FIGURE 6. Split-Frame Operation for HDLC/DMA Receiver**

For the DMA channel which supports the HDLC transmitter, each pair of registers contains a single pointer and a byte counter which holds the number of bytes to be transmitted, as illustrated in *Figure 7*. When the split-frame mode is not used, each pair of registers in the transmit DMA, and each address pointer in the receive DMA, refers to a separate complete frame. This means that the HDLC receiver can receive four frames before the DMA address pointer registers need to be updated, provided the EOM is serviced after each frame to prevent an overrun interrupt.

In the previous section, the extended memory configuration of the HPC16400 using bankswitching was described. The DMA channels are capable of taking full advantage of this extended memory by a programmable field in the control-status registers whose value is written to the external bank-switch control lines during a DMA cycle. This allows the extended memory banks to be used for storing frame information.

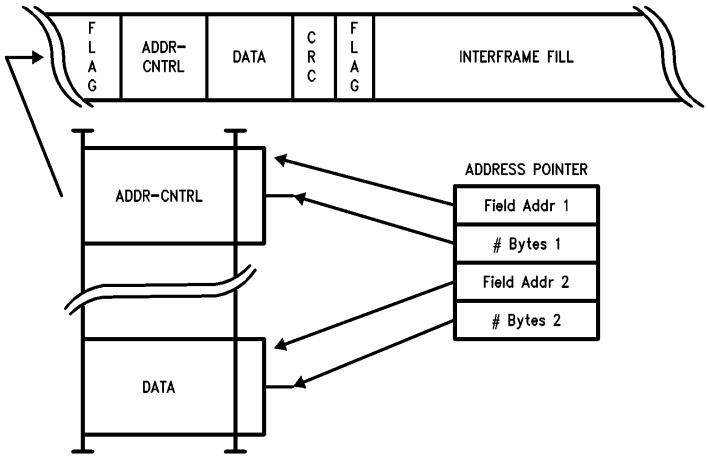
The DMA controller is only capable of taking control of the processor bus when the CPU has finished executing the current instruction. When the HPC16400 executes long instructions, such as the Multiply or Divide instructions, and the HDLC channels are being used at very high data rates (in excess of 2.2 Mbps with a 20 MHz HPC), it may be possible that the DMA cannot gain control of the processor bus in time to service the HDLC channels. In this situation, the receiver is forced to overwrite the last byte received and a receiver overrun is flagged in the error status register. When this occurs during transmission, the transmitter no longer has any valid information to send and so it transmits an abort character and sets a transmitter underrun bit in the error status register. Programming the HDLC/DMA controllers is relatively straightforward, both for their initialization and interrupt servicing. Because the DMA controllers have two sets of registers, it means that the pointers to the next message to be received or transmitted can be set up while reception or transmission is in progress, thereby maximizing the throughput of the HDLC channels.

## THE SERIAL DECODER—BASIC RATE ISDN AS AN EXAMPLE

As already described, the HDLC channels of the HPC16400 can be used in general purpose communications and networking applications. To enhance their capabilities, and provide on-chip support for ISDN, a serial decoder has been implemented to time division multiplex the two HDLC channels onto common transmit and receive lines.

Each HDLC channel can be enabled and disabled both internally by the serial decoder, and externally by individual receiver and transmitter enable pins. The internal enable signals are generated by the serial decoder according six time division multiplexing (TDM) formats. The framing of these TDM formats, or modes, is synchronized by an externally generated frame sync. pulse which will normally be derived from an external clock signal used to clock the HDLC channels. With these inputs, the serial decoder generates the internal enable signals for the HDLC channels at the correct time within the frame according to mode that has been selected. The serial decoder can also be programmed to generate enable signals for the HDLC channels based on combinations of both the external enable signals and those generated internally by the serial decoder, thereby giving the designer a wide choice of possibilities.

As an example of the use of the serial decoder, we shall look at Basic Rate ISDN. Basic Rate ISDN specifies that a terminal equipment, such as a telephone or computer, should have two general purpose B channels (Bearer channels) for voice data or perhaps computer packet switched data, and a D channel which is used specifically for control of the ISDN network, such as setting up a call to another user. These 2B+D channels are time division multiplexed within a 125 $\mu$s frame on a bus which interconnects functional blocks within a piece of equipment. The time slot for each B channel is the transmission time for 8 bits at a data rate of 64 kbps, and the D channel time slot is 2 bits at 16 kbps.



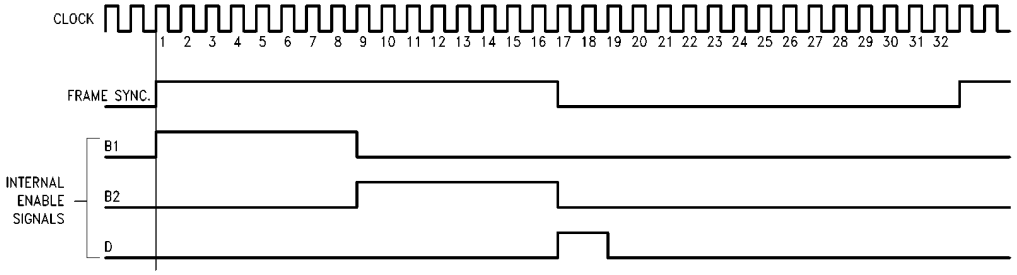

TL/DD/10361–8

**FIGURE 7. Split Frame Operation for HDLC/DMA Transmitter**

The overall scheme is shown in *Figure 8*. Now the HPC16400, having two HDLC channels, could be set up so that one HDLC channel is a B channel, and the other HDLC channel is the D channel. The serial decoder therefore has to be programmed so that its mode corresponds to the format shown in *Figure 7*, and that the enable signals are chosen internally such that the D time slot is assigned to one of the HDLC channels, and the correct B channel is assigned to the other HDLC channel. The remaining B channel could be occupied by any other device capable of generating a 64 kbps data stream within its time slot, such as a voice COMBO. The frame sync. signal and the HDLC clock will be generated externally to the HPC16400, typically by the ISDN line interface circuit as described in the next section.

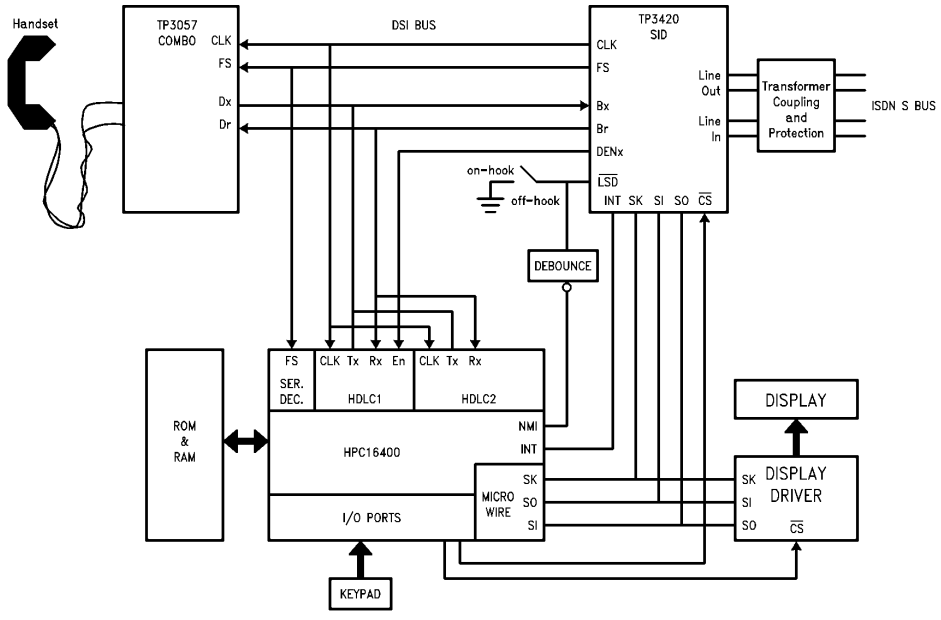
**AN ISDN TELEPHONE**

*Figure 9* shows the block diagram of an ISDN telephone. The three main components of the system are the HPC16400 microcontroller, the TP3420 ''S'' Interface Device (SID) which is the line interface to the ISDN subscriber (S) link, and the TP3057 COMBO which provides the interface to the system for a handset. The inter-chip data bus, whose timing format was used as an example in the previous section, is called the Digital System Interface (DSI) bus, and combines the B and the D channels into common transmit and receive lines. Hence, the HDLC Tx outputs are tied together with the Dx output of the COMBO and are input to the SID DSI input pin Bx, and the HDLC Rx pins are combined with the Dr input from the COMBO and are driven by the SID DSI output pin Br. The SID, when configured in master mode, generates the frame sync. and clock signals which are derived from the received signal on the S bus. These signals are both connected to the HPC16400 and the COMBO so that the correct multiplexing format for the DSI bus as shown in *Figure 9* can be achieved. An additional output from the SID, DENx, indicates the presence of D channel bits on the DSI bus, and is used to enable the HDLC channel of the HPC16400 which has been assigned to handle the D channel communications, in this case HDLC channel 1.




TL/DD/10361–9

**FIGURE 8. The Serial Decoder Format for ISDN**




TL/DD/10361–10

**FIGURE 9. Block Diagram of an ISDN Telephone**

The SID is a programmable device with various modes and functions that conform to the CCITT I.430 specification for the physical layer of ISDN. Programming of the SID is achieved with the MICROWIRE/PLUS interface which is also used to drive the display for the telephone using a COP472-3 liquid crystal display controller. Selection of either the SID or the display driver is achieved with port lines from one of the general purpose I/O ports on the HPC16400 so that the chip selects for each device are software driven.

The Halt power saving mode can be used whenever the telephone is not active. This is indicated by the on/off hook signal from the handset which is interfaced to the NMI input of the HPC. When a telephone conversation is finished and the handset placed on-hook, the HPC can be put into Halt mode by software. When the handset is subsequently picked up for another call, and so goes off-hook, it will generate an NMI which will wakeup the HPC.

**THE ISDN SOFTWARE**

The control of end-to-end communications in a telephone system can be a complex procedure. Many things have to be taken into consideration, such as procedures for establishing a call, dial-plans, disconnecting calls, and so on. All of these procedures amount to the protocols which are part of ISDN. In particular, the control protocols for ISDN are those which are used on the D channel to establish and disconnect physical links between two (or more) users of the telephone network. *Figure 10* shows the three protocol layers of ISDN according to the ISO seven layer reference model for Open Systems Interconnection.

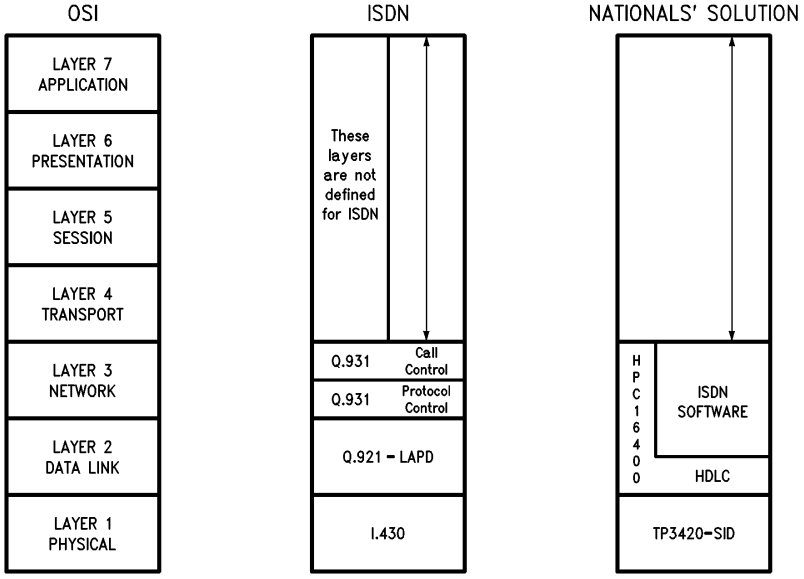
At the physical layer, the CCITT standard I.430 is used to specify the requirements of the ISDN S-Bus interface device. The TP3420 SID conforms to this specification, and in fact exceeds it in some aspects such as its ability to drive longer cable lengths. (The DSI bus is not part of this standard as it refers to the equipment side of the network.)

The data link layer protocol is responsible for the safe delivery of frames across the network. Here, ISDN uses the CCITT standard Q.921 which is more commonly known as LAPD, or ''Link Access Protocol on the D Channel''. LAPD defines the ''HDLC'' frame format and a set of procedures to control the flow of information on the network, and recovery from errors. It is similar to the LAPB link access protocol used in X.25 and true HDLC networks, but defines an expanded set of procedures to cope with communications on a telephone network instead of a typical computer network.

Finally, in Layer 3, the CCITT Q.931 standard specifies a series of procedures for establishing, maintaining, and disconnecting calls between users on the network. Part of these services are application dependent, so in order to make the ISDN standard generic as possible, the Layer 3 is split into two parts. The generic part of Layer 3 executes the ''protocol control procedures'' and the application dependent part performs the ''Call Control Procedures''.

*Figure 10* also shows how the ISDN protocols are mapped onto the hardware components. The SID is the Layer 1 device and the HPC16400 provides hardware support, by means of its HDLC channels, for the Layer 2 protocol. The clear boundary between the Layer 1 and Layer 2 devices results in a well structured system architecture, with the DSI bus creating the physical interface between these two layers. The remaining parts of Q.921 and Q.931 are implemented as a software package which includes drivers for the SID and HDLC/DMA channels, and tools which aid the debugging of application tasks that interface to the software at the Layer 3 call control level.

Within the software, the individual layers and drivers are implemented as tasks which run under a multi-tasking executive. The operation of the executive has been optimized to work with layered tasks, and includes features such as a mail manager, timer manager, and memory manager. The entire software package is written in ''C'' so that application tasks can be developed, run with the layer software (excluding the drivers), and debugged on a PC before being ported to the target hardware.




TL/DD/10361–11

**FIGURE 10. Nationals' Solution to ISDN**

## CONCLUSIONS

The HPC16400 is a versatile high performance 16-bit CMOS microcontroller for embedded communications applications. Its fast CPU together with dual HDLC channels provides an ideal platform for implementing proprietary or standard communication protocols that use the HDLC framing structure.

## REFERENCES

1. *"Expanding the HPC Address Space",* National Semiconductor Application Note 497.
2. *"Intuitive ISDN—An ISDN Tutorial",* National Semiconductor Application Note 492.





**National Semiconductor Corporation**
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

**National Semiconductor GmbH**
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

**National Semiconductor Japan Ltd.**
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

**National Semiconductor Hong Kong Ltd.**
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

**National Semiconductores Do Brazil Ltda.**
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

**National Semiconductor (Australia) Pty, Ltd.**
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998