AP-283

### APPLICATION NOTE

# Flexibility in Frame Size with the 8044

PARVIZ KHODADADI APPLICATIONS ENGINEER

November 1990

Order Number: 292019-001

Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

\*Third-party brands and names are the property of their respective owners.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation P.O. Box 7641 Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

COPYRIGHT © INTEL CORPORATION, 1996

# FLEXIBILITY IN FRAME<br/>SIZE WITH THE 8044CONTENTS1.0 INTRODUCTI

<b>CONTENTS</b> PAGE
1.0 INTRODUCTION         1           1.1 Normal Operation         1           1.2 Expanded Operation         2
2.0 THE SERIAL INTERFACE UNIT22.1 Hardware Description22.2 Reception of Frames32.3 Transmission of Frames3
3.0 TRANSMIT AND RECEIVE STATES
3.1 Receive State Sequence43.2 Transmit State Sequence4
4.0 TRANSMISSION/RECEPTION OF LONG FRAMES (Expanded Operation)       7         4.1 Description       7         4.2 SIU Registers       7
<ul><li>4.3 Other Possibilities</li></ul>
5.0 MODES OF OPERATION         9           5.1 Flexible Mode         9           5.2 Auto Mode         9
6.0 APPLICATION EXAMPLES96.1 Point-To-Point Application Example96.1.1 Polling Sequence106.1.2 Hardware106.1.3 Primary Station Software106.1.4 Secondary Station Software14

#### CONTENTS

6.2 Multidrop Application Example 17
6.2.1 Polling Sequence 17
6.2.2 Hardware 17
6.2.3 Primary Station Software 19
6.2.4 Secondary Station Software 20
6.2.5 Receive Interrupt Routine 21
6.2.6 Transmit Subroutine

PAGE

### 

### intel

#### **1.0 INTRODUCTION**

The 8044 is a serial communication microcontroller known as the RUPI (Remote Universal Peripheral Interface). It merges the popular 8051 8-bit microcontroller with an intelligent, high performance HDLC/SDLC serial communication controller called the Serial Interface Unit (SIU). The chip provides all features of the microcontroller and supports the Synchronous Data Link Control (SDLC) communications protocol.

There are two methods of operation relating to frame size:

1) Normal operation (limited frame size)

2) Expanded operation (unlimited frame size)

In Normal operation the internal 192 byte RAM is used as the receive and transmit buffer. In this operation, the chip supports data rates up to 2.4 Mbps externally clocked and 375 Kbps self-clocked. For frame sizes greater than 192 bytes, Expanded operation is required. In Expanded operation the external RAM, in conjunction with the internal RAM, is used as the transmit and receive buffer. In this operation, the chip supports data rates up to 500 Kbps externally clocked and 375 Kbps self-clocked. In both cases, the SIU handles many of the data link functions in hardware, and the chip can be configured in either Auto or Flexible mode.

The discussion that follows describes the operation of the chip and the behavior of the serial interface unit. Both Normal and Expanded operations will be further explained with extra emphasis on Expanded operation and its supporting software. Two examples of SDLC communication systems will also be covered, where the chip is used in Expanded operation. The discussion assumes that the reader is familiar with the 8044 data sheet and the SDLC communications protocol.

#### 1.1 Normal Operation

In Normal operation the on-chip CPU and the SIU operate in parallel. The SIU handles the serial communication task while the CPU processes the contents of the on-chip transmit and receiver buffer, services interrupt routines, or performs the local real time processing tasks.

The 192 bytes of on-chip RAM serves as the interface buffer between the CPU and the SIU, used by both as a receive and transmit buffer. Some of the internal RAM space is used as general purpose registers (e.g. RO-R7). The remaining bytes may be divided into at least two sections: one section for the transmit buffer and the other section for the receive buffer. In some applications, the 192 byte internal RAM size imposes a limitation on the size of the information field of each frame and, consequently, achieves less than optimal information throughput.

Figure 1 illustrates the flow of data when internal RAM is used as the receive and transmit buffer. The on-chip CPU allocates a receive buffer in the internal RAM and enables the SIU. A receiving SDLC frame is processed by the SIU and the information bytes of the frame, if any, are stored in the internal RAM. Then, the SIU informs the CPU of the received bytes (Serial Channel interrupt). For transmission, the CPU loads the transmitting bytes into the internal RAM and enables the SIU. The SIU transmits the information bytes in SDLC format.

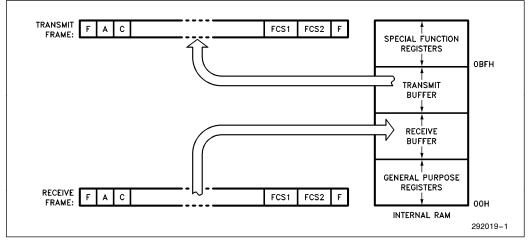


Figure 1. Transmission/Reception Data Flow Using Internal RAM



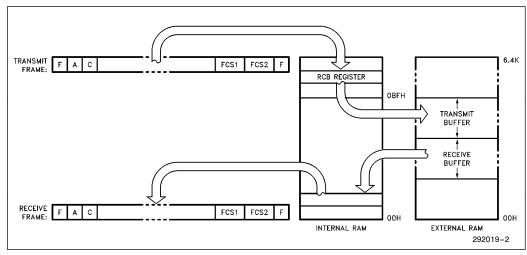


Figure 2. Transmission/Reception Data Flow Using External RAM

#### 1.2 Expanded Operation

In Expanded operation the on-chip CPU monitors the state of the SIU, and moves data from/to external buffer to/from the internal RAM and registers while reception/transmission is taking place. If the CPU must service an interrupt during transmission or reception of a frame or transmit from internal RAM, the chip can shift to Normal operation.

There is a special function register called SIUST, the contents of which dictate the operation of the SIU. Also, at data rates lower than 2.4 Mbps, one section of the SIU, in fixed intervals during transmission and reception, is in the "standby" mode and performs no function. The above two characteristics make it possible to program the CPU to move data to/from external RAM and to force the SIU to repeat or skip some desired hardware tasks while transmission or reception is taking place. With these modifications, external RAM can be utilized as a transmit and receive buffer instead of the internal RAM.

Figure 2 graphically shows the flow of data when external RAM is used. For reception, the receiving bytes are loaded into the Receive Control Byte (RCB) register. Then, the data in RCB is moved to external RAM and the SIU is forced to load the next byte into the RCB register - The chip believes it is receiving a control byte continuously. For transmission, Information bytes (Ibytes) are loaded into a location in the internal RAM and the chip is forced to transmit the contents of this location repeatedly.

Discussion of expanded operation is continued in sections 4 and 5. First, however, sections 2 and 3 describe features of the 8044 which are necessary to further explain expanded operation.

#### 2.0 THE SERIAL INTERFACE UNIT

#### 2.1 Hardware Description

The Serial Interface Unit (SIU) of the RUPI, shown in Figure 3, is divided functionally into a Bit Processor (BIP) and a Byte Processor (BYP), each sharing some common timing and control logic. The bit processor is the interface between the SIU bus and the serial port pins. It performs all functions necessary to transmit/receive a byte of data to/from the serial data line (shifting, NRZI coding, zero insertion/deletion, etc.). The byte processor manipulates bytes of data to perform message formatting, transmitting, and receiving functions. For example, moving bytes from/to the special function registers to/from the bit processor.

The byte processor is controlled by a Finite-State Machine (FSM). For every receiving/transmitting byte, the byte processor executes one state. It then jumps to the next state or repeats the same state. These states will be explained in section 3. The status of the FSM is kept in an 8-bit register called SIUST (SIU State Counter). This register is used to manipulate the behavior of the byte processor.

As the name implies, the bit processor processes data one bit at a time. The speed of the bit processor is a function of the serial channel data rate. When one byte of data is processed by the bit processor, a byte bounda-

ry is reached. Each time a byte boundary is detected in the serial data stream, a burst of clock cycles (16 CPU states) is generated for the byte processor to execute one state of the state machine. When all the procedures in the state are executed, a wait signal is asserted to terminate the burst, and the byte processor waits for the next byte boundary (standby mode). The lower the data rate, the longer the byte processor will stay in the standby mode.

#### 2.2 Reception of Frames

Incoming data is NRZI decoded by the on-chip decoder. It is then passed through the zero insertion/deletion (ZID) circuitry. The ZID not only performs zero insertion/deletion, but also detects flags and Go Aheads (GA) in the data stream. The data bits are then loaded into the shift register (SR) which performs serial to parallel conversion. When 8 bits of data are collected in the shift register, the bit processor triggers the byte processor to process the byte, and it proceeds to load the next

#### FLEXIBILITY IN FRAME SIZE WITH THE 8044

block of data into the shift register. The serial data is also shifted, through SR, to a 16-bit register called "FCS GEN/CHK" for CRC checking. The byte processor takes the received address and control bytes from the SR shift register and moves them to the appropriate registers. If the contents of the shift register is expected to be an information byte, the byte processor moves them through a 3-byte FIFO to the internal RAM at a starting location addressed by the contents of the Receive Buffer Start (RBS) register.

#### 2.3 Transmission of Frames

In the transmit mode, the byte processor relinquishes a byte to the bit processor by moving it to a register called RB (RAM buffer). The bit processor converts the data to serial form through the shift register, performs zero bit insertion, NRZI encoding, and sends the data to the serial port for transmission. Finally, the contents of the FCS GEN/CHK and the closing flag are routed to the serial port for transmission.

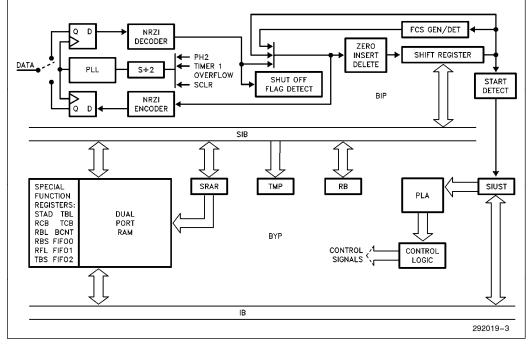


Figure 3. SIU Block Diagram

### 3.0 TRANSMIT AND RECEIVE STATES

The simplified receive and transmit state diagrams are shown in Figures 4 and 5, respectively. The numbers on the left of each state represent the contents of the SIUST register when the byte processor is in the standby mode, and the instructions on the right of each state represent the "state procedures" of that state. When the byte processor executes these procedures the least three significant bits of the SIUST register are being incremented while the other bits remain unchanged. The byte processor will jump from one state to another without going into the standby mode when a conditional jump procedure executed by the byte processor is true.

#### 3.1 Receive State Sequence

When an opening flag (7EH) is detected by the bit processor, the byte processor is triggered to execute the procedures of the FLAG state. In the FLAG state, the byte processor loads the contents of the RBS register into the Special RAM (SRAR) register. SRAR is the pointer to the internal RAM. The byte processor decrements the contents of the Receive Buffer Length (RBL) register and loads them into the DMA Count (DCNT) register. The FCS GEN/CHK circuit is turned on to monitor the serial data stream for Frame Check Sequence functions as per SDLC specifications.

Assuming there is an address field in the frame, contents of the SIUST register will then be changed to 08H, causing the byte processor to jump to the AD-DRESS state and wait (standby) for the next byte boundary. As soon as the bit processor moves the address byte into the SR shift register, a byte boundary is achieved and the byte processor is triggered to execute the procedures in the ADDRESS state.

In the ADDRESS state the received station address is compared to the contents of the STAD register. If there is no match, or the address is not the broadcast address (FFH), reception will be aborted (SIUST = 01H). Otherwise, the byte processor jumps to the CONTROL state (SIUST = 10H) and goes into standby mode.

The byte processor jumps to the CONTROL state if there exists a control field in the receiving frame. In this state the control byte is moved to the RCB register by the byte processor. Note that the only action taken in this state is that a received byte, processed by the bit processor, is moved to RCB. There is no other hardware task performed, and DCNT and SRAR are not affected in this state.

The next two states, PUSH-1 and PUSH-2, will be executed if Frame check sequence (NFCS = 0) option is selected. In these two states the first and second bytes



of the information field are pushed into the 3-byte FIFO (FIFO0, FIFO1, FIFO2) and the Receive Field Length register (RFL) is set to zero. The 3-byte FIFO is used as a pipeline to move received bytes into the internal RAM. The FIFO prevents transfer of CRC bytes and the closing flag to the receive buffer (i.e., when the ending flag is received, the contents of FIFO are FLAG, FCS1, and FCS0.) The three byte FIFO is collapsed to one byte in No FCS mode.

In the DMA-LOOP state the byte processor pushes a byte from SR to FIFO0, moves the contents of FIFO2 to the internal RAM addressed by the contents of SRAR, increments the SRAR and RFL registers, and decrements the DCNT register. If more information bytes are expected, the byte processor repeats this state on the next byte boundaries until DMA Buffer End occurs. The DMA Buffer End occurs if SRAR reaches OBFH (192 decimal), DCNT reaches zero, or the RBP bit of the STS register is set.

The BOV-LOOP state, the last state, is executed if there is a buffer overrun. Buffer overrun occurs when the number of information bytes received is larger than the length of the receive buffer (RFL > RBL). This state is executed until the closing flag is received.

At the end of reception, if the FCS option is used, the closing flag and the FCS bytes will remain in the 3-byte FIFO. The contents of the RCB register are used to update the NSNR (Receive/Send Count) register. The SIU updates the STS register and sets the serial interrupt.

#### 3.2 Transmit State Sequence

Setting the RTS bit puts the SIU in the transmit mode. When the CTS pin goes active, the byte processor goes into START-XMIT state. In this state the opening flag is moved into the RAM Buffer (RB) register. The byte processor jumps to the next state and goes into the standby mode.

If the Pre-Frame Sync (PFS) option is selected, the PFS1 and PFS2 states will be executed to transmit the two Pre-Frame Sync bytes (00H or 55H). In these two states the contents of the Pre-Frame Sync generator are sent to the serial port while the Zero Insertion Circuit (ZID) is turned off. ZID is turned back on automatically on the next byte boundary.

If the PFS option is not chosen, the byte processor jumps to the FLAG state. In this state, the byte processor moves the contents of TBS into the SRAR register, decrements TBL and moves the contents into the DCNT register. The byte processor turns off the ZID and turns on FCS GEN/CHK. The contents of FCS GEN/CHK are not transmitted unless the NFCS bit is

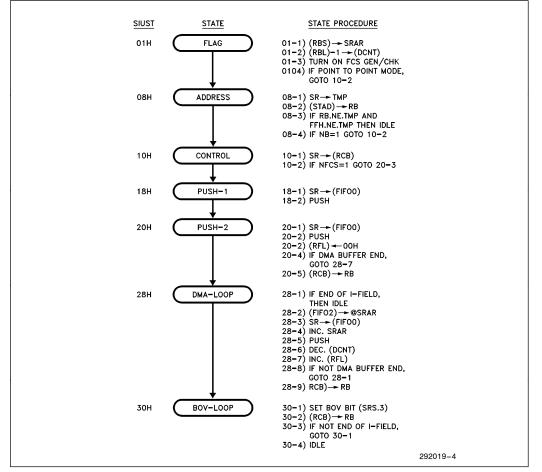


Figure 4. Receive State Diagram

set. If a frame with the address field is chosen, it moves the contents of the STAD register into the RB register for transmission. At the same time, the opening flag is being transmitted by the bit processor.

In the ADDRESS (SIUST = A0H) and CONTROL (SIUST = A8H) states, TCB and the first information byte are loaded into the RB register for transmission, respectively. Note that in the CONTROL state, none of the registers (e.g. DCNT, SRAR) are incremented, and ZID and FCS GEN/CHK are not turned on or off.

The procedures in the DMA-LOOP state are similar to the procedures of the DMA-LOOP in the receive state diagram. The SRAR register pointer to the internal RAM is incremented, and the DCNT register is decremented. The contents of DCNT reach zero when all the information bytes from the transmit buffer are transmitted. A byte from RAM is moved to the RB register for transmission. This state is executed on the following byte boundaries until all the information bytes are transmitted.

The FCS1 and the FCS2 states are executed to transmit the Frame Check Sequence bytes generated by the FCS generator, and the END-FLAG state is executed to transmit the closing flag.

The XMIT-ACTION and the ABORT-ACTION states are executed by the byte processor to synchronize the SIU with the CPU clock. The XMIT-ACTION or the ABORT-ACTION state is repeated until the byte processor status is updated. At the end, the STS and the TMOD registers are updated.

The two ABORT-SEQUENCE states (SIUST = E0H and SIUST = E8H) are executed only if transmission is aborted by the CPU (RTS or TBF bit of the STS register is cleared) or by the serial data link (CTS signal goes inactive or shut-off occurs in loop mode.)

### intel

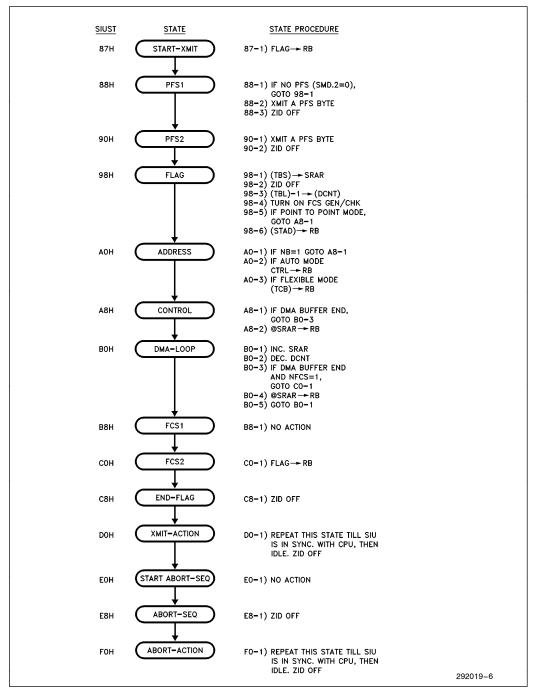


Figure 5. Transmit State Diagram

### intel

#### 4.0 TRANSMISSION/RECEPTION OF LONG FRAMES (EXPANDED OPERATION)

In this application note, a frame whose information field is more than 192 bytes (size of on-chip RAM) is referred to as a long frame. The 8044 can access up to 64000 bytes of external RAM. Therefore, a long frame can have up to 64000 information bytes.

#### 4.1 Description

During transmission or reception of a frame, while the bit processor is processing a byte, the byte processor, after 16 CPU states, is in the standby mode, and the internal registers and the internal bus are not used. The period between each byte boundary, when the byte processor is in the standby mode, can be used to move data from external RAM to one of the byte processor registers for transmission and vice versa for reception. The contents of the SIUST register, which dictate the state of the byte processor, can be monitored to recognize the beginning of each SDLC field and the consecutive byte boundaries.

By writing into the SIUST register, the byte processor can be forced to repeat or skip a specific state. As an example, the SIU can be forced to repeatedly put the received bytes into the RCB register. This is accomplished by writing E7H into the SIUST register when the byte processor goes into the standby mode. The byte processor, therefore, executes the CONTROL state at the next byte boundary.

For transmission, the byte processor is put in the transmit mode. When transmission of a frame is initiated, the user program calls a subroutine in which the state of the byte processor is monitored by checking the contents of the SIUST register. When the byte processor reaches a desired state and goes into standby, the CPU loads the first byte of the internal RAM buffer with data and moves the byte processor to the CONTROL state. The routine is repeated for every byte. At the end, the program returns from the subroutine, and the SIU finishes its task (see application examples).

For reception, a software routine is executed to move data to external RAM and to force the SIU to repeat the CONTROL state. The CONTROL state is repeated because, as shown in the receive state diagram, the only action taken by the byte processor, in the CONTROL state, is to move the contents of SR to the RCB register. None of the registers (e.g. SRAR and DCNT) are incremented. A similar comment justifies the use of the CONTROL state for transmission. In the transmit CONTROL state, contents of a location in the on-chip RAM addressed by TBS is moved to RB for transmission.

#### 4.2 SIU Registers

To write into the SIUST register, the data must be complemented. For example, if you intend to write 18H into the SIUST register, you should write E7H to the register. The data read from SIUST is, however, true data (i.e. 18H).

Read and write accesses to the SIUST, STAD, DCNT, RCB, RBL, RFL, TCB, TBL, TBS, and the 3-byte FIFO registers are done on even and odd phases, respectively. Therefore, there is no bus contention when the CPU is monitoring the registers (e.g. SIUST), and SIU is simultaneously writing into them.

There is no need to change or reset the contents of any SIU register while transmitting or receiving long frames, unless the byte processor is forced to repeat a state in which the contents of these registers are modified. Note that the SRAR register can not be accessed by the CPU; therefore, avoid repeating the DMA-LOOP states. If SRAR increments to 192, the SIU will be interrupted and communication will be aborted.

#### 4.3 Other Possibilities

The internal RAM, in conjunction with an external buffer (RAM or FIFOs), can be used as a transmit and receive buffer. In other words, Expanded and Normal operation can be used together. For example, if a frame with 300 Information bytes is received and only 255 of them are moved to an external buffer, the remaining bytes (45 bytes) will be loaded into the internal RAM by the SIU (assuming RBL is set to 45 or more). The contents of RFL indicate the number of bytes stored in the internal RAM. For transmission, the contents of the external buffer can be transmitted followed by the contents of the internal buffer.

If the internal RAM is not used, contents of the RBL register can be 0 and contents of the TBL register must be set to 1. The contents of the TBS register can be any location in the internal RAM.

The transmission and reception procedures for long frames with no FCS are similar to those with FCS. The exception is the contents of the SIUST register should be compared with different values since the two FCS states of the transmit and receive flow charts are skipped by the byte processor.

If a frame format with no control byte is chosen, a location in the RAM addressed by TBS should be used for transmission as with control byte format. The FIFO can be used for reception. The STAD register can be used for transmission if no zero insertion is required.



If the RUPI is used in Auto mode (see Section 5), it will still respond to RR, RNR, REJ, and Unnumbered Poll (UP) SDLC commands with RR or RNR automatically, without using any transmit routine. For example, if the on-chip CPU is busy performing some real time operations, the SIU can transmit an information frame from the internal buffer or transmit a supervisory frame without the help of CPU (Normal operation).

Maximum data rate using this feature is limited primarily by the number of instructions needed to be executed during the standby mode.

Transmission or reception of a frame can be timed out so that the CPU will not hang up in the transmit or receive procedures if a frame is aborted. Or, if the data rate allows enough time (standby time is long enough), the CPU can monitor the SIUST register for idle mode (SIUST = 01H).

It is also possible to transmit multiple opening or closing flags by forcing the byte processor to repeat the END-FLAG state.

#### 4.4 Maximum Data Rate in Expanded Operation

Assuming there is no zero-insertion/deletion, the bit processor requires eight serial clock periods to process one block of data. The byte processor, running on the CPU clock, processes one byte of data in 16 CPU states (one state of the state diagrams). Each CPU state is two oscillator periods. At an oscillator frequency of 12 MHz, the CPU clock is 6 MHz, and 16 CPU states is 2.7  $\mu$ s. At a 3 Mbit rate with no zero-insertion/deletion, there is exactly enough time to execute one state per byte (16 states at 6 MHz = 8 bits at 3M baud). In other words, the standby time is zero.

Figure 6 demonstrates portions of the timing relationship between the byte processor and the bit processor. In each state, the actions taken by the processors, plus the contents of the SIUST register, are shown. When the byte processor is running, the contents of SIUST are unknown. However, when it is in the standby mode, its contents are determinable.

The maximum data rate for transmitting and receiving long frames depends on the number of instructions needed to be executed during standby, and is propor-

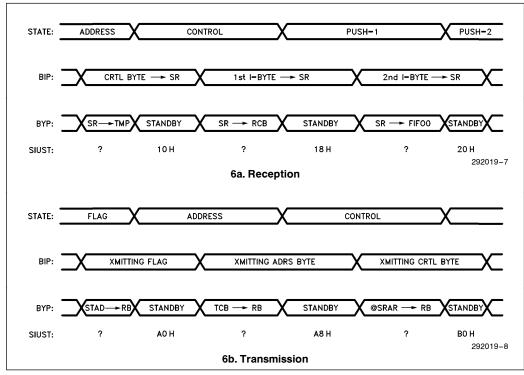


Figure 6. Portions of the BIP/BYP Timing Relationship

tional to the oscillator frequency. The time the byte processor is in the standby mode, waiting for the bit processor to deliver a processed byte, is at least equal to eight serial clock periods minus 16 CPU states. If an inserted zero is in the block of data, the bit processor will process the byte in nine serial clock periods.

The equation for theoretical maximum data rate is given as:

 $\begin{array}{ll} (\mbox{2TCLCL}) \times (\mbox{16 states}) + (\mbox{\# of instruction cycles}) \times \\ (\mbox{12TCLCL}) = (\mbox{8TDCY}) & \mbox{Equation (1)} \end{array}$ 

Where: TCLCL is the oscillator period. TDCY is the serial clock period.

At an oscillator frequency of 12 MHz and baud rate of 375 Kbps, about 18 instruction cycles can be executed when the byte processor is in the standby mode. At a 9600 baud rate, there is time to execute about 830 instruction cycles—plenty of time to service a long interrupt routine or perform bit-manipulation or arithmetic operations on the data while transmission or reception is taking place.

#### 5.0 MODES OF OPERATION

The 8044 has two modes: Flexible mode and Auto mode. In Auto mode, the chip responds to many SDLC commands and keeps track of frame sequence numbering automatically without on-chip CPU intervention. In Flexible mode, communication tasks are under control of the on-chip CPU.

#### 5.1 Flexible Mode

For transmission, the CPU allocates space for transmit buffer by storing values for the starting location and size of the transmit buffer in the TBS and the TBL registers. It loads the buffer with data, sets the TBF and the RTS bits in the STS register, and proceeds to perform other tasks. The SIU activates the RTS line. When the CTS signal goes active, the SIU transmits the frame. At the end of transmission, the SIU clears the RTS bit and interrupts the CPU (SI set).

For reception, the CPU allocates space for receive buffer by loading the beginning address and length of the receive buffer into the RBS and RBL registers, sets the RBE bit, and proceeds to perform other tasks. The SIU, upon detection of an opening flag, checks the next received byte. If it matches the station address, it will load the received control byte into RCB, and received information bytes into the receive buffer. At the end of reception, if the Frame Check Sequence (FCS) is correct, the SIU clears RBE and interrupts the CPU.

#### 5.2 Auto Mode

In the Auto mode, the 8044 can only be a secondary station operating in the SDLC "Normal Response Mode". The 8044 in Auto mode does not transmit messages unless it is polled by the primary.

For transmission of an information frame, the CPU allocates space for the transmit buffer, loads the buffer with data, and sets the TBF bit. The SIU will transmit the frame when it receives a valid poll-frame. A frame whose poll bit of the control byte is set, is a poll-frame. The poll bit causes the RTS bit to be set. If TBF were not set, the SIU would respond with Receive Not Ready (RNR) SDLC command if RBP = 1, or with Receive Ready (RR) SDLC command if RBP = 0. After transmission RTS is cleared, and the CPU is not interrupted.

For reception, the procedure is the same as that of Flexible mode. In addition, the SIU sets the RTS bit if the received frame is a poll-frame (causing an automatic response) and increments the NS and NR counts accordingly.

#### 6.0 APPLICATION EXAMPLES

Two application examples are given to provide additional details about the procedures used to transmit and receive long frames. In the first application example, procedures to construct receive and transmit software routines for the point-to-point frame format are described. The point-to-point frame has the information field and the FCS field enclosed between two flags (see Figure 7). In the second example software code is generated for reception and transmission of the standard SDLC frame. The SDLC frame has the pattern: flag, address, control, information, FCS, flag.

The first example focuses on the construction of transmit and receive code which allow the chip to transmit and receive long frames. The second example shows how to make more use of the 8044 features, such as the on-chip phase locked loop for clock recovery and automatic responses in the Auto mode to demonstrate the capability of the 8044 to achieve high throughput when Expanded operation is used.

#### 6.1 Point-to-Point Application Example

A point-to-point communication system was developed to receive and transmit long frames. The system consists of one primary and one secondary station. Although multiple secondary stations can be used in this

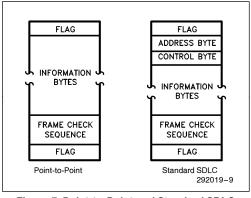


Figure 7. Point-to-Point and Standard SDLC Frame Formats

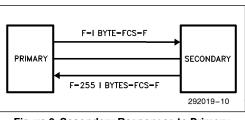
system, one secondary is chosen to simplify the primary station's software and focus on the long frame software code. Both the primary and the secondary stations are in Flexible mode and the external clock option is used for the serial channel. The maximum data rate is 500 Kbps. The FCS bytes are generated and checked automatically by both stations.

#### 6.1.1 POLLING SEQUENCE

The polling sequence, shown in Figure 8, takes place continuously between the primary and the secondary stations. The primary transmits a frame with one information byte to the secondary. The information byte is used by the secondary as an address byte. The secondary checks the received byte, and if the address matches, the secondary responds with a long frame. In this example, the information field of the frame is chosen to be 255 bytes long. Since there is only one secondary station, the address always matches. Upon successful reception of the long frame, the primary transmits another frame to the secondary station.

#### 6.1.2 HARDWARE

The schematic of the secondary station is given in Figure 9. The circuit of the primary station is identical to the secondary station with the exception of pin 11



INTo

Figure 8. Secondary Responses to Primary Station Commands

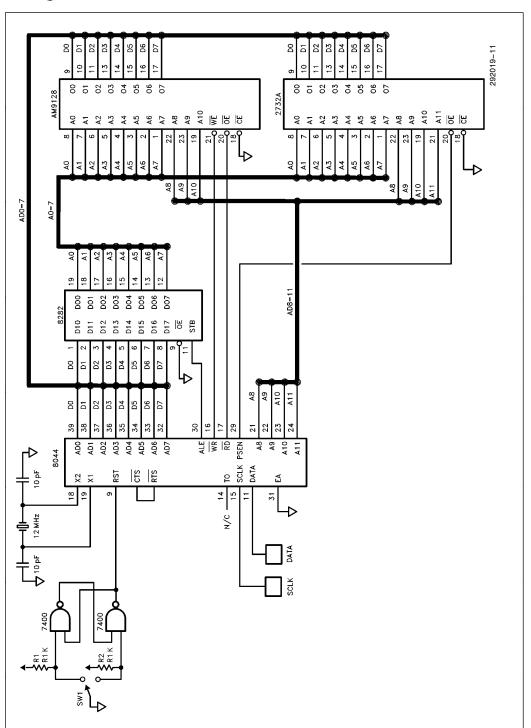
(DATA) being connected to pin 14 (T0). In the primary station, the 8044 is interrupted when activity is detected on the communication line by the on-chip timer (in counter mode). This is explained more later. The serial clock to both stations is supplied by a pulse generator. The output of the pulse generator (not shown in the diagram) is connected to pin 15 of the 8044s. Since the two stations are located near each other (less than 4 feet), line drivers are not used.

The central processor of each station is the 8044. The data link program is stored in a 2Kx8 EPROM (2732A), and a 2Kx8 static RAM (AM9128) is used as the external transmit and receive buffer. The RTS pin is connected to the CTS pin. For simplicity, the stations are assumed to be in the SDLC Normal Respond Mode after Hardware reset.

#### 6.1.3 PRIMARY STATION SOFTWARE

The assembly code for the primary station software is listed in Appendix A. The primary software consists of the main routine, the SIU interrupt routine, and the receive interrupt routine. The receive interrupt routine is executed when a long frame is being received.

In the flow charts that follow, all actions taken by the SIU appear in squares, and actions taken by the on-chip CPU appear in spheres.



#### Figure 9. Secondary Station Hardware

#### FLEXIBILITY IN FRAME SIZE WITH THE 8044



#### Main Routine

First, the chip is initialized (see Figure 10). It is put in Flexible mode, externally clocked, and "Flag-Information Field-FCS-Flag" frame format. Pre-Frame Sync option (PFS = 1) and automatic Frame Check Sequence generation/detection (NFCS = 0) are selected. The on-chip transmit buffer starts at location 20H and the transmit buffer length is set to 1. This one byte buffer contains the address of the secondary station. There is no on-chip receive buffer since the long frame being received is moved to the external buffer. The RTS, TBF, and RBE bits are set simultaneously. Setting the RTS and TBF bits causes the SIU to transmit the contents of the transmit buffer.

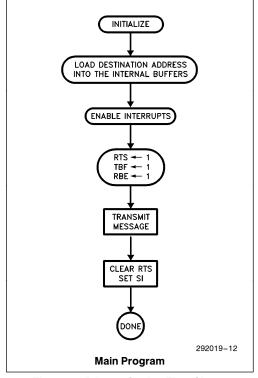


Figure 10. Primary Station Flow Charts

#### **SIU Interrupt Routine**

After transmission of the frame, the SIU interrupts the on-chip CPU (SI is set). In the SIU interrupt service routine, counter 0 is initialized and turned on (see Figure 11). The user program returns to perform other tasks. After reception of the long frame, the SIU interrupt routine is executed again. This time, RTS, TBF, and RBE are set for another round of information exchange between the two stations.

SIU never interrupts while reception or transmission is taking place. The SIU registers are updated and the SI is set (serial interrupt) after the closing flag has been received or transmitted. An SIU interrupt never occurs if the receive interrupt routine or the transmit subroutine is being executed.

Setting the RBE bit of the STS register puts the RUPI in the receive mode. However, the jump to the receive interrupt routine occurs only when a frame appears on the serial port. Incoming frames can be detected using the Pre-Frame Sync. option and one of the CPU timers in counter mode. The counter external pin (T0) is connected to the data line (pin 11 is tied to pin 14). Setting the PFS (Pre-Frame Sync.) bit will guarantee 16 transitions before the opening flag of a flame.

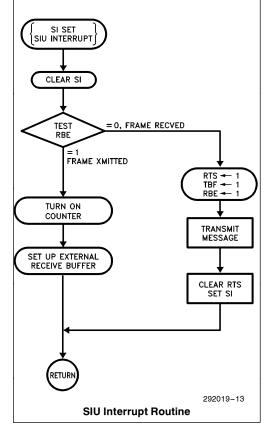


Figure 11. Primary Station Flow Charts

The counter registers are initialized such that the counter interrupt occurs before the opening flag of a frame. When the PFS transitions appear on the data line, the counter overflows and interrupts the CPU. The CPU program jumps to the timer interrupt service routine and executes the receive routine. In the receive routine, the received frame is processed, and the information bytes are moved to the external RAM. Note that the maximum count rate of the 8051 counter is  $\frac{1}{24}$  of the oscillator frequency. At 12 MHz, the data rate is limited to 500 Kbps.

Another method to detect a frame on the data line and cause an interrupt is to use an external "Flag-Detect" circuit to interrupt the CPU. The "Flag Detect" circuit can be an 8-bit shift register plus some TTL chips. If this option is used, the RUPI must operate in externally clocked mode because the clock is needed to shift the incoming data into the shift register. With this option, the maximum data rate is not limited by the maximum count rate of the 8051 counter.

#### **Receive Interrupt Routine**

In Normal operation, the byte processor executes the procedures of the FLAG state, jumps to the CON-TROL state without going into the standby mode, and executes 10–2 procedure of the state (see Figure 4). It then jumps to the PUSH-1 state and goes into the standby mode. At the following byte boundaries, the byte processor executes the PUSH-1, PUSH-2, and DMA-LOOP states, respectively. The receive interrupt routine as shown in the flow chart of Figure 12 and described below forces the byte processor to repeatedly execute the CONTROL state before the PUSH-1 state is executed. The following is the step by step procedure to receive long frames:

 Turn off the CPU counter and save all the important registers. Jump to the receive interrupt routine, execution of the instructions to save registers, and initialization of the receive buffer pointer take place while the Pre-Frame Sync bytes and the opening flag are being received. This is about three data byte periods (48 CPU cycles at 500 Kbps).

#### FLEXIBILITY IN FRAME SIZE WITH THE 8044

- 2) Monitor the SIUST register for standby in the PUSH-1 state (SIUST = 18H). When the SIUST contents are 18H, the byte processor is waiting for the first information byte. The bit processor has already recognized the flag and is processing the first information byte.
- 3) In the standby mode, move the byte processor into the CONTROL state by writing "EFH" (complement of 10H) into the SIUST register. When the next byte boundary occurs, the bit processor has processed and moved a byte of data into the SR register. The byte processor moves the contents of SR into the RCB register, jumps to the PUSH-1 state (SIUST = 18H), and waits.
- 4) Monitor the SIUST register for standby in the PUSH-1 state. When the contents of SIUST becomes 18H, the contents of RCB are the first information byte of the information field.
- 5) While the byte processor is in the standby mode, move the contents of RCB to an external RAM or an I/O port.
- 6) Check for the end of the information field. The end can be detected by knowing the number of bytes transmitted, or by having a unique character at the end of information field. The length of the information field can be loaded into the first byte(s) received. The receive routine can load this byte into the loop counter.
- 7) If the byte received is not the last information byte, move the byte processor back to standby in the CONTROL state and repeat steps 4 through 6. Otherwise, return from the interrupt routine.

Upon returning from the receive interrupt routine, the byte processor automatically executes the PUSH-1, PUSH-2, and DMA-LOOP before it stops. This causes the remaining information bytes (if any) to be stored in the internal RAM at the starting location specified by the contents of RBS register. At the end of the cycle, the closing flag and the CRC bytes are left in the FIFO. The RFL register will be incremented by the number of bytes stored in the internal RAM. Then, the STS and NSNR registers are updated, and an appropriate response is generated by the SIU.

The software to perform the above task is given in Table 1. In this example, the number of instruction cycles executed during standby is 12 cycles.



Receive Codes		Cycles	
	•	•	
	•	•	
	•	•	
REC:	CLR	TRO	
	MOV	A,#18H	
WAIT1:	CJNE	A, SIUST, WAITI	
NEXTI:	MOV	SIUST, #0EFH2	
	MOV	A,#18Hl	
WAIT2:	CJNE	A,SIUST,WAIT22	
	MOV	A,RCB1	
	MOVX	@DPTR,A2	
	INC	DPTR2	
	DJNZ	R5.NEXTI2	
	RETI		
END		12 Cycl	

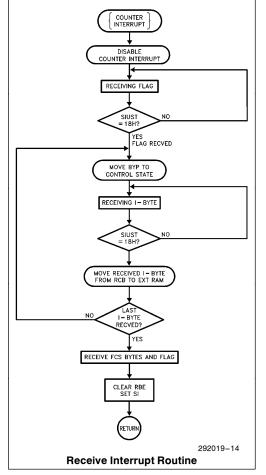


Figure 12. Primary Station Flow Charts

#### **6.1.4 SECONDARY STATION SOFTWARE**

The assembly code for the secondary station software is given in Appendix A. The secondary station contains the transmit subroutine which is called for transmission of long frames.

inte

#### Main Routine

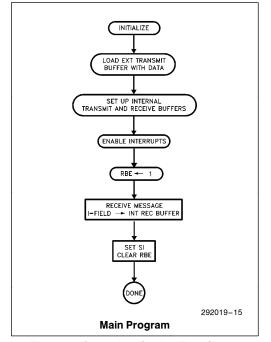
As shown in the secondary station flow chart (Figure 13), the external transmit buffer (external RAM) is loaded with the information data (FFH, FEH, FDH, ...) at starting location 200H. The internal transmit buffer (on chip RAM) starts at location 20H (TBS = 20H), and the transmit buffer length (TBL) is set to 1. The on-chip CPU, in the transmit subroutine, moves the information bytes from the external RAM to this one byte buffer for transmission. The receive buffer starts at location 10H and the receiver buffer length is 1. This buffer is used to buffer the frame transmitted by the primary. The received byte is used as an address byte.

The Secondary is configured like the Primary station. It is put in Flexible mode, externally clocked, Point-topoint frame format. The PFS bit is set to transmit two bytes before the first flag of a frame. The RBE bit is set to put the chip in receive mode. Upon reception of a valid frame, the SIU loads the received information byte into the on-chip receive buffer and interrupts the CPU.

#### **SIU Interrupt Routine**

In the serial interrupt routine, the RBE bit is checked (see Figure 14). Since RBE is clear, a frame has been received. The received Information byte is compared with the contents of the Station Address (STAD) register.

#### FLEXIBILITY IN FRAME SIZE WITH THE 8044



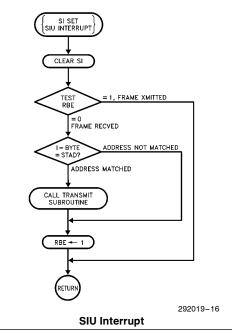


Figure 13. Secondary Station Flow Charts

If they match, the secondary will call the transmit subroutine to transmit the long frame. Upon returning from the transmit subroutine, the RBE bit is set, and program returns from the SIU interrupt. After transmission of the closing flag, SIU interrupt occurs again. In the interrupt routine, the RBE is checked. Since the RBE is set, the program returns from the SIU interrupt routine and waits until another long frame is received.

If the secondary were in Auto mode, the chip must be ready to execute the transmit routine upon reception of a poll-frame; otherwise, the chip automatically transmits the contents of the internal transmit buffer if the TBF bit is set, or transmits a supervisory command (RR or RNR) if TBF is clear.

#### **Transmit Subroutine**

In Normal operation the byte processor executes the START-TRANSMIT state and jumps to the PFS1 state. While the bit processor is transmitting some unwanted bits, the byte processor executes the PFS1 state and jumps to the standby mode in the PFS2 state.

Figure 14. Secondary Station Flow Charts

While the bit processor is transmitting the first Pre-Frame Sync byte, the byte processor executes the PFS2 state and jumps to the standby mode in the FLAG state. The FLAG state is executed when the bit processor begins to transmit the second Pre-Frame Sync byte. When the flag is being transmitted, the byte processor executes the 98-1, 98-2, 98-3, and 98-4 procedures of the FLAG state, and jumps to execute the A8-1 procedure of the CONTROL state. When the opening flag is transmitted, the contents of RB are the first information byte. (See transmit State diagram.)

In the transmit subroutine (see Figure 15), the byte processor is forced to repeat the CONTROL state before the DMA-LOOP state. In the CONTROL state, the contents of a RAM location addressed by the TBS register are moved to the RB register. The following is the step by step procedure to transmit long frames:

- 1) Put the chip in transmit mode by setting the RTS and TBF bits.
- Move an information byte from external RAM to a location in the internal RAM addressed by the contents of TBS.

### intel

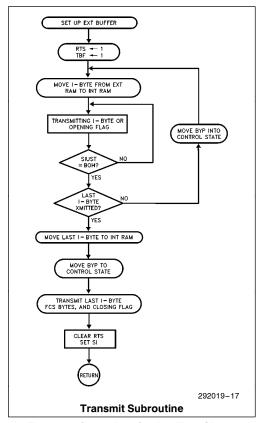


Figure 15. Secondary Station Flow Charts

- 3) Monitor the SIUST register for the standby mode in the DMA-LOOP state (SIUST = B0H). When SIUST is B0H, the opening flag has been transmitted, and the first information byte is being transmitted by the bit processor.
- 4) If there are more information bytes, move the byte processor back to the CONTROL state, and repeat steps 2 through 4. Otherwise, continue.
- 5) Move byte processor to the Standby mode in the CONTROL state (SIUST = A8H) and return from the subroutine.

The byte processor automatically executes the remaining states to send the FCS bytes and the closing flag. After the completion of transmission, SIU updates the STS and NSNR registers and interrupts the CPU.

If the contents of the TBL register were more than 1, the SIU transmits (TBL)-1 additional bytes from the internal RAM at starting address (TBS)+1 because it executes the DMA-LOOP state (TBL)-1 additional times. The byte processor should not be programmed to skip the DMA-LOOP state, because the transmission of FCS bytes is enabled in this state.

The maximum baud rate that can be used with these codes is calculated by adding the number of instruction cycles executed, during the standby mode, between each byte boundaries (see Table 2).

Using Equation 1, the maximum data rate, based on the transmit software, is 509 Kbps; However, the maximum count rate of the counter limits the data rate to 500 Kbps.

Fransmit Codes			Cycle
	•	•	
	•	•	
	•	•	
TRAN:	MOV	DPTR,#200H	
	MOV	R5,#OFFH	
	SETB	TBF	
	SETB	RTS	
LOOP:	MOVX	A, @DPTR	
	MOV	@R1.A	
	MOV	A, #OBOH	
WAIT1:	CJNE	A,SIUST,WAIT1	2
	INC	DPTR	
	MOVX	A,@DPTR	2
	MOV	@R1,A	
	DJNZ	R5,NEXTI	
	MOV	SIUST.#57H	•••~
	RET	51651, # 0711	
NEXTI:	MOV	CTUCT #570	0
NEALL:		SIUST, #57H	1
	MOV	A,#0BOH	· • • <u>+</u>
TND	JMP	WAIT1	•••
END			

#### Table 2. Codes for Long Frame Transmission

#### 6.2 Multidrop Application

Performance of long frame in addition to the features of the 8044 are described using a simple multidrop communication system in which three RUPIs, one as a master and the other two as secondary stations, transmit and receive long frames alternately (see Figure 16). All stations perform automatic zero bit insertion/deletion, NRZI decoding/encoding, Frame Check Sequence (FCS) generation/detection, and on-chip clock recovery at a data rate of 375 Kbps.

The primary and the secondary station's software code is given in Appendix B. These programs, for simplicity, assume only reception of information and supervisory frames. It is also assumed that the frames are received and transmitted in order. All stations use very similar transmit and receive routines. This code is written for standard SDLC frames (see Figure 7).

#### 6.2.1 POLLING SEQUENCE

The primary station, in Flexible mode, transmits a long frame (for this example, 255 I-bytes), polls one of the

#### FLEXIBILITY IN FRAME SIZE WITH THE 8044

secondary stations, and acknowledges a previously received frame simultaneously (see Figure 17). Both secondary stations, in Auto mode, detect the transmitted frame and check its address byte. One of the secondary stations receives the frame, stores the Information bytes in an external RAM buffer, and transmits the same data back to the primary. After reception of the frame, the primary polls and transmits a long frame to the other secondary station which will respond with the same long frame.

#### 6.2.2 HARDWARE

The schematic of the secondary station hardware is shown in Figure 18. The primary station's hardware is similar to the secondary station's hardware. The exception is in secondary stations only, where the RTS signal is inverted and tied to the interrupt 0 input pin (INT0). In the primary station, RTS is tied to CTS. At each station, software codes are stored in external EPROM (2732A). Static RAM (2Kx8) is used as external transmit/receive buffer. There is no hardware handshaking done between the stations. The serial clock is extracted from the data line using the on-chip phase locked loop.

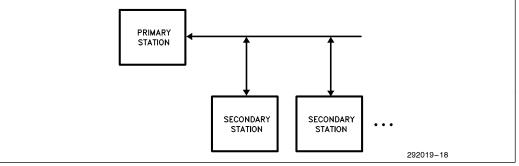


Figure 16. SDLC Multidrop Application Example

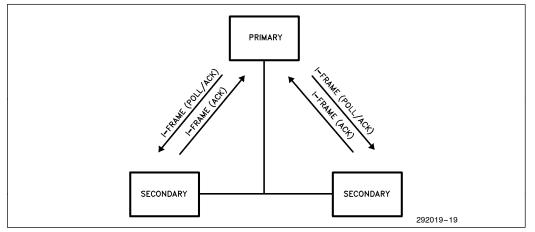
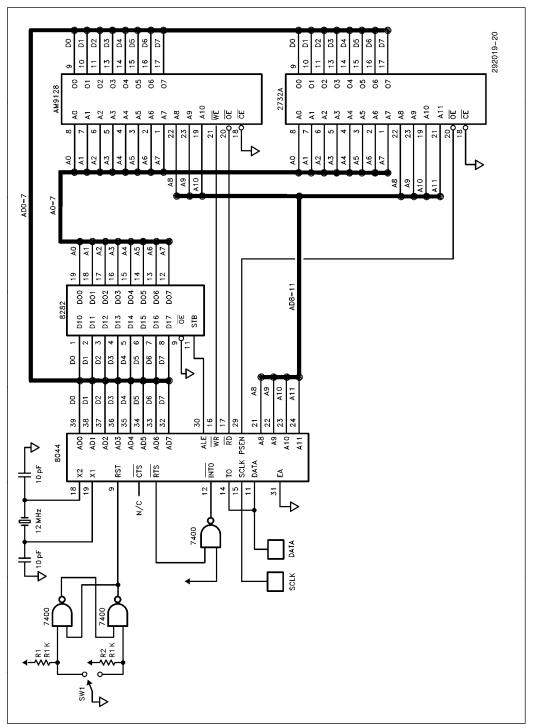


Figure 17. Polling Sequence Between the Primary and Secondary Stations



intel

Figure 18. Secondary Station Hardware

#### 6.2.3 PRIMARY SOFTWARE

#### **Main Routine**

During initialization (see Figure 19), the 8044 is set to Flexible mode, internally clocked at 375 Kbps, and configured to handle standard SDLC frames. The onchip receive and transmit buffer starting addresses and lengths are selected. The external transmit buffer is chosen from physical location 200H to location 2FFH (255 bytes). The external transmit buffer (external RAM) is loaded with data (FFH, FEH, FDH, FCH, ... 00H). Timer 0 is put in counter mode and set to priority 1. The counter register (TL0) is loaded such that interrupt occurs after 8 transitions on the data line. The Pre-Frame Sync option (setting bit 2 of the SMD register) is selected to guarantee at least 16 transitions before the opening flag of a frame.

#### FLEXIBILITY IN FRAME SIZE WITH THE 8044

The station address register (STAD) is loaded with address of one of the secondary stations. The RTS, TBF, and RBE bits of the STS register are simultaneously set and a call to the transmit routine follows. The transmit routine transmits the contents of the external transmit buffer. At the end of transmission, RTS and TBF are cleared by the SIU, and SIU interrupt occurs. In Flexible mode, SIU interrupt occurs after every transmission or reception of a frame.

#### SIU Interrupt Routine

In the SIU interrupt service routine (see Figure 20), SI is cleared and the RBE bit is checked. If RBE is set, a long frame has been transmitted. The first time through the SIU interrupt service routine, the RBE test indicates a long frame has been transmitted to one of the secondary stations. Therefore, the Counter is initialized

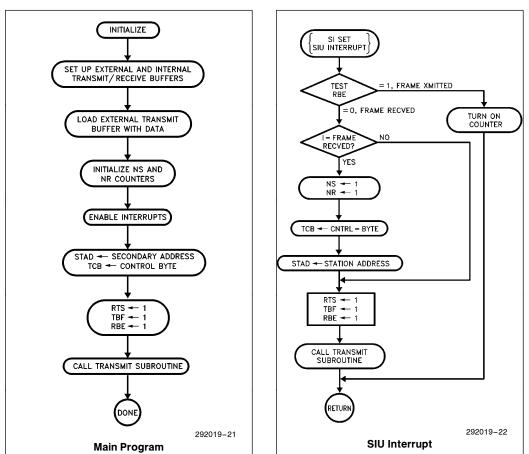


Figure 19. Primary Station Flow Charts

Figure 20. Primary Station Flow Charts



and turned on. The program returns from the interrupt routine before a frame appears on the communication channel.

When a frame appears on the communication line, counter interrupt occurs and the receive routine is executed to move the incoming bytes into the external RAM. After reception of the frame and return from the receive routine, SIU interrupt occurs again.

In the SIU interrupt routine, RBE is checked. Since the RBE bit is clear, a frame has been received. Therefore, the appropriate NS and NR counters are incremented and loaded into the TCB register (two pairs of internal RAM bytes keep track of NS and NR counts for the two secondary stations). Transmission of a frame to the next secondary stations). Transmission of a frame to the TBF bits. The chip is also put in receive mode (RBE set), and a call to transmit routine is made. After transmission, SIU interrupt occurs again, and the process continues.

#### 6.2.4 SECONDARY SOFTWARE

#### Main Routine

Both secondary stations have identical software (Appendix B). The only differences are the station addresses. Contents of the STAD register are 55H for one station and 44H for the other.

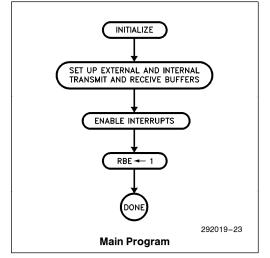


Figure 21. Secondary Station Flow Charts

During initialization, the chip is set to Auto mode, standard SDLC frame, and internally clocked at 375 Kbps (see Figure 21). Internal buffer registers: RBS, RBL, TBS, and TBL are initialized. The RBE bit is set and the counter 0 is turned on. The secondary is configured to transmit an Information frame every time it is polled. The RTS pin is inverted and tied to INT1 pin. External interrupt 1 is enabled and set to interrupt on low to high transition of the RTS signal. This will cause an interrupt (EX1 set) after a frame is transmitted. In the interrupt routine the CTS pin is cleared to prevent any automatic response from the secondary. If the CTS pin were not disabled, the secondary station would respond with a supervisory frame (RNR) since the TBF is set to zero by the SIU due to the acknowledge. In the SIU interrupt routine, the CTS pin is cleared after the TBF bit is set. If this option is not used, the primary should acknowledge the previously received frame and poll for the next frame in two separate transmissions.

#### **SIU Interrupt Routine**

When a frame is received, counter 0 interrupt occurs and the receive routine is executed (see Figure 22). If the incoming frame is addressed to the station, the information bytes are stored in external RAM; Otherwise, the program returns from the receive routine to perform other tasks. At the end of the frame, SIU interrupt occurs. In Auto mode, SIU interrupt occurs whenever an Information frame or a supervisory frame is received. Transmission will not cause an interrupt. In the SIU interrupt service routine, the AM bit of the STS is checked.

If AM bit is set, the interrupt is due to a frame whose address did not match with the address of the station. In this case, NFCS, AM, and the BOV bits are cleared, the RBE bit is set, the counter 0 is initialized and turned on, and program returns from the interrupt routine.

If AM bit is not set, a valid frame has been received and stored in the external RAM. TBF bit is set, CTS pin is activated, counter 0 is disabled and a call to transmit routine is made which transmits the contents of external transmit buffer. This frame also acknowledges the reception of the previously received frame (NS and NR are automatically incremented). Upon return from the transmit routine RBE is set and counter 0 is turned on, thereby putting the chip in the receive mode for another round of data exchange with the primary.

Note that, if the second station is in receive mode, and the counter is enabled and turned on, the CPU will be interrupted each time a frame is on the communication channel. If the frame is not addressed to the secondary station, the chip enters the receive routine, executes only a few lines of code (address comparison) and returns to perform other tasks. This interrupt will not occupy the CPU for more than two data byte periods (43 microseconds at 375 Kbps). At the end of the frame, the BOV bit is set by the SIU, and the SIU interrupt occurs. In the SIU interrupt service routine,

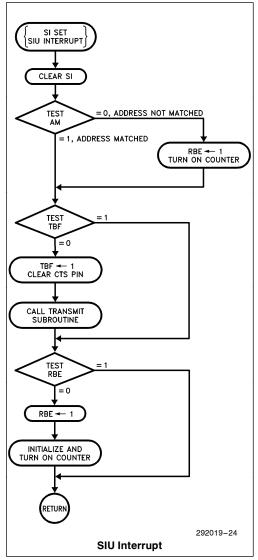


Figure 22. Secondary Station Flow Charts

the RBE bit is set and the counter is turned on which put the chip back in the receive mode.

#### **6.2.5 RECEIVE INTERRUPT ROUTINE**

Assembly code for the receive interrupt routine can be found in both primary and secondary software (Appendix B). The receive interrupt routine of the primary station is very similar to that of the primary station in example 1. In the following two sections the receive and transmit routine of the secondary stations are discussed.

#### FLEXIBILITY IN FRAME SIZE WITH THE 8044

In the receive interrupt service routine (see Figure 23), counter 0 is turned off, important registers are saved, receive buffer starting address and receive buffer length of the external RAM are set (do not confuse the external RAM settings with that of the internal RAM buffer.)

After reception of an opening flag, the byte processor jumps to the ADDRESS state and waits until the bit processor processes and moves the receiving address byte to SR. Then, the byte processor is triggered to execute the state. In the secondary stations, the CPU monitors the SIUST register for the ADDRESS state (SIUST = 08H). When the ADDRESS state is reached, the byte processor is moved to the next state (CONTROL state), and the ADDRESS state is skipped. Therefore, when the address byte is moved to SR, the byte processor executes the CONTROL state rather than the ADDRESS state and then jumps to the PUSH-1 state. The execution of the CONTROL state causes the contents of SR (the received address byte) to be loaded into the RCB register.

The CPU checks the contents of RCB with the contents of the STAD (Station Address) register. If they match, the receive routine continues to store the received Information bytes in the external RAM buffer; Otherwise, the byte processor is moved to the very last state (BOV-LOOP), and the program returns from the routine to perform other tasks. The byte processor executes the BOV-LOOP state in each byte boundary until the closing flag of the frame is reached. It then sets the BOV bit and interrupts the CPU (serial interrupt SI set). In the serial interrupt routine the counter 0 is turned back on, and the station is reset back to the receive mode (RBE set).

In Normal operation, in the ADDRESS state, the received address byte is automatically compared with the station address. If they match, the byte processor executes the remaining states; otherwise, the byte processor goes into the idle mode (SIUST = 01H) and waits for the opening flag of the next frame. In the expanded operation, this state is skipped to avoid idle mode. If the byte processor went into the idle mode, clocks which run the byte processor would be turned off, and the byte processor can not be moved to any other states by the CPU. When the byte processor is in idle mode, counter 0 can not be turned on immediately because counter interrupt occurs on the same frame, and program returns to the receive routine and stays there.

If the address byte matches the station address, the byte processor is moved to the CONTROL state again. This time, after execution of the CONTROL state the contents of RCB are the received control byte.

CPU investigates the type of received frame by checking the received control byte. If the receiving frame is not an information frame (i.e. Supervisory frame), execution of receive routine will be terminated to free the



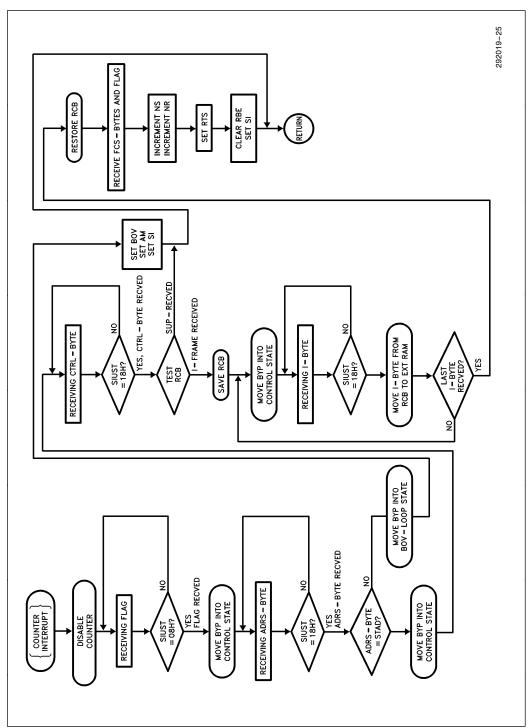


Figure 23. Receive Flow Chart (secondary station)

### intel

CPU. In Auto mode, the SIU checks the control byte and responds automatically in response to the supervisory frame.

After the control byte is received, it is saved in the stack. The byte processor is moved to the CONTROL state so that the next incoming byte will also be loaded into the RCB register. The byte processor remains in CONTROL state until a byte is processed by the bit processor and moved to SR. The byte processor is then triggered to move the contents of SR to the RCB register. The CPU monitors SIUST and waits until the first Information byte is loaded into the RCB register.

When byte processor reaches the PUSH-1 state (SIUST = 18H), RCB contains the first Information byte. The byte is moved to external RAM (receive buffer), and the byte processor is moved back to the CONTROL state. The process continues until all of the Information bytes are received. When all the Information bytes are received, the program returns from the routine. The byte processor automatically goes through the remaining states, updates the STS register, and interrupts the CPU as it would in Normal operation.

#### 6.2.6 TRANSMIT SUBROUTINE

The transmit subroutine codes can be found in the primary and the secondary software (Appendix B). The transmit subroutines of the Primary and secondary stations are identical. A call to transmit routine is made when the RTS and TBF bits of the STS register are set. In Auto mode, RTS is set automatically upon reception of a poll-frame (poll bit of the control byte is set).

In the transmit routine (see Figure 15), the starting address and the transmit buffer length of the external buffer are set. Then the CPU monitors the SIUST register for CONTROL state (SIUST = A8H). In the CONTROL state the bit processor transmits the control byte, while the byte processor goes into the standby mode after it has moved the contents of a location in the internal RAM addressed by the contents of Transmit Buffer Start (TBS) register to the RB register.

While the control byte is being transmitted and the byte processor is in standby, the CPU moves an Information

byte from external RAM to the internal RAM location addressed by TBS. The byte processor is then moved to CONTROL state. This will cause the byte processor, in the next byte boundary, to move the contents of the same location in the internal RAM to the RB register (see transmit state diagram.)

When this byte is being transmitted, the byte processor jumps to the DMA-LOOP state (SIUST = B0H) and waits. When the DMA-LOOP state is reached (CPU monitors SIUST for B0H), the CPU loads the next Information byte into the same location in the internal RAM and moves the byte processor to the CONTROL state before it gets to execute the DMA-LOOP state. Note that the same location in the internal RAM is used to transmit the subsequent Information bytes.

When all the Information bytes from the external RAM are transmitted, the byte processor is free to go through the remaining states so that it will transmit the FCS bytes and the closing flag.

#### 7.0 CONCLUSIONS

The RUPI, with addition of only a few bytes of code, can accept and transmit large frames with some compromise in the maximum data rate. It can be used in Auto or Flexible mode, with external or internal clocking, automatic CRC checking, and zero bit insertion/ deletion. In addition, almost all of the internal RAM is available to be used as general purpose registers, or in conjunction with the external RAM as transmit and receive buffers.

All in all, this feature opens up new areas of applications for this device. Besides transmitting/receiving long frames, it may now be possible to perform arithmetic operations or bit manipulation (e.g. data scrambling) while transmission or reception is taking place, resulting in high throughput. Transmission of continuous flags and transmission with no zero insertion are also possible.

In addition to unlimited frame size, an on-chip controller, automatic SDLC responses, full support of SDLC protocol, 192 bytes of internal RAM, and the highest data rate in self clocked mode compared to other chips make this product very attractive.



### APPENDIX A LISTING OF SOFTWARE MODULES FOR APPLICATION EXAMPLE 1

		344.PDF)		
; ASSI ; FLE:	EMBLY C XIBLE N	CODE FOR PRIMARY MODE; FCS OPTION	STATION (POINT TO POINT)	
	ORG	оон	; LOCATIONS 00 THRU 26H ARE USED	
	ORG	INIT OBH	; BY INTERRUPT SERVICE ROUTINES. ; VECTOR ADDRESS FOR TIMERO INT.	
	JMP	REC	, VICTOR ADDRESS FOR TIMERO INT.	
	ORG SJMP	23H SIINT	; VECTOR ADDRESS FOR SIU INT.	
;*****	*****	***** INITIA	LIZATION ****************************	
	ORG	26H		
INIT:	MOV MOV		; EXT CLOCK; PFS=NB=1	
	MOV	TBS,#20H TBL,#01H	; INT TRANSMIT BUFFER START ; INT TRANSMIT BUFFER LENGTH	
	MOV	20H, #55H	; STATION ADDRESS	
	MOV MOV	TMOD, #00000111B	COUNTER FUNCTION; MODE 3	
	MOV	STS.#11100000B	; EA=1; SI=1; ETO=1 ; TRANSMIT A FRAME	
DOT:	SJMP	DOT	; WAIT FOR AN INTERRUPT	
	CLR	SI	, INTERRUPT ROUTINE ****************	
	CLR JNB MOV MOV	SI RBE,RECVED TLO,#0F8H DPTR,#200H	; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START	
	CLR JNB MOV	SI RBE,RECVED TLO,#OF8H	; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0	202040, 28
SIINT: ; WHEN ; INTE:	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C	SI RBE,RECVED TLO,#0F8H DPTR,#200H R5,#0FFH TR0 4E APPEARS ON THE	; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH	292019–28
; WHEN ; INTE ; INTE	CLR JNB MOV MOV SETB RETI A FRAM RRUPT C	SI RBE, RECVED TLO, #0F8H DFTR, #200H R5, #0FFH TRO ME APPEARS ON THE CCURS. AFTER SEF	; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN SERIAL CHANNEL, COUNTER (RECEIVE) RVICING THE INTERRUPT ROUTINE, SIU	292019–28
; WHEN ; INTE ; INTE ; INTE RECVED	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C RRUPT C : MOV RETI	SI RBE,RECVED TLO,#0F8H DPTR,#200H RS,#0FFH TRO ME APPEARS ON THE DOCCURS. AFTER SEF DOCCURS. STS,#11100000B	; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN 2 SERIAL CHANNEL, COUNTER (RECEIVE) XVICING THE INTERRUPT ROUTINE, SIU ; TRANSMIT A FRAME	292019–28
HINT: WHEN INTE INTE RECVED	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C RRUPT C : MOV RETI	SI RBE,RECVED TLO,#OF8H DPTR,#200H R5,#OFFH TRO ME APPEARS ON THE DCCURS. AFTER SEF DCCURS. STS,#11100000B	<pre>; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN SERIAL CHANNEL, COUNTER (RECEIVE) WICING THE INTERRUPT ROUTINE, SIU ; TRANSMIT A FRAME ; RETURN YERRUPT ROUTINE ************************************</pre>	292019–28
WHEN ; INTE; ; INTE; ; INTE; RECVED ;******	CLR JNB MOV MOV SETB RETI A FRAM RRUPT C RRUPT C : MOV RETI	SI RBE,RECVED TLO,#0F8H DPTR,#200H R5,#0FFH TRO AE APPEARS ON THE OCCURS. AFTER SEF OCCURS. AFTER SEF OCCURS. STS,#11100000B	; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN 2 SERIAL CHANNEL, COUNTER (RECEIVE) XVICING THE INTERRUPT ROUTINE, SIU ; TRANSMIT A FRAME ; RETURN 2 SERUPT ROUTINE ****************	292019–28
WHEN ; WHEN ; INTE; RECVED ;****** REC: VAIT1:	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C RRUPT C : MOV RETI CLR MOV CJNE	SI RBE,RECVED TLO,#OF8H DPTR,#200H R5,#OFFH TRO ME APPEARS ON THE DCCURS. AFTER SEF DCCURS. STS,#11100000B	<pre>; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN SERIAL CHANNEL, COUNTER (RECEIVE) WICING THE INTERRUPT ROUTINE, SIU ; TRANSMIT A FRAME ; RETURN *ERRUPT ROUTINE ************************************</pre>	292019–28
WHEN ; WHEN ; INTE; RECVED ;***** REC: WAIT1: VEXT1:	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C RRUPT C : MOV RETI CLR MOV CJNE MOV	SI RBE, RECVED TLO, #0F8H DPTR, #200H R5, #0FFH TRO ACCURS. AFTER SEF DCCURS. STS, #11100000B ***** RECEIVE INT TRO A, #18H A, SIUST, WAIT1 SUUST, #0EFH A, #18H	; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN ? SERIAL CHANNEL, COUNTER (RECEIVE) WICING THE INTERRUPT ROUTINE, SIU ; TRANSMIT A FRAME ; RETURN ************************************	292019–28
WHEN ; WHEN ; INTE; RECVED ;***** REC: WAIT1: VEXT1:	CLR JNB MOV MOV SETB RETI A FRAN REUPT C RRUPT C RRUPT C RRUPT C CRUPT C CLR MOV CJNE	SI RBE,RECVED TLO,#0F8H DPTR,#200H R5,#0FFH TRO IE APPEARS ON THH DCCURS. AFTER SEF DCCURS. STS,#11100000B ***** RECEIVE INT TRO A,#18H A,SIUST,WAIT1 SUUST,#0EFH A,#18H A,SIUST,WAIT2	<pre>; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN SERIAL CHANNEL, COUNTER (RECEIVE) NVICING THE INTERRUPT ROUTINE, SIU ; TRANSMIT A FRAME ; RETURN VERRUPT ROUTINE ************************************</pre>	292019–28
WHEN ; WHEN ; INTE; RECVED ;***** REC: WAIT1: VEXT1:	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C RETI ******* CLR MOV CJNE MOV CJNE MOV	SI RBE, RECVED TLO, #0F8H DPTR, #200H R5, #0FFH TRO ACCURS. AFTER SEF DCCURS. STS, #11100000B ***** RECEIVE INT TRO A, #18H A, SIUST, WAIT1 SUUST, #0EFH A, #18H	<pre>; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN ? SERIAL CHANNEL, COUNTER (RECEIVE) WICING THE INTERRUPT ROUTINE, SIU ; TRANSMIT A FRAME ; RETURN ************************************</pre>	292019–28
SIINT: ; WHEN ; INTE ; INTE RECVED	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C RRUPT C RRUPT C RRUPT C RRUPT C CIR MOV CINE MOV CJNE MOV MOV MOV MOV	SI RBE, RECVED TLO, #0F8H DPTR, #200H R5, #0FFH TRO 4E APPEARS ON THE OCCURS. AFTER SEF OCCURS. STS, #11100000B ***** RECEIVE INT TRO A, #18H A, \$IUST, WAIT1 SIUST, #0EFH A, #18H A, SIUST, WAIT2 A, RCB @DPTR, A DEPTR	<pre>; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN ? SERIAL CHANNEL, COUNTER (RECEIVE) VVICING THE INTERRUPT ROUTINE, SIU ? TRANSMIT A FRAME ; RETURN *ERRUPT ROUTINE ************************************</pre>	292019–28
SIINT: ; WHEN ; INTE; ; INTE; RECVED ;****** REC: WAIT1: NEXTI:	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C RRUPT C RRUPT C RETI CLR MOV CJNE MOV CJNE MOV MOV MOV MOVX INC DJNZ	SI RBE,RECVED TLO,#OF8H DPTR,#200H R5,#OFFH TRO ME APPEARS ON THE OCCURS. AFTER SEF OCCURS. STS,#11100000B ***** RECEIVE INT TRO A, \$10ST, WAIT1 SUUST, #OEFH A, \$10ST, WAIT2 A, SCB @DPTR,A	<pre>; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN ; SERIAL CHANNEL, COUNTER (RECEIVE) WICING THE INTERRUPT ROUTINE, SIU ; TRANSMIT A FRAME ; RETURN ; TRANSMIT A FRAME ; RETURN *ERRUPT ROUTINE ************************************</pre>	292019–28
SIINT: ; WHEN ; INTE; ; INTE; RECVED ;****** REC: WAIT1: NEXTI:	CLR JNB MOV MOV SETB RETI A FRAN RRUPT C RRUPT C RRUPT C RRUPT C RRUPT C CIR MOV CINE MOV CJNE MOV MOV MOV MOV	SI RBE, RECVED TLO, #0F8H DPTR, #200H R5, #0FFH TRO 4E APPEARS ON THE OCCURS. AFTER SEF OCCURS. STS, #11100000B ***** RECEIVE INT TRO A, #18H A, \$IUST, WAIT1 SIUST, #0EFH A, #18H A, SIUST, WAIT2 A, RCB @DPTR, A DEPTR	<pre>; TRANSMITTED A FRAME ? ; YES, INITIALIZE COUNTER REGISTER ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; TURN ON COUNTER 0 ; RETURN ? SERIAL CHANNEL, COUNTER (RECEIVE) VVICING THE INTERRUPT ROUTINE, SIU ? TRANSMIT A FRAME ; RETURN *ERRUPT ROUTINE ************************************</pre>	292019–28

### intel

\$DEBUG NOMOD51 \$INCLUDE (REG44.PDF) ASSEMBLY CODE FOR SECONDARY STATION (POINT TO POINT) FLEXIBLE MODE; FCS OPTION ORG оон SJMP INIT ORG 23H SJMP SIINT ; VECTOR ADDRESS FOR SIU INT. ORG 26H INIT: MOV MOV LDRAM: MOV MOV DPTR, #200H MOV R3, #0FFH MOV A, R3 MOVX @DPTR, A ; EXT RAM XMIT BUFFER START ; EXT RAM XMIT BUFFER LENGHT ; LOAD EXT BUFFER WITH FFH, FEH, ... ; INCREMENT POINTER INC DPTR DJNZ R3,LDRAM 

 SMD,#00000110B
 ; EXT CLOCK; PFS=NB=1

 R1,#10H
 TBS,R1
 ; INT RAM XMIT BUFFER START

 TBL,#01H
 ; INT RAM XMIT BUFFER START

 RBS,#20H
 ; INT RAM RECEIVE BUFFER START

 RSL,#01H
 ; INT RAM RECEIVE BUFFER START

 RL,#01H
 ; INT RAM RECEIVE BUFFER LENGTH

 STAD,#55H
 ; STAD ADDRESS=55H

 TCON,#00H
 ; RESET TCON REGISTER

 IP,#0FFH
 ; ALL INTERRUPTS: PRIORITY 1

 STS,#0100000B
 ; RBE=1, RECEIVE A FRAME.

 DOT
 ; WAIT FOR AN INTERRUPT

 MOV SJMP DOT: ; SIU INTERRUPT OCCURS AT THE END OF A RECEIVED FRAME OR 292019-30 SIINT: CLR ; RECEIVED A FRAME? ; YES ; STATION ADDRESS MATCHED? ; YES, CALL TRANSMIT SUBROUTINE JB RBE, RETRN MOV A, STAD CJNE A, 20H, NMACH ACALL TRAN MOV ; TRANSMIT SUBROUTINE IS CALLED TO TRANSMIT A LONG FRAME. ; AFTER TRANSMISSION, SI IS SET. SIU INTERRRUPT IS SERVICED ; AFTER THE CURRENT ROUTINE (SIINT) IS COMPLETED. NMACH: SETB RBE RETRN: RETI ; RBE=1, RECEIVE A FRAME ; RETURN ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH TRAN: MOV ; EXT RAM RECEIVE BUFFER START ; EXT RAM RECEIVE BUFFER LENGTH ; SET TRANSMIT BUFFER FULL : ENABLE XMISSION OF AN I-FRAME ; MOVE THE IST I-BYTE INTO ACC. ; THEN, MOVE TO INT. RAM @ (TBS) ; DMA-LOOP STATE ; WAIT FOR XMISSION OF AN I-FRAME ; INCREMENT POINTER TO EXT. RAM ; ALL BYTES XMITTED? ; YES, EXCEPT THE LAST BYTE. ; MOVE DATA INTO INT. RAM @ (TBS) ; MOVE DATA INTO INT. RAM @ (TBS) ; MOVE DATA INTO INT. RAM @ (TBS) ; MOVE DATA ENTROL STATE ; THE SIU TRANSMITS THE FCS-BYTES ; AND THE CLOSING FLAG. ; RETURN DPTR, #200H MOV R5,#0FFH SETB TBF SETB RTS LOOP: A, @DPTR @Rl,A A, #0BOH A,SIUST,WAIT1 MOVX MOV MOV MOV WAIT1: CJNE INC DPTR DJNZ R5,NEXTI MOVX A,@DPTR MOV @R1,A SIUST, #57H MOV RET RETURN NEXTI: MOV JMP ; MOVE BYP TO CONTROL STATE (A8H). ; TRANSMIT THE NEXT BYTE SIUST,#57H LOOP END 292019-31



### APPENDIX B LISTING OF SOFTWARE MODULES FOR APPLICATION EXAMPLE 2

		DDE; FCS OPTION	ATION (MULTIPOINT)	
	ORG	00H	; LOCATIONS 00 THRU 26H ARE USED	
	SJMP ORG JMP	INIT OBH REC	; BY INTERRUPT SERVICE ROUTINES. ; VECTOR ADDRESS FOR TIMERO INT.	
	ORG SJMP	23H SIINT	; VECTOR ADDRESS FOR SIU INT.	
*****	*****	***** LOAD TRANSMI	T BUFFER WITH DATA **********	
	ORG	26H		
NIT:	MOV	DPTR,#200H	; EXT RAM XMIT BUFFER START	
	MOV	R3,#OFFH	; EXT RAM XMIT BUFFER LENGHT	
DRAM:	MOV	A,R3 @DPTR,A	; LOAD BUFFER WITH FFH, FEH, 00	
	INC	DPTR	; INCREMENT POINTER	
		R3,LDRAM		292019-32
;*****	*****	********** INITIA	LIZATION *****************	
	MOV	R0,#0BFH	; PUT ZEROS INTO INT. RAM	
LOOP:	MOV MOV	A,#00H	; FROM BFH TO 40H.	
	DEC	@RO,A RO	; MOVE 0 INTO RAM ADDRESSD BY RO	
	CJNE	R0, #40H, LOOP		
;		•		
	MOV	30H, #00H	; NS COUNTER FOR STAD=55	
	MOV MOV	31H, #00H 32H, #0FFH 33H, #0FFH	; NR COUNTER FOR STAD=55	
	MOV	33H #OFFH	; NS COUNTER FOR STAD=44 ; NR COUNTER FOR STAD=44	
	MOV	34H,#01H	; PONITER TO SECONDARY STATIONS	
	MOV		; INT. CLKED @ 375K; NRZI=1; PFS=1	
	MOV	RBS,#10H	; INT. RAM RECEIVE BUFFER START=10H	
	MOV	RBL,#00H	; INT. RAM RECEIVE BUFFER LENGTH=0	
	MOV MOV	R1,#20H TBS,R1	; INT. RAM XMIT BUFFER START=20H	
	MOV	TBL, #01H	; INT. RAM XMIT BUFFER LENGTH=1	
	MOV	NSNR, #OOH	; NS=NR=0	
	MOV		; COUNTER FUNCTION, MODE 3	
	MOV	TCON, #00H		
	MOV MOV		; EA=1; SI=1; ET0=1	
	MOV	IP,#00000010B	; TIMER 0 INT. PRIORITY 1 ; I-FRAME W/POLL	
	MOV	STAD, #55H	; ADDRESS BYTE=55H	
	MOV	STS,#11100000B	; RBE=TBF=RTS=1	
; TRANS	SMIT A	LONG FRAME WITH H	POLL BIT SET, WAIT FOR A	



J3 REF_RETURN ; RECEIVED A FRME ? W3 A/CC . Y45 LAAC AC UT RECEIVED A FRME ? W5 A/CC . Y55 LAAC AC UT RECEIVER FOR CONTRET FOR CONTRET PUTE W5 A/CC . W5 A/CC .	<pre>NOV A.RCS</pre>	SIINT:	CLR	SI	; CLEAR SI	
JB ACC.0.CETT ; IS IT AN I-FERRET , WITH THE ADDRESS AND ALL AND ALL ALL ALL ALL ALL ALL ALL ALL ALL AL	JB       ACC.0, ACT       ; IS TO MN I-FRAME ?       Content of the second se					
MOV       A.40.H       / YES         COUND A.14H, STUCENEMENT NE       / WOCK NOT THE LEAST 3 SIG. DITS         MALL A.400000111B       / MAKK OUT THE LEAST 3 SIG. DITS         MOV       JOH, A       / WOCK NEW NOT DACC.         MIKE A.400000111B       / MAKK OUT THE LEAST 3 SIG. DITS         MOV       JIH, A       / WOCK NEW NOT DACC.         MIK A.400000111B       / MAKK OUT THE LEAST 3 SIG. DITS         MOV JIH, A       / SHIFT 4 BITS TO LEFT         MIK A.4001       / BUTT 10 LETT         OLL A.4001       / BUTT 10 LETT         OLL A.4001       / BUTT 10 LETT         OKU JIH, 4001       / BUTT 10 LETT         OLL A.4001       / BUTT 10 LETT         OLL A.4001       / BUTT 10 LETT         OLL A.4001       / BUTT 10 LEAST 3 SIG. BITS         MOV SIA,40000111B       / MOVE NOT THE LEAST 3 SIG. BITS         MOV JSH,4001       / SAVE NR         SKIP       / A001         MOV A.31H       / SOVE NR INFO ACC.         MOV JSH,401       / MAKK OUT THE LEAST 3 SIG. BITS         MOV JSH,401       / MAKK OUT THE LEAST 3 SIG. BITS         MOV SHA,414       / SAVE NR         MOV TSH,414H       / MAKK OUT THE LEAST 3 SIG. BITS         MOV SIG,441H       / MAKK OUT THE	<pre>MOV A.401M ; YES CUPE A.300K ; MOVENEN INF INF INF ACC. Not A.4000C011B ; MONENDER THE LEAST 3 SIG. BITS MOV 300,A ; JANK OUT THE LEAST 3 SIG. BITS MOV 301,A ; JANK OUT THE LEAST 3 SIG. BITS MOV 301,A ; JANK OUT THE LEAST 3 SIG. BITS MOV 301,A ; JANK OUT THE LEAST 3 SIG. BITS MOV 301,A ; JANK OUT THE LEAST 3 SIG. BITS MOV 301,A ; JANK OUT THE LEAST 3 SIG. BITS MOV 301,A ; JANK OUT THE LEAST 3 SIG. BITS MOV 314,A ; JANK OUT THE LEAST 3 SIG. BITS MOV 314,A ; JANK OUT NO ACC. A , 400010000 ; JETT THE LEAST 3 SIG. BITS MOV 314,A ; MOVE ONLY INT ALL AND A SIG. BITS MOV 314,A ; MOVE ONLY INT ALL AND 3 SIG. BITS MOV 314,A ; MOVE NE INTO ACC. INC A , 3341 ; MOVE NE INTO ACC. INC A , 40000011B ; MOVE NE INTO ACC. INC A , 40000011B ; INCERNIT NE LEAST 3 SIG. BITS MOV 334,A ; SHIFT 4 BITS TO LEFT RE A ; MOVE NE INTO ACC. INC A , 40000011B ; INCERNIT NE LEAST 3 SIG. BITS MOV 334,A ; SHIFT 4 BITS TO LEFT RE A ; MOVE NE INTO ACC. INC A , 40000011B ; INCERNIT NE LEAST 3 SIG. BITS MOV 334,A ; SHIFT 4 BITS TO LEFT RE A ; THE FOLL BIT O LEFT RE A ; A ; THE FOLL BIT O LEFT RE A ; A ; CALL TEAMENT FOULTE O TO ACC. RE A ; A ; CALL TEAMENT FOULTE O TO ACC. RE A ; A ; CALL TEAMENT FOULTE O TO ACC. RE A ; A ; CALL TEAMENT FOULT FOULTE RE A ; CALL FAMENT ; CALL TEAMENT FOULTE RE A ; CALL TEAMENT ; CALL TEAMENT FOULTE RE A ; CALL TEAMENT FOULTE COUNTS CONTAGE SET FOR A ; COUNT A ; STATE ; THE FOLL BIT MOV STAD, 444H ; CALL TEAMENT FOULTER COUNTS CONTAGE SET FOR A ; COUNT A ; STATE ; THE RECEIVE BUTFER STATE MOV STAD, 444H ; THE RECEIVE BUTFER STATE MOV DETR A ; STATE A ; THE RECEIVE BUTFER STATE MOV STAD, 444H ; THE RECEIVE BUTFER STATE MOV STAD, 445H ; THE STATE THE CONTAGE STATE MOV A ; A ; A ; STATE A ; THE REC</pre>					
MOV A.30H / HOVE NS INTO ACC. INC A.0000111B / HOVE NS INTO ACC. MOV JOH, A.0000111B / MARE OF MOV A.31H / MOVE NS INTO ACC. INC A / 00000111 / MARE OF MOV A.31H / HOVE NS INTO ACC. INC A / 00000111 / MARE OF JEAST 3 SIG. BITS MOV A.4,00000011 / MARE OF TO LEFT RL A / SHIFT 4 BITS TO LEFT RL A / 00010000 : SET THE POLL BIT MOV TCB, A / 00010000 : SET THE POLL SIT MOV TCB, A / 00010000 : SET THE POLL SIT MOV TCB, A / 00010000 : SET THE POLL SIT MOV TCB, A / 0001011B / MARE OUT THE LEAST 3 SIG. BITS MOV STAD, #551H / NOVE NS COUNT TO ACC. RL A / 0001011B / MARE OUT THE LEAST 3 SIG. BITS MOV STAD, #551H / NOVE NS LEFT A. ALL A / 0001011B / MARE OUT THE LEAST 3 SIG. BITS MOV 344, #604 MOV A, 353H / NOVE NS COUNT TO ACC. MOV 344, #604 MOV 344, #604 MOV A, 353H / NOVE NS COUNT TO ACC. HC A / 5000011B / MARE OUT THE LEAST 3 SIG. BITS MOV 344, # 504 MOV ACG, A / 0000011B / MARE OUT THE LEAST 3 SIG. BITS MOV 344, # / 000011B / MARE OUT THE LEAST 3 SIG. BITS MOV 354, A / SAVE NS MOV 765, A / MOVE NS COUNT TO ACC. HC A / 50000000B / SHIFT 1 BIT TO LEFT RC A / 0000000 / SHIFT 1 BIT TO LEFT RC A / 0000000 / SHIFT 1 BIT TO LEFT RC A / 0000000 / SHIFT 1 BIT 00 LEFT RC A / 0000000 / SHIFT 1 BIT 00 LEFT RC A / 0000000 / SHIFT 1 BIT 00 LEFT RC A / 0000000 / SHIFT 1 BIT 00 LEFT RC A / 0000000 / SHIFT 1 BIT 00 LEFT RC A / 0000000 / SHIFT A BITS 0 LEFT RC A / 00000000 / SHIFT 1 BIT 00 LEFT RC A / 00000000 / SHIFT A BITS ACC. MOV TCB, A / 000000000 / SHIFT 1 BIT 00 LEFT RC A / 000000000 / SHIFT A BITS ACC. MOV STAD, #441 ETTIN A / 00000000000 / SHIFT A BITS ACC. MOV STAD, #441 HC A / 0000000000000000000000000000000000	MOV A.30H ; MOVE NS INTO ACC. IN A A0000011B ; INCERNITY NG MCL A 40000011B ; INCERNITY NG MCL A 40000011B ; INCERNITY NG MCL A 40000011B ; INCERNITY NG MCL A 400000011B ; INCERNITY NG MCL A 400000011B ; INCE AND YOU CALL MCL A 400000011B ; INCE AND YOU CALL MCL A 400010001 ; INCE NS COUNT TO ACC. HCL A 400010000 ; IST THE PLANT J SIG. BITS MCL A 400010000 ; IST THE FLANT J SIG. BITS MCL A 40001000 ; IST THE FLANT J SIG. BITS MCL A 40001000 ; IST THE FLANT J SIG. BITS MCL A 40001000 ; IST THE FLANT J SIG. BITS MCL A 40001000 ; IST THE FLANT J SIG. BITS MCL A 40001000 ; IST THE FLANT J SIG. BITS MCL A 4000101B ; MASK OUT THE LEAST J SIG. BITS MCL A 40000011B ; MASK OUT THE LEAST J SIG. BITS MCL A 40000011B ; MASK OUT THE LEAST J SIG. BITS MCL A 40000011B ; MASK OUT THE LEAST J SIG. BITS MCL A 40000011B ; MASK OUT THE LEAST J SIG. BITS MCL A 400000011B ; MASK OUT THE LEAST J SIG. BITS MCL A 4000000B ; INTER THE TO LETT MCL A ; INCERMENT NG ALL A 4000000B ; IST THE FOLL BIT MCV STAD, 444H ; COUNT TO ACC. RL A 40001000B ; INSECTION TO ACC. RL A ; JOUTE NG NRL, NRL, NRL, MSD, J, MSD, NG, 0 MCV STAD, 444H ; COUNT TO ACC. RL A ; JOUTE NG NRL, NRL, MRL, MSD, J, MSD, 0 MCV STAD, 444H ; COUNT TO ACC. RL A ; JOUTE NG NRL, NRL, MRL, MSD, J, MSD, 0 MCV STAD, 444H ; COUNT NG ACC. RL A ; JOUTE NG NRL, NRL, MRL, MSD, J, MSD, 0 MCV STAD, 444H ; COUNT NG ACC. RETT N ; CLL TAMANTI ROUTING MSTT INC, A ; SITT THE COUNT TO ACC. RETT N ; CLL TAMANTI ROUTING MSTT INC A ; SITT THE ADDIT TO LEFT GL A ; 40000 ; IST THE POLL BIT MCV A 418H ; CUN H SCUUNE NG COUNTRO MCV TAD, 40001 ; INTERLET AND SCUUNTS MCV A ; 40000 ; IST THE DOUTING ACC. MCV A ; 4000 ; IST THE ADDIT SCUUNTS MCT I, 4007 STAD ; 4007 ; TUNN OV COUNTRO ; STATE (LONT) MCV A ; 418H ; DIST NO LEFT RATE MCV A ; 418H ; DIST NO LEFT RATE MCV A ; 418H ; DIST NO LEFT RATE MCV A ; 418H ; DIST NO LEFT RATE SCUUNTS MCV A ; 418H ; DIST NO LEFT RATE SCUUNTS MCV A ; 418H ; DIST NO LEFT RATE SCUUNTS MCV A ; 418H ; MIT FOR ATT RE COUNTS COU			A,#01H		
<pre>NC A .00000111 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.1000001113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.1000001113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.4000001113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400001113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400001113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400001113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.40000113 / MOVE NS COUNT TO ACC. MAIL A.40000113 / MOVE NS COUNT TO IDET OKL A.40010003 / SET THE POLL BIT MOV STAD,459H MARK OUT THE LEAST 3 SIG. BITS MAIL A.40000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.40000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.400000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.4000000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.4000000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.4000000113 / MARK OUT THE LEAST 3 SIG. BITS MAIL A.40000000 / SET THE FOLL BIT MAIL A.40000000 / SET THE FOLL BIT MAIL A.40000000 / SET THE FOLL BIT MAIL A.4000000 / SET THE FOLL BIT MAIL A.400000 / SET THE FOLL BIT MAIL A.4000000 / SET THE FOLL BITE MAIL A.400000 / SET THE FOLL BITE MAIL A.40000000 / SET THE FOLL BITE MAIL A.400000000 / SET THE FLOOL BITE MAIL A.40000000 / SET THE FOLL BITE MAIL A.400000 / SET THE FOLL BITE MAIL A.400000000 / SET THE FOLL BITE MAIL A.400000000 / SET THE FLOOL BITE MAIL A.400000000 / SET THE FOLL BITE MAIL A.40000000000000000 / SET THE OLEFT MAIL A.4000000000000000000000000000000000000</pre>	<pre>HEC A intermediate interme</pre>		MOV		: MOVE NS INTO ACC.	
MOV 50H, A 111 / MOVE NS IN INC ACC. MOV A, 31H / MOVENS IN INC ACC. MOV A, 31H / MOVENS IN INC ACC. MOV A, 31H / MOVENS SHIFT 4 BLTS TO LEFT RL A / SHIFT 4 BLTS TO LEFT RL A / SHIFT 4 BLTS TO LEFT RL A / SHIFT 1 BLT TO LEFT MOV TCS, A / SOULD / SHIFT 1 BLT TO LEFT MOV TCS, A / SOULD / SHIFT 1 BLT TO LEFT MOV TCS, A / SOULD / SHIFT 1 BLT TO LEFT MOV TCS, A / SOULD / SHIFT 1 BLT TO LEFT MOV TCS, A / SOULD / SHIFT 1 BLT TO LEFT MOV TCS, A / SOULD / SHIFT 1 BLT TO LEFT MOV TCS, A / SOULD / SHIFT 1 BLT TO LEFT MOV STAD, 4501 / MOVE NS LOUNT TO ACC. RL A / SOULD / SHIFT 1 BLT TO LEFT MOV STAD, 4501 / MOVE NS LOUNT TO ACC. RL A / SOULD / SHIFT 1 BLT TO LEFT MOV STAD, 4501 / MOVE NS LOUNT TO ACC. SHIFT 1 MOVA / SAUL / MOVE NS LOUNT TO ACC. MOV STAD, 4501 / MOVE NS LOUT THE LEAST 3 SIG. BITS MOV 324, A / SOUDOOLD / MASK OUT THE LEAST 3 SIG. BITS MOV 324, A / SOUDOOLD / MASK OUT THE LEAST 3 SIG. BITS MOV 334, A / SOUDOOLD / MASK OUT THE LEAST 3 SIG. BITS MOV 334, A / SOUDOOLD / MASK OUT THE LEAST 3 SIG. BITS MOV 334, A / SOUDOOLD / SET THE LEAST 3 SIG. BITS MOV 334, A / SOUDOOLD / MASK OUT THE LEAST 3 SIG. BITS MOV 344 / SOULT / MOVE NS COUNT TO ACC. RL A / SOULD / SIG / MASK OUT THE LEAST 3 SIG. BITS MOV SIG, A / SOUDOOD / SET THE LOT LEFT RL A / SOULD / SIG / MASK OUT THE LEAST 3 SIG. BITS MOV SIG, A / SOUDOOD / SET THE LEAST 3 SIG. BITS MOV SIG, A / SOUDOOD / SET THE LEAST 3 SIG. BITS MOV SIG, A / SOUDOOD / SET THE LEAST 3 SIG. BITS MOV SIG, A / SOUDOOD / SET THE LEAST 3 SIG. BITS MOV SIG, A / SOUDOOD / SET THE LEAST 3 SIG. BITS MOV SIG, A / SOUDOOD / SET THE LEAST 3 SIG. BITS MOV SIG, A / SOUDOOD / SET THE LEAST 1 SIG. BITS MOV A / SOUDOOD / SET THE LEAST 1 SUCCOMTON SET THE CONTOL SITE MOV A / SOUD / SOUDOOD / SET THE LEAST 1 SUCCOMTON SET THE CONTOL SITE MOV A / SIG / SOUTH / DUVE NO COUNTER 0 CONTOL SITE SOUTH / SOUDOOD / SET THE LEAST 1 SUCCOMTON SET SOUTH / SOUT	<pre>MOV 30H, A 31H ; SAVE NS MILTER LEAST 3 SIG. BITS MOV A, 31H ; MOV N, A, 40000011B ; MOVE NS COUNT TO ACC. HL A ; SHIFT 4 BITS TO LEFT HL A ; SHIFT 4 BITS TO LEFT MOV TOB, A ; SHIFT 1 BIT TO LEFT MOV TOB, A ; SHIFT 4 BITS TO LEFT MOV TOB, A ; SHIFT 4 BITS TO LEFT MOV TOB, A ; SHIFT 4 BITS TO LEFT MOV 34H, #0000 34H, #0000 34H, #0000011B ; MOVE NS COUNT TO ACC. HL A ; MOV 33H, #0000011B ; MONE NG INTE INTO TOB REG. MOV 33H, #0000011B ; MONE NG INTE LEAST 3 SIG. BITS MOV 33H, #0000011B ; MONE NG INT THE LEAST 3 SIG. BITS MOV 33H, A ; SAVE NS MOV 33H, A ; SAVE NS COUNT TO ACC. HL A ; A ; SHIFT 4 BITS TO LEFT RE A ; A ; SHIFT 4 BITS TO LEFT RE A ; A ; SHIFT 4 BITS TO LEFT RE A ; A ; SHIFT 4 BITS TO LEFT RE A ; A ; SHIFT 4 BITS TO LEFT RE A ; A ; SHIFT 4 BITS TO LEFT RE A ; A ; SHIFT 4 BITS TO LEFT RE A ; A ; SHIFT 4 BITS TO LEFT NOV 34H, 6010 ; TURN OFF COUNTER 0 ; TURN OFF SOUNT TO ACC. HC MOV 95 FAD, 44H I ; MOV 4 ; HOINDOB ; ENALE TANNENTSSION MOV 35, 80FTH ; TOGEN NO, 1, NS2, NS1, NS0, 0 MOV 95, 80FTH ; TOGEN NO, 2, NST TER 2 COUNTS SETS EA ; TURN OFF COUNTER 0 MOV 7 D, 40FH ; HUTCRAUF AFTER 8 COUNTS SETS EA ; TURN OFF COUNTER 0 MOV 7 D, 41BH I ; MOVE DA ACCL TEAN REAL INTERVUETS MOV 7 D, 41BH ; MOVE DATA FOR THE LEAST 1 FOR ACC. MOV 7 D, 41BH ; FURTH I FAN TO THE EAST 1 FOR ACC. MOV 7 D, 41BH ; FURTH I ; FURTH STORE THE CONTROL STATE (ABNT ACCL MOV 7 D, 41BH ; FURTH I ; FURTH ATT TO THE EAST 1 FOR ACCL MOV 7 D, 41BH ; MOVE DATA FOR ACCL STATE (ABNT ACCL MOV 7 D, 41BH ; FURTH I</pre>		INC	A	; INCREMENT NS	
MOV A.31R / FOVE NR INTO ACC. HCC A HCC A	<pre>MOV A, 3iH ; FOVE NR INTO ACC. INC A, 40000011B ; MARK OUT THE LEAST 3 SIG. BITS MUL A, 40000011B ; MARK OUT THE LEAST 3 SIG. BITS MUL A, 40001000B ; SHIFT 4 BITS TO LEFT RL A RL A, 30H ; MOVE NS COUNT TO ACC. RL A, 400010000B ; SHIFT HE DIL DIT TO LEFT GRL A, 40001000B ; SHIFT HE DIL DIT TO RESC. WOV STAD, 45H ; MOVE NS COUNTED YTE INTO TCB RESC. WOV STAD, 45H ; MOVE NS INTO ACC. INC A, 400010011B ; MARK NOT THE LEAST 3 SIG. BITS WOV 3.41, 4000 JAM, 40000 JAM, 40000 JAM, 4000 JAM, 40000 JAM, 400</pre>					
<pre>HC A .40000111B ; MARK MOUT THE LEAST 3 SIG. BITS MUL A.4,00000111B ; MARK MOUT THE LEAST 3 SIG. BITS MUL A.4, 5 SHIFT 4 BITS TO LEFT RL A RL A RL A RL A RL A RL A RL A RL A</pre>	INC A       A #00000111       ; MARKENINT THE LEAST 3 SIG. BITS         MM AND A       ; SATE NOT THE LEAST 3 SIG. BITS         MM A       ; SATE NOT THE LEAST 3 SIG. BITS         RU A       ; SHIFT 4 BITS TO LEFT         RL A       ; SHIFT 1 BIT TO LEFT         RL A       ; SHIFT 1 BIT TO LEFT         RL A       ; SHIFT 1 BIT TO LEFT         NOV STAD, 455H       ; TCS: MARKINKA, NA, J, MS, MS, NS, 0         NOV STAD, 455H       ; MOTE NS COUNT TO LEAST         NOV STAD, 455H       ; MOTE NS LEAST 3 SIG. BITS         MOV 34H, 4000011B       ; MARK MOT THE LEAST 3 SIG. BITS         MOV 33H, 4000011B       ; MARK MOT THE LEAST 3 SIG. BITS         MOV 33H, 4000011B       ; SATE NR THE LEAST 3 SIG. BITS         MOV 33H, 4, 6000011B       ; SATE NR THE LEAST 3 SIG. BITS         MOV 34H, 40000011B       ; SATE NR THE LEAST 3 SIG. BITS         MOV 35H, A       ; SATE NR THE DLEAST         RL A       ; SATE THE POLL BIT         RL A       ; SATE NR THE LEAST 3 SIG. BITS         MOV 34H, 40100       ; SATE THE POLL BIT         MOV 35H, A       ; SATE NR THE LEAST 3 SIG. BITS         MOV 35H, A       ; SATE NR         RET A       ; SATE NR THE LEAST 3 SIG. BITS         MOV TG, A, 400010000B ; EST THE POLL BIT				; MOVE NR INTO ACC.	
NOV 318,A ; SAVE NR EL A ; SHIFT 4 BITS TO LEFT EL A RL A	MOV 31H, A ; SAVE N. RL A ; SHIT 4 BITS TO LEFT RL A RL A				; INCREMENT NR	
RL A ; SHIFT 4 BITS TO LEFT RL A RL A ORL A, JOOL 00000 ; SET THE POLL BIT NOVE NS COUNT TO ACC. RL A, HOOL 00000 ; SET THE POLL BIT NOV TCS, A, HOOL 0000 ; NEXT THE POLL BIT NOV TCS, A, HOOL 0000 ; NEXT THE POLL BIT NOV STAD, 4501 NOV STAD, 4501 JUN GETI	RL A ; SHIFT 4 BITS TO LEFT RL A RL A R					
<pre>RL Å RL Å RL Å, A RL Å, A</pre>	RL Å RL Å RU Å			A		
<pre>RL A ORL A,300 ; MOVE NS COUNT TO ACC. RL A ,400010000 ; SET THE POLLENT MOV TCS,A , THE POLLENT MOV TCS,A , 'NOVE NE CONTROL ENTE INTO CREEG. MOV STAD,495H NOV A4,400H JHM GETI</pre>	RL A OKL A, 30H ; MOVE NS COUNT TO ACC. RL A ; 0010000B ; SET THE FOLL BIT OK A, 40010000B ; SET THE FOLL BIT MOV TCB,A ; TCB: NEL NEL ALL TO LEFT MOV TCB,A ; TCB: NEL NEL ALL THE FOLL BIT MOV 34H, 4001 JMF GETI . P: MOK A, 32H ; MOVE NS LNTO ACC. ALL A, 40000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV 33H, 40000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV 33H, 400000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV 33H, 400000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV 33H, 400000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV 33H, 400000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV A, 33H ; MOVE NE INTO ACC. INC A ; 100000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV 33H, 400000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV 33H, 400000111B ; MASK NOT THE LEAST 3 SIG. BITS MOV 34H, 401H RL A ; SHIFT 1 BIT TO LEFT RL A ; SHIFT 1 BIT TO LEFT MOV TCB,A ; MOVE NCONTROL BYTE INTO TCB MOV 34H, 401H : MOVE SIGNED ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTINE SETB TAO SETB EA ; TOB INTERRUPT NOUTINE ************************************					
RL A CRL A,40010000 ; SETTHE TOLDET CRL A,40010000 ; SETTHE TOLDET ROV TCB,A ; TOTE THE TOLDET ROV 348,4000 JMP GETI : NOVE NETE INTO TOE REC. ; 292019-34 SKIP: MOV A,32H ; NOVE NS INTO ACC. ; 292019-34 SKIP: MOV A,32H ; NOVE NS INTO ACC. ; 292019-34 KOV 3,31H ; SAVE NR ; NOVE NS INTO ACC. ; 292019-34 KOV 3,31H ; SAVE NR ; 1000 RI INTO ACC. ; 292019-34 KOV 3,31H ; SAVE NR ; 1000 RI INTO ACC. ; 292019-34 KOV 33H,A ; SAVE NR ; 1000 RI INTO ACC. ; 292019-34 KOV 33H,A ; SAVE NR ; 1000 RI INTO ACC. ; 292019-35 KOV 33H,40H ; SAVE NR ; 1000 RI INTO ACC. ; 2020170 ACC. ; 202019-35 KOV STAD,444H ; SAVE NR ; 2021 KIE INTO TCB ; 2020170 ACC. ; 202019-35 KOV STAD,444H ; 100000B ; INABLE TRANSMISSION ; 202018-35 SETE TRO ; 100040000B ; INABLE TRANSMISSION ; 202019-35 KECL CLR FA ; 40071 ; 1015ABLE ALL INTERVENTS ; 202019-35 KECL CLR FA ; 1016000B ; TURM OF COUNTER 0 SETE FA ; 202019-35 KECL CLR FA ; 1015ABLE ALL INTERVENTS ; 202019-35 KECL CLR FAO ; TURM ON COUNTER 0 SETE FA ; 202019-35 KECL CLR FAO ; TURM ON COUNTER 0 KECL CLR FAO ; TURM ON COUNTER 0 KECL CLR FAO ; TURM ON COUNTER 0 KECL CLR FAO ; TURM SUP COUNTER 0 KECL CLR FAO ; SUST,	RL A ORL A, #00010000 SET THE POLL BYTE INTO CLEPT MOV TCB,A FOUR CONTROL BYTE INTO TCB REG. MOV STD,458T MOVE STD,458T MOVE STD,458T P. MOVA A,32R INCREMENT NACA. INC A A,00000111B ; MASK OUT THE LEAST 3 SIG. BITS MAIL A,0000011B ; MASK OUT THE LEAST 3 SIG. BITS MAIL A,0000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV 33H,A MIL A,00000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV 33H,A MIL A,00000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV 33H,A MIL A,00000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV 33H,A RL A RL A R			A		
ORL A, #00010000B ; SET THE FOLL BIT MOV STAD, #55H ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 STD: #05H ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 STD: #05H ; TCR: NR2,NR1,NR0,1,NS2,NS1,NS0,0 STD: #05H ; TCR: KR2,NR1,NR0,1,NS2,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS0,1,NS2,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS0,1,NS2,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS0,1,NS2,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS1,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS1,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS1,NS1,NS1,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS1,NS1,NS1,NS1,NS1,NS0,0 ; TCR: KR2,NS1,NS1,NS1,NS1,NS1,NS1,NS1,NS0,0 ; TCR: KR3,NS1,NS1,NS1,NS1,NS1,NS1,NS1,NS1,NS1,NS1	ORL A. #00010000B ; SET THE FOLL NIT.       292019-34         MOV STAD, #55H       ; TCB: NR2, NR1, NR0, 1, NS2, NS1, NS0, 0         NOV STAD, #55H       ; TCB: NR2, NR1, NR0, 1, NS2, NS1, NS0, 0         NOV STAD, #55H       ; TCB: NR2, NR1, NR0, 1, NS2, NS1, NS0, 0         NOV STAD, #55H       ; NOVE NS INTO ACC.         P: MOV A, 12H       ; NOVE NS INTO ACC.         NIC A, 40000111B       ; MARE NR INTO ACC.         NU A, 33H       ; NOVE NR INTO ACC.         NU A, 33H, ; SAVE NS       ; NOVE NR INTO ACC.         NU A, 33H, ; SAVE NS       ; NOVE NR INTO ACC.         NU A, 13H, ; SAVE NS       ; NOVE NR INTO ACC.         NU A, 13H, ; SOUTT TO LEAST 3 SIG. BITS       ; NOVE AND ACC.         NU A, 14H, ; NAUE NR INTO ACC.       ; TGB: NR2, NR1, NR0, 1, NS2, NS1, NS0, 0         WOY AGA, 4000       ; SET THE FOLL DITT         MOV TCB, A       ; NOVE CONTROL BYTE INTO TCB         WOY TCB, A       ; DISABLE ALL INTERCUPTS         MOV STAD, #44H       ; COLL TANNON ; COUNTER 0         SETS TRA       ; TURN ON COUNTER 0         SETB TRA       ; TURN OFF COUNTER 0         SETB TRA       ; SAVE NAM SECEIVE DUFFER START         MOV PDTF, #400H       ; EXT. RAM RECEIVE DUFFER START         MOV STUDT, #057 ; INCREMENT FO CANT AD XAT. RAM <t< td=""><td></td><td></td><td></td><td></td><td></td></t<>					
MOV       TCB, A       ; MOVE CONTROL BYTE INTO TCB REC.         MOV       STAD, 455H       ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0         MOV       Gain       Good       232019-34         NULP:       MOV A       ; MOVE NS INTO ACC.       232019-34         NULP:       MOV A, 31H       ; MOVE NS INTO ACC.       341,4001         MOV 32H,A       ; SAVE NS       SIG. BITS         MOV 32H,A       ; SAVE NS       SIG. BITS         MOV A, 33H       ; MOVE NT THE LEAST 3 SIG. BITS       341,400000111B         MIL A, 400000111B       ; MASK OUT THE LEAST 3 SIG. BITS       341,40         MUL A, 4,0000011B       ; BASK OUT TO ACC.       341,40         RL A       ; BHIFT & BITS TO LEFT       RL         NOV TCB,A       ; SHIFT IST TOL LEFT       341,401         MOV TCB,A       ; CTB, THENFOLD ENT INTO TCB       341,401         MOV STS, 41100000B       ; BHABLE TRANSHITSION       342,401         ACALL TRAM       ; CALL TRANSHIT ROUTINE       282019-35         MOV TLO, 40FBH       ; INTERMUPT ATTER & CONTS       342,401         MOV TLO, 40FH       ; INTERMUPT ATTER & CONTS       342,401         MOV TLO, 40FH       ; INTERMUPT ATTER & CONTS       342,401         MOV TLO, 40FH       ; INTE	MOV       TCS, A       ; MOVE CONTEND PYTE INTO TOE REC.       ; TCB: NR., NRL, NRC, 1, NS2, NS1, NS0, 0         MOV       STAT OUT       ; TCB: NR., NRL, NRC, 1, NS2, NS1, NS0, 0       202019-34         P:       MOV       A, 221       ; MOVENENT NS       202019-34         P:       MOV       A, 221       ; MOVENENT NS       30.         MOV       A, 231       ; GAVEN NS       GAU       BUT         MOV       A, 131       ; GAVEN NS       GAU       BUT         MOV       A, 314       ; MOVEN NS       BLAST S IG. BITS       BUT         MOV       331, A       ; SAVEN NS       GAU       BITS       BUT         MOV       331, A       ; SAVEN NS       GAU       BITS       BUT					
MOV STAD, #55H MOV GETI MOV GETI MUT GETI MUT A:240 SETUR: MOV A:23H ; MOVE NS INTO ACC. MOV A:23H, ; SAVE NS MOV A:33H ; MOVE NT THE LEAST 3 SIG. BITS MOV 32H,A MOV A:33H ; MOVE NT HIND ACC. INC A ; INCREMENT NG AND A:40000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV 32H,A A ; DAVE NT HIND ACC. INC A ; INCREMENT NG AND A:40000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV A:33H ; MOVE NS COUNT TO ACC. RL A ; DEHIT 4 BITS TO LEFT NG A; A ; DEHIT 4 BITS TO LEFT NG A; A ; DEHIT 4 BITS TO LEFT NG A; A ; DOUDOOD ; SETT THE NEASH SIG. BITS MOV STAD, #44 ; DEHIT 1 & DITS TO LEFT NG TCG,A ; MOVE NS COUNT TO ACC. RL A ; A ; DEHIT 1 & DITS TO LEFT NG TCG,A ; SHIFT 1 BIT TO LEFT NG TCG,A ; CONSIDER SIGNAL ALL NEASH SIGNA ACALL TRAN ; CALL TRANSMIT ROUTINE SETI KOV STS, #1100000B ; NAME ALL INTERRUPTS MOV TG, 40FBH ; INTERRUPT AFTER 8 COUNTS SETE EA ; TURN ON COUNTER 0 NOV TG, 40FFH ; DITS COUNTER 0 NOV TG, 40FFH ; DITS COUNTER 0 NOV TG, 40FFH ; DITS COUNTER 0 NOV TG, 40FFH ; EXT. RAM RECEIVE BUFFER LENGTH NOV TG, 40FFH ; EXT. RAM RECEIVE BUFFER START NOV A: S.50FF ; EA ; SWF RECEIVE BUFFER START NOV A: S.50FF ; FA ; SWF RECEIVE BUFFER LENGTH NOV A: S.50FF ; FA ; SWF RECEIVE BUFFER LENGTH NOV A: S.50FF ; FA ; SWF RECEIVE BUFFER LENGTH NOV A: SUBST, 40FFH ; EXT. RAM RECEIVE BUFFER START NOV A: SUBST, 40FFH ; EXT. RAM RECEIVE BUFFER START NOV A: SUBST, 40FFH ; EXT. RAM RECEIVE BUFFER START NOV A: SUBST, 40FFH ; EXT. RAM RECEIVE BUFFER LENGTH NOV A: SUBST, 40FFH ; EXT. RAM STADIES START NOV A: SUBST, 40FFH ; EXT. RAM STADIES START [10]). HOV A: SUBST, 40FFH ; EXT. RAM STADIES START [10]). HOV A: A, SIUST, MAITZ ; WAIT FOR AN I-BYTE NOV A: A, SIUST, MAITZ ; WAIT FOR AN I-BYTE NOV A: A, SUBST, WAITZ ; WAIT FOR AN I-BYTE NOV A: A, SUBST, WAITZ ; WAIT FOR AN I-BYTE NOV A: A, BAETI ; IS IT THE LAST I-BYTE NOA CC. MOV A: A, BOFFH ; MOVE BATA FOR MA I-BYTE NOV A: A, BAETI ; IS IT THE LAST I-BYTE MANAMINE HINN B: A, BENTI ; IS IT THE LAST I-BYTE MAIT NOV A: A, SUBST, WAIT ; IS IT THE LAS	MOY STAD, #55H MOY STAD, #55H MOY GET: 292019-34 P: MOY A.22H ; HOVE NG INTO ACC. 292019-34 MOY A.22H ; HOVE NG INTO ACC. NAL A, #0000011B ; MAGK OUT THE LEAST 3 SIG. BITS MOY A,33H ; MOVE NG INTO ACC. INC A ; INCREMENT NG AND A, 4000011B ; MAGK OUT THE LEAST 3 SIG. BITS MOY 33H,A ; SAVE NG MOY 35H,A ; SAVE NG MOY 35H,A ; SAVE NG MOY CGA,A ; MOVE NG CONTOL EFT MOY TCA,A ; MOVE NG CONTOL BITS INTO TCB MOY STAD, #44H MOY CGA,A ; MOVE CONTOL BITE INTO TCB MOY 35H, #01H MOY 5H, #1010000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTING MOY STA, #44H MOY A: #4100000B ; INTERANDPA AFTER & CONTS SETE DA RETI ************************************					
MOV 344, \$ 100 TMP GETI 222019-34 SKTF: MOV A, 32H ; NOVE NS INTO ACC. INC A ; INCREMENT NS ANL A, \$00000111B ; MARK OUT THE LEAST 3 SIG. BITS MOV 32H,A ; SAVE NS MOV 33H,A ; SAVE NS MOV 33H,A ; SAVE NS MOV 33H,A ; SAVE NS MOV 33H,A ; SAVE NS RC A ; INCREMENT NS ANL A, \$0000111B ; MARK OUT THE LEAST 3 SIG. BITS MOV 33H,A ; SAVE NS RC A ; SHIFT 4 BITS TO LEFT RL A ; CAL X ; SHIFT 4 BITS TO LEFT RL A ; CAL X, SAVE NS MOV STAD, \$44H MOV STAD, \$44H MOV STAD, \$44H MOV STAD, \$44H MOV TCB,A ; MOVE CONTROL BITE INTO TCB RETI TRAN ; CALL TRANSMIT ROUTINE RETI RECEIVE INTERCUPT AFTER 8 COUNTS SETE EA REC: CLE TRA ; DISABLE ALL INTERCUPTS MOV STAD, \$460H ; INTERCUPT AFTER 8 COUNTS SETE EA REC: CLE TRA ; JUTSABLE ALL INTERCUPTS MOV DTD, \$400H ; EVT. MAR PEDELVE BUTFER START MOV A, \$18H ; MAIT FOR THE CONTROL STATE (10H). ARETI RECEIVE INTERCUPT ROUTINE ************************************	MOVJAH, éGOR292019-34JMMGETI: INCREMENT NSANCA, 4000001118: MASK OUT THE LEAST 3 SIG. BITSMOVA, 311: DICREMENT NSMALA, 4000001118: MASK OUT THE LEAST 3 SIG. BITSMOVA, 311: DICREMENT NSMALA, 4000001118: MASK OUT THE LEAST 3 SIG. BITSMOV33H, A: SAVE NSMOV33H, A: SAVE NSRLA: DICREMENT NSMOV33H, A: SAVE NSRLA: SHIFT 4 BITS TO LEFTRLA: DICREMENT NSORLA, 4000100008: SET THE POLL BITMOVSTAD, #44H: DISTANTSIONMOVSTAD, #44H: DISTANT STATEMOVSTAD, #1000000: ENABLE ALL INTERPUTESMOVSTAD, #14H: DISTANT STATEMOVSTAD, #14H: DISTANT STATEMOVSTAD, #1000000: TURN OF COUNTER 0MOVSETB FRO: TURN OF COUNTER 0MOV: STATE TRA: 292019-35***********************************		MOV		; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0	
<pre>SMP GETT</pre>	JMP GETT 232019-34 MOV A,22H : MOVE NS INTO ACC. INC A : INCREMENT NS ANL A,400000111 : MASK OUT THE LEAST 3 SIG. BITS MOV 3.1.,A : SAVE NS NOVE NS INTO ACC. INC A : INCREMENT NS ANL A,400000111B : MASK OUT THE LEAST 3 SIG. BITS MOV 3.3.,A : SAVE NR RL A : SHIFT 4 BITS TO LEFT RL A : SHIFT 4 BITS TO LEFT RL A : SHIFT 1 BIT TO LEFT NOV TCB,A : MOVE NS COUNT TO ACC. RL A : SHIFT 1 BIT TO LEFT NOV TCB,A : MOVE CONTROL BYTE INTO ACG. RL A : SHIFT 1 BIT TO LEFT NOV TCB,A : MOVE CONTROL BYTE INTO TCB MOV 3.40,0001000B : SET THE POLL BIT MOV TCB,4 : MOVE CONTROL BYTE INTO TCB NOV TCB,4 : MOVE CONTROL BYTE INTO TCB NOV TCB,4 : MOVE CONTROL BYTE INTO TCB ACALL TRAN : CALL TRANSMIT ROUTINE RETI MOV TCB,6 : THEN FOLL BIT MOV A,4 : SHIF T, NAT FOLL STATE FUSH RCB : SAVE T. RAM RECEIVE BUFFER START MOV A,4 : SHIF T, WAIT FOR THE CONTROL STATE (10H). TI HOV SIGN 40 : FXT. RAM RECEIVE BUFFER START MOV A,4 : SHIF T, WAIT FOR THE CONTROL STATE (10H). TI HOV SIGN 40 : FXT. RAM TRANSMIT SUFFER START MOV A,4 : SHIF T, WAIT FOR THE CONTROL STATE (10H). TI HOV SIGN 40 : FXT. RAM TRANSMIT SUFFER START MOV A, ACB : A : MOVE DATA TO EXT. RAM INC DFTR : INCREMENT FIT TO EXTERNAL RAM DANZ RS,NEXT : IS IT THE LAST I-BYTE RETI MOV SIDST,457H : EXT. RAM TRANSMIT SUFFER START MOV A,40ABH : CONTROL STATE NOV A,40ABH : INCREMENT FOINTE					
<pre>NC A ALC A, #000001115 ; MARK OUT THE LEAST 3 SIG. BITS MOV 3241, A ; SAVE NS MOV 341, A ; SAVE NS MOV 241, A ; SAVE NS MOV 241, A ; SAVE NS MOV A, 3314 ; MOVE NR INTO ACC. INC A ; MOODOO1115 ; MARK OUT THE LEAST 3 SIG. BITS MOV 7 31, A ; SHIFT 4 BITS TO LEFT RL A ; A ; SHIFT 4 BITS TO LEFT RL A ; A ; SHIFT 1 BIT TO LEFT ORL A, #0001000B ; SIST THE POLL BIT ORL A, #0010000B ; SIST THE POLL BIT ORL A, #0010000B ; SIST THE POLL BIT ORL A, #0010000B ; SIST THE POLL BIT MOV STAD, 444H MOV STAD, 441H MOV STAD, 441H ETII: MOV STAD, 441H MOV STAD, 441H ETII: MOV STAD, 441H MOV DETER, 441D MOV A, 4, 10H MOV A, 4, 10H</pre>	<pre>NRC A' INCREMENT NS ANLL A, 400000111 ; MASK OUT THE LEAST 3 SIG. BITS MOV A, 33H ; MOVE NR INTO ACC. INC A , 400000111 ; MASK OUT THE LEAST 3 SIG. BITS MOV A, 400000111 ; MASK OUT THE LEAST 3 SIG. BITS MOV A, 400000111 ; MASK OUT THE LEAST 3 SIG. BITS MOV A, 400000111 ; MASK OUT THE LEAST 3 SIG. BITS MOV 31 H, A ; SHIFT 1 BIT TO LEFT RL A , sHIFT 1 BIT TO LEFT RL A , sHIFT 1 BIT TO LEFT ORL A, 40010000 ; SET THE FOLL BIT MOV TGD, A ; MOVE NS COUNT TO ACC. RL A, 400010000 ; SET THE FOLL BIT MOV TGD, A ; MOVE NS COUNT TO ACC. RL A, 40010000 ; SET THE FOLL BIT MOV TGD, A ; MOVE CONTROL BITE INTO TCB MOV STAD, 44AH ; TCB: NR2, NR3, NR0, 1, NS2, NS1, NS0, 0 MOV 35TA, 44AH ; TCB: NR2, NR1, NR0, 1, NS2, NS1, NS0, 0 MOV 35TA, 44AH ; TCB: INTERNUPTS MOV TLO, 40FBH ; INTERNUPT AFTER 8 COUNTS SETE TRO ; JUNN ON CONTER 0 RCTI URN: CLR TRA ; DISABLE ALL INTERNUPTS MOV TLO, 40FBH ; INTERNUPT AFTER 8 COUNTS SETE TRO ; JUNN ON CONTER 0 RETT WOV DTD, 4400H ; FIXT. RAM RECEIVE BUFFER START MOV DTTR, 4400H ; FIXT. RAM RECEIVE BUFFER START MOV A, 418H ; PUSH-1 STATE I: CLR TRO ; JUNN OFF COUNTER 0 RETT FUSH RCB ; SAVE RECIVE BUFFER LENGTH MOV A, 418H ; PUSH-1 STATE FUSH RCB ; SAVE RECIVE BUFFER LENGTH MOV A, 418H ; PUSH-1 STATE FUSH RCB ; SAVE RECIVE BUFFER LENGTH MOV A, 418H ; FUSH-1 STATE FUSH RCB ; SAVE RECIVE BUFFER LENGTH MOV A, 418H ; FUSH-1 STATE FUSH RCB ; MOVE RECEIVES DIFFER LENGTH MOV A, 418H ; FUSH-1 STATE FUSH RCB ; MOVE RECEIVES TATE INTO ACC. MOV A, 418H ; FUSH-1 STATE FUSH RCB ; MOVE RECEIVES TATE INTO ACC. MOV A, 418H ; FUSH-1 STATE FUSH RCB ; MOVE RECEIVES TATE INTO ACC. MOV A, 418H ; FUSH-1 STATE FUSH RCB ; MOVE RECEIVES TO EXTERNAL FRAM NOV A, 418H ; FUSH-1 STATE FUSH RCB ; MOVE RECEIVES TO EXTERNAL FRAM NOV A, 418H ; FUSH-1 STATE FUSH RCB ; YET, RAM TRANSMIT BUFFER LENGTH MOV A, 418H ; CONTROL STATE [USHFER START MOV S, 4185, MATT ; MAIT FOR ATHE DEFTER START MOV S, 4185, MATT ; KAIT FOR ATH FOR START MOV S, 4185, MATT ; KAIT FOR ATH TO EXTERNAL FRAM NOV S, 4185, MATT ; KA</pre>			GETI		292019-34
ANL A, #00000111B ; MASK OUT THE LEAST 3 SIG. BITS MOV A, 33H, A ; SAVE NS MOV 33H, A ; SAVE NS NS A, A ; SHIF7 4 BITS TO LEFT NS A, A ; SHIF7 4 BITS TO LEFT NS A, #0010000B ; SET THE FOLD BIT MOV TCD, A ; TOCE NS2, NIN, NRO, 1, NS2, NS1, NS0, 0 MOV SETS, #1110000B ; SET THE FOLD BIT MOV TCD, 6FEH ; INTERNET FARMENTSSION ACALL TEAM ; DISABLE ALL INTERNUES NOV SITS, #1110000B ; SET START NOV TLO, 6FEH ; INTERNET FARMENTSSION ACALL TEAM ; DISABLE ALL INTERNUES NOV TLO, 6FEH ; INTERNET FARMENTSSION ACALL TEAM ; DISABLE ALL INTERNUES NOV TLO, 6FEH ; INTERNET FRE & COUNTS SETE TA NOV TLO, 6FEH ; FURENCET ACTIVE SUFFER START NOV CE, 40FFH ; EXT. RAM RECEIVE BUFFER START MOV DETR, #400H ; EXT. RAM RECEIVE SUFFER START MOV DETR, #400H ; EXT. RAM RECEIVE SUFFER START MOV DETR, #400H ; FUSH. MSYME CONTROL BYTE FUSH RCS NATT ; SAVE RECEIVE ONTROL STATE (10H). AJ.FISH ALT : MOV SUET, 40FFH ; FUSH. MSYME TO CONTROL STATE (10H). MOV A, #18H ; FUSH. MSYME THE CONTROL STATE (10H). MOV A, #18H ; FUSH. MSYME THE CONTROL STATE (10H). MOV A, #18H ; FUSH. MOVE DATA TO EXT. RAM DINZ R5, NEXTI ; IS IT THE LAST I-BYTE MOV A, & RCS ; MOVE BRAT TO EXT. RAM TRANSMIT SUBROUTINE ************************************	ANL A, 40000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV A, 33H ; MOVE NE INTO ACC. INC A , 1000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV 3,34H ; SAVE NE ANL A, 40000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV 31H,A ; SAVE NE RE A ; SHIFT 4 BITS TO LEFT H A ; RE A ; SHIFT 4 BITS TO LEFT H A ; NOW SIG. A, 33H ; MOVE NS COUNT TO ACC. RL A ; 00010000 ; SST THE FOLL BUT MOV 7CB,A ; MOVE NS COUNT TO ACC. RL A ; 00010000 ; SST THE FOLL BUT MOV TCB,A ; MOVE NS COUNT TO ACC. RL A ; 00010000 ; SST THE FOLL BUT MOV TCB,A ; MOVE CONTROL BYTE INTO TCB MOV SIG., 40001000B ; SST THE FOLL BUT MOV SIG., 40001000B ; CALL TRANSMITSION KCON SIG., 6110000B ; CALL TRANSMIT ROUTINE RETI * CLR EA ; DISABLE ALL INTERRUPTS MOV TCD,40FBH ; INTERRUPT FREE & COUNTS SETB EA RETI * RECEIVE INTERRUPT ROUTINE ************************************	SKIP:				
<pre>MOV 32E,A ; SAVE NS MOV A,33H ; MOVE NE INTO ACC. INC A at ; INCERMENT NR ALL A,400000111B ; MASK OUT THE LEAST 3 SIG. BITS MOV 33E,A ; SAVE NR RL A ; SAVE NR RL A ; SAVE NR RL A ; SAVE NR RL A RL A RL A RL A RL A ; SHIFT 4 BITS TO LEFT RL A RL A ; SHIFT 4 BITS TO LEFT RL A ; SHIFT 1 BITS TO LEFT RL A ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV TCB,A ; MOVE ONTOL BYTE INTO TCB RC AT ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV STAD,444H MOV 3TA,401H SETI: MOV STS,41100000B ; ENABLE TRANSMISSION RCALL TRAN ; CALL TRANSMIR NOUTINE SETB FR0 ; INTERUPT AFTER 8 COUNTS SETB ERA ; DISABLE ALL INTERUPTS HOV TLO,40FBH ; INTERUPT AFTER 8 COUNTS SETB EA ; DISABLE ALL INTERUPTS SETB EA ; DUSABLE ALL INTERUPTS SETB EA ; SUUST,40CH ; EXT. RAM RECEIVE BUFFER START MOV DPTR,440CH ; EXT. RAM RECEIVE BUFFER START MOV RS,40FFH ; EXT. RAM RECEIVE BUFFER LENGTH MOV A,418H ; PUSH-1 STATE FUSH RCB ; SAVE RECEIVE CONTROL SYTE FUSH RCB ; SAVE RECEIVE CONTROL SYTE FUSH RCB ; SAVE RECEIVE CONTROL SYTE FUSH RCB ; SAVE RECEIVE CONTROL STATE(10H). MOV SIUST,40FFH ; FUSH 'STATE MOV A,418H ; PUSH-1 STATE FUSH RCB ; NAVE RCE INDERCEI I-BUTE INTO ACC. MOV OS SUUST,40FFH ; FUSH 'STATE MOV A,418H ; FUSH-1 STATE FUSH RCB ; NETTI ; IS TT HEL LAST I-BUTE? MOV SIUST,40FFH ; FUSH 'STATE FUSH RCB ; RESTORE THE CONTROL STATE(10H). MOV A,460FFH ; FUSH 'STATE FUSH RCB ; RESTORE THE CONTENTS OF RCB RETURN COV RS,40FFH ; FUSH STATE MOV A,40AFH ; CONTROL STATE FOOP RCB ; RESTORE THE CONTENTS OF RCB RETURN MOV RS,40FFH ; EXT. RAM TRANSMIT BUFFER LENGTH MOV A,40AFH ; CONTROL STATE MOV A,40AFH ; CONTROL STATE MOV SUUST,457FH ; MAIT FOR CTL BATE TANTA TANTA MOV RS,40FFH ; WAIT FOR CTL BATE TANTA TANTA MOV SUUST,457FH ; WAIT FOR CTL BATE TANTE MOV A,40AFH ; CONTROL STATE MOV SUUST,457FH ; MOVE DATA FOR THE CONTENTS OF RCB RETURN KOY DFTR,57H ; NATE TO REAL BATE THE STATE MOV SUUST,457FH ; MOVE DATA FOR THE CONTENT STATE MOV SUUST,4</pre>	<pre>MOV 32H,A ; SAVE NS MOV 4,33H ; MOVE NR INTO ACC. INC A ; INCREMENT NR ANL A,400000111 ; MAKE NOT THE LEAST 3 SIG. BITS MOV 33H,A ; SAVE NR RL A ; SHIFT 4 BITS TO LEFT RL A ; SHIFT 4 BITS TO LEFT RL A ; SHIFT 1 BIT TO LEFT ORL A,400010000B ; STT THE POLL BIT MOV TCB,A ; MOVE ONTROL BYTE INTO TCB ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV STAD,444H MOV 34H,401H : MOV STAD,444H MOV 34H,401H : MOV STAD,444H MOV 34H,401H : MOV STAD,444H MOV 10,40FBH ; INTERRUPT SAVE SETE EA RETI RETI EX MOV TCD,40FBH ; INTERRUPT FRE 8 COUNTS SETE EA RETI WOV TCD,40FBH ; INTERRUPT AFTER 8 COUNTS SETE EA RETI ******** RECEIVE INTERRUPT ROUTINE ************************************</pre>		ANL	A,#00000111B		
<pre>INC A ANL A, #0000011B ; MASK OUT THE LEAST 3 SIG. BITS MOV 338,A ; SAVE NR RL A RL A RL A RL A RL A RL A RL A RL</pre>	INC A ANL A, 400000116 ; MASK OUT THE LEAST 3 SIG. BITS MOV 33H,A, i SANK UT, RE LEAST 3 SIG. BITS MOV 33H,A, ; SHIFT 4 BITS TO LEFT RL A RL A R			32H,A	; SAVE NS	
ANL A,40000111B ; MASK OUT THE LEAST 3 SIG. BITS MOV 338,A ; SAVE NR RL A ; SHIFT 4 BITS TO LEFT RL A RL	ANL A, #00000111B ; MASK OUT THE LEAST 3 SIG. BITS MOV 33H,A ; SAVE NR RL A ; SHIFT 4 BITS TO LEFT RL A RL A RC A					
MOV 33H,A ; SAVE NR RL A ; SHIFT 4 BITS TO LEFT RL A RL A	<pre>MVV 33H,A ; SAVENR RL A ; SHIFT 4 BITS TO LEFT RL A RL A RL A RL A RL A RL A RL A RL A</pre>		ANL	A,#00000111B	; MASK OUT THE LEAST 3 SIG. BITS	
RL A RL A	RL A RL A RL A ORL A, 33H ; MOVE NS COUNT TO ACC. RL A ; SHIFT 1 BIT TO LEFT ORL A, 4000100005 ; SHIFT 1 BIT TO LEFT MOV TCB,A ; MOVE NS COUNTOL BTT MOV TCB,A ; MOVE NS COUNTOL BTT MOV STAD,#44H MOV STS,#11000005 ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIR NOUTINE RETT WOV STS,#11000005 ; ENABLE TRANSMIRSION ACALL TRAN ; CALL TRANSMIR NOUTINE RETT WOV STS,#11000005 ; ENABLE TRANSMIRSION ACALT TRAN ; CALL TRANSMIR NOUTINE RETT WOV STS,#0100005 ; ENABLE TRANSMIRSION ACALT TRAN MOV STS,#11000005 ; ENABLE TRANSMIRSION ACALT TRAN MOV STS,#0100005 ; ENABLE TRANSMIRSION ACALT TRAN MOV STS,#0100005 ; ENABLE TRANSMIRSION ACALT TRAN MOV STS,#0100005 ; ENABLE TRANSMIRSION ACALT TRAN MOV COUPEN ; DISABLE ALL INTERRUPTS MOV TDO, 00FH ; DISABLE ALL INTERRUPTS MOV TDO, 00FH ; INTERRUPT ACUTINE COUNTS SETE TRO SETE TRO NOV DETR,#4001 ; EXT. RAM RECIVE BUFFER START MOV A, #00FH ; PIST. TAM RECIVE BUFFER START MOV A, #10H TI: MOV SIUST,#0FH ; PIST. FONTEOL BYTE TI: MOV SIUST,#0FH ; PUSH-1 STATE MOV A, #10H MOV A, #10H MOV A, RCS ; MOVE MALT TOO CONTROL STATE(10H). MOV A, RCS ; MOVE WIT FOR THE NOT CONTROL STATE NOV A, RCS ; MOVE WIT FOR THE TOO ACC. MOV BOPTR, ; INCEMENT FIT TO EXT. RAM INC DFTR ; INCEMENT FIT TO EXT. RAM NOV A, #0ASH ************************************			33H,A	; SAVE NR	
<pre>RL A ORL A, 31H ; MOVE NS COUNT TO ACC. RL A * SHIFT 1 BIT TO LEFT ORL A, #00010000B ; SET THE POLL BIT MOV TCB,A ; MOVE CONTROL BYTE INTO TCB ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV STAD,#44H MOV STAD,#44H MOV STA,#1110000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT SOUTHE RETT WETURN: CLE EA ; DISABLE ALL INTERRUTS MOV TLO,#0FBH ; INTERRUFT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB TRO ; TURN ON COUNTER 0 SETB EA RETI FOULD ; MOV DFTR, #400H ; EXT. RAM RECEIVE BUFFER START MOV DFTR, #400H ; EXT. RAM RECEIVE BUFFER START MOV DFTR, #400H ; EXT. RAM RECEIVE BUFFER LENGTH MOV DFTR, #400H ; FUSH.1 STATE FUSH. CLE SAVE RECEIVE INTERRUFT NOT THE CONTROL BYTE FUSH. RCB : SAVE RECEIVE CONTROL BYTE FUSH. SETT I SAVE RECEIVE CONTROL STATE MOV A, #14H ; FUSH.1 STATE FUSH. RCB : AN UST, WAIT1 ; WAIT FOR THE CONTROL BYTE FUSH. RCB : AN UST, #0EFH ; FUSH. STATE FUSH. RCB : AN UST, #0EFH ; FUSH. STATE MOV A, #14H ; FUSH.1 STATE FUSH. RCB : AN UST, WAIT1 ; WAIT FOR THE CONTROL STATE (10H). MOV A, #14H ; FUSH.1 STATE FUSH. RCB : AN UST, #0EFH ; FUSH. "BYTE" INTO ACC. MOV A, #16H ; FUSH.1 STATE MOV A, &amp; A HOM ; FUSH.1 STATE MOV A, &amp; A HOM ; FUSH.1 STATE MOV A, &amp; A HOM ; FUSH.1 STATE MOV A, &amp; A, SIUST, WAIT2 ; WAIT FOR AN 1-BYTE MOV A, &amp; A, SIUST, WAIT2 ; WAIT FOR AN 1-BYTE MOV A, &amp; A, SIUST, WAIT2 ; WAIT FOR AN 1-BYTE MOV A, &amp; A, SIUST, WAIT2 ; WAIT FOR AN 1-BYTE MOV A, &amp; A, CDFTR ; INCREMENT PT TO EXTERNAL RAM DJNZ R5, NEXTI ; IS IT THE LAST I-BYTE? MOV A, &amp; A, CDFTR ; WAIT FOR THE CONTENTS OF RCB RETI ; CONTROL STATE MOV A, &amp; A, CDFTR ; MOVE DATA TO EXT. RAM TO ACC. MOV A, &amp; A, OPTR ; MOVE DATA FROM EXT. RAM TO ACC. MOV A, &amp; A, OPTR ; MOVE DATA ROM EXT. RAM AO ACC. MOV A, &amp; A, OPTR ; MOVE DATA ROM EXT. RAM AO ACC. MOV A, &amp; A, OPTR ; MOVE DATA ROM EXT. RAM AO ACC. MOV SIUST, &amp; STATE ; NOVE DATA ROM EXT. RAM AO ACC. MOV A, &amp; A, OPTR ; MOVE DATA ROM EXT. RAM AO ACC. MOV SUST, &amp; STATE ; MAT FOR THE LAST I-BYTE ? MOV SIUST, &amp; STATE ; MAUT FOR THE LAST I-BYTE ? MOV SIUST, &amp; STATE ; NONE MAUT INTO INT. RAM &amp; (TBS) I</pre>	RL       A         ORL       A, 31H       ; MOVE NS COUNT TO ACC.         RL       A       ; SHIFT 1 BIT TO LEFT         ORL       A, 400010000B       ; SET THE FOLL BIT         MOV       TCB, A       ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0         MOV       STAD, 444H       ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0         MOV       STAD, 444H       ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0         MOV       STAD, 444H       ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0         NCV       STAD, 444H       ; CALL TRANSMIT ROUTINE         MOV       STAD, 444H       ; CALL TRANSMIT ROUTINE         RATI       ; CALL TRAN       ; CALL TRANSMIT ROUTINE         RETI       ; CLR FA       ; DISABLE ALL INTERRUPTS         MOV TLO, 40FBH       ; INTERRUPT AFTER 8 COUNTS       SETB EA         SETB       FR       ; TURN ON COUNTER 0         MOV TS, 40FFH       ; EXT. RAM RECEIVE BUFFER START         MOV RS, 40FFH       ; EXT. RAM RECEIVE BUFFER LENGTH         MOV A, 418H       ; PUSH "STATE         TI:       MOV RA, 60FH       ; EXT. RAM RECEIVE BUFFER LENGTH         MOV A, 418H       ; PUSH "STATE         MOV SIUST, 40EFH       ; PUSH "STATE         MOV A, 418H       ; PUSH "STATE <tr< td=""><td></td><td></td><td></td><td>, GALLE 4 BILD TO LEFT</td><td></td></tr<>				, GALLE 4 BILD TO LEFT	
<pre>ORL A, 31R ; MOVE NS COUNT TO ACC. RL A ; SHIFT 1 BIT TO LEFT ORL A, #00010000B ; SET THE POLL BIT MOV TCB,A ; MOVE CONTROL BYTE INTO TCB ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV STAD,#44H MOV 3TA,#1100000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETI WOV TCD,#0FBH ; INTERRUPT AFTER 8 COUNTS SETB TR0 ; TURN ON COUNTER 0 SETB EA ; DISABLE ALL INTERRUPTS MOV DTC,#0FBH ; INTERRUPT AFTER 8 COUNTS SETB TR0 ; TURN ON COUNTER 0 MOV DFTR,#400H ; EXT. RAM RECEIVE BUFFER START MOV DFTR,#400H ; EXT. RAM RECEIVE BUFFER START MOV DFTR,#400H ; EXT. RAM RECEIVE BUFFER LENGTH MOV DFTR,#400H ; EXT. RAM RECEIVE BUFFER LENGTH MOV A, #18H ATT ; FUSH-1 STATE FUSH RCB : SAVE RECEIVE CONTROL BYTE FUSH RCB : ASUUST,WAIT1 ; WAIT FOR THE CONTROL BYTE FUSH RCB : AND STATE MOV A, #18H ATT ; FUSH-1 STATE FUSH RCB : INCERVENT FT R 0 EXT. RAM RECEIVE BUFFER LENGTH MOV A, #18H ATT ; FUSH-1 STATE FUSH RCB : SAVE RECEIVE CONTROL BYTE FUSH RCB : AND STATE : AND RECEIVE BUFFER LENGTH MOV A, #18H ATT ; FUSH-1 STATE FUSH RCB : SAVE RECEIVE CONTROL BYTE FUSH RCB : SAVE RECEIVE CONTROL BYTE FUSH RCB : INCERCHIP FT TO CANTOL STATE(10H). MOV A, #18H ATT ; FUSH-1 STATE MOV A, #18H ATT ; STATE FOR ALL -BYTE MOV A, &amp; SIUST, WAIT ? WAIT FOR ALL -BYTE MITO ACC. MOV A, &amp; SUUST, #0EFH ; FUSH "BYP" INFO CONTROL STATE(10H). MOV A, &amp; ACB ; MOVE DATA TO EXT. RAM INC DPTR ; INCREMENT PTR TO EXTERNAL RAM DJNM R5,NEXTI ; IS IT THE LAST I-BYTE? FOP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN</pre>	ORL       A, 33H       ; MOVE NS COUNT TO ACC.         RL       A       ; SHIFT 1 BIT TO LEFT         ORL       A, #00010000B       ; SET THE POLL BIT         MOV       TCB; A       ; MOVE CONTROL BYTE INTO TCB         MOV       STAD, #44H       ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0         MOV       STAD, #44H       ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0         MOV       STAD, #44H       ; CALL TRANSMIT ROUTINE         MOV       STS,#1100000B       ; ENABLE ALL INTERRUPTS         NCALL TRAN       ; CALL TRANSMIT ROUTINE         RETT       ; TURN ON COUNTER 0         MOV       TLO, #0FH         SETB EA       ; TURN OFF COUNTER 0         MOV DPTR, #400H       ; EXT. RAM RECEIVE BUFFER START         MOV A, #18H       ; PUSH-1 STATE         TL:       CIR A, \$SIUST, #0FH       ; FUSH "BYF" INTO CONTROL STATE(10H).         MOV A, #18H       ; PUSH-1 STATE         TL:       CIR A, \$SIUST, #0FFH       ; FUSH "BYF" INTO CONTROL STATE(10H).         MOV A, #18H       ; PUSH-1 STATE         TL:       CIR A, \$SIUST, #0FFH       ; FUSH "BYF" INTO CONTROL STATE(10H).         MOV A, #18H       ; PUSH-1 STATE         MOV SUST, #0FFH       ; FUSH "BYF" INTO CONTROL STATE(10H).					
RL A' ORL A, #0010000B MOV TCB,A MOV CCB,A TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV STAD,#44H MOV 344,#01H SETI: MOV STS,#11100000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMITSION ACALL TRAN ; CALL TRANSMITSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETI NETURN CLR EA ; DISABLE ALL INTERRUPTS MOV TLO,#0FBH ; INTERRUPT AFTER 8 COUNTS SETE TRO ; TURN ON COUNTER 0 SETE EA RETI RECEIVE INTERRUPT ROUTINE ************************************	<pre>RL A' ; SHIFT 1 BIT TO LEFT GRL A, #001000B ; SET THE POLL BIT MOV TCB,A ; MOVE CONTROL BITE INTO TCB ; MOVE CONTROL BITE INTO TCB ; MOV STAD, #44H MOV 34H, #01H I: MOV STS, #11100000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT SUDTINE RETT UNN: CLR EA ; DISABLE ALL INTERRUFTS MOV TLO, #0FBH ; INTERRUFT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB TRO ; TURN ON COUNTER 0 SETB EA RETTI CLR TRO ; TURN OFF COUNTER 0 MOV DFTR, #400H ; EXT. RAM RECEIVE BUFFER START MOV R5, #0FFH ; EXT. RAM RECEIVE BUFFER START MOV R5, #0FFH ; EXT. RAM RECEIVE BUFFER START MOV SIUST, #0EFH ; EXT. RAM RECEIVE BUFFER LENGTH MOV A, #18H ; PUSH-1 STATE TI: MOV SIUST, #0EFH ; FUSH-1 STATE TI: MOV SIUST, #0EFH ; PUSH-1 STATE TI: MOV SIUST, #0EFH ; PUSH-1 STATE TI: MOV SIUST, #0FFR ; EXT. RAM TRANSHIT BUFFER START MOV &amp; A, FCA MOV &amp; A, PCC ; MOVE DATA TO EXT. RAM MOV &amp; A, FCA MOV &amp; MOPTR, A ; MOVE DATA TO EXT. RAM MOV &amp; A, FCA MOV &amp; MOPTR, A ; MOVE DATA TO EXT. RAM MOV &amp; A, FCA MOV &amp; MOPTR, A ; INCREMENT PIT TO EXTERNAL RAM DVNZ &amp; A, GDATA ; EXT. RAM TRANSMIT BUFFER START MOV A, #0ABH ; CONTROL STATE TI CANE A, SIUST, WAIT ; WAIT FOR CTLE BYTE XMISSION MOVA A, #0ABH ; CONTROL STATE MOVA A, #0ABH ; CONTROL STATE MOVA A, #0ABH ; MOVE DATA FROM EXT. RAM TO ACC. MOV &amp; BI, A ; MOVE DATA FROM EXT. RAM TRANSMIT BUFFER START MOVA A, #0DFTR ; MOVE DATA FROM EXT. RAM 10 ACC. MOVA SIUST, #57NH ; MOVE DATA FROM EXT. RAM 10 ACC. MOVA SIUST, #57NH ; MOVE DATA FROM EXT. RAM 10 ACC. MOVA SIUST, #57NH ; MOVE DATA FROM EXT. RAM 10 ACC. MOVA SIUST, #57NH ; NOVE DATA FROM EXT. RAM 10 ACC. MOVA SIUST, #57NH ; NOVE DATA</pre>				: MOVE NS COUNT TO ACC.	
<pre>MOV TCB,A ; MOVE CONTROL BYTE INTO TCB ; TCB: NR2,NR1,NR0,1,NS0,0 MOV STAD,#44H HOV 344,#0H BETI: MOV STS,#11100000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMISSION ACALL TRAN ; CALL TRANSMISSION ACALL TRAN ; CALL TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETI NETURN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO,#0FBH ; INTERRUPT AFTER 8 COUNTS SETB TR0 ; TURN ON COUNTER 0 SETB EA RETI REC: CLR TR0 ; TURN OFF COUNTER 0 MOV DPTR,#400H ; EXT. RAM RECEIVE BUFFER START MOV AS,#0FFH ; EXT. RAM RECEIVE BUFFER LENGTH MOV AS,#18H ; PUSH-1 STATE PUSH RCB ; SAVE RECEIVE CONTROL BYTE PUSH RCB ; SAVE RECEIVE CONTROL BYTE NAIT1: CLN A,SILST,WAIT1 ; WAIT FOR THE CONTROL STATE(10H). MOV A,#18H ; PUSH-1 STATE MOV A, ARCB ; MOVE RECEIVED I-BYTE INTO ACC. MOV &amp; A,RCB ; MOVE RECEIVED I-BYTE NIC DPTR,A ; NOVE RECEIVED I-BYTE MOV A, RCB ; MOVE RECEIVED I-BYTE MOV A, RCB ; MOVE RECEIVED I-BYTE MOV A, ARCB ; MOVE RECEIVED I-BYTE MOV A, BOFTR,A ; INCREMENT FIR TO EXTERNAL RAM DINL RS,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN</pre>	<pre>MOV TCB,A ; MOVE CONTROL BYTE INTO TCB ; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV STD0,#44H MOV 34H,#01H I: MOV STS,#11100000B ; ENABLE TRANSMISSION ACALL TERAN ; CALL TRANSMISSION ACALL TERAN ; CALL TRANSMISSION MOV TL0,#0FBH ; DISABLE ALL INTERRUPTS METT SETE EA ************ RECEIVE INTERRUPT AFTER 8 COUNTS SETE EA **********************************</pre>		RL	A	; SHIFT 1 BIT TO LEFT	
; TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV STAD,444H MOV STA,411 MOV STA,411 MOV STS,411100000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETI RETURN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO,40FBH ; INTERRUPT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB EA RETI RECEIVE INTERRUPT ROUTINE ************************************	<pre>// TCB: NR2,NR1,NR0,1,NS2,NS1,NS0,0 MOV STAD,#44H MOV 34H,#01H // MOV STS,#11100000B / ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETI URN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO,#0FDH ; INTERRUPT AFTER 8 COUNTS SETB TRA ; 292019-35 *********** RECEIVE INTERRUPT ROUTINE ************************************</pre>					
<pre>MOV STAD,#44H MOV 34H,#01H EFTI: MOV STS,#11100000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETT RETT WOV TLO,#0FBH ; DISABLE ALL INTERRUPTS MOV TLO,#0FBH ; INTERRUPT AFTER 8 COUNTS SETB TR0 ; TURN ON COUNTER 0 SETB EA RETI 2000 SETB EA RETI 2000 MOV DPTR,#400H ; EXT. RAM RECEIVE BUFFER START MOV DPTR,#400H ; EXT. RAM RECEIVE BUFFER START MOV DPTR,#400H ; EXT. RAM RECEIVE BUFFER LENGTH MOV AS,#0FFH ; EXT. RAM RECEIVE BUFFER LENGTH MOV A,#18H ; PUSH-1 STATE PUSH RCB ; SAVE RECEIVE CONTROL BYTE FUSH RCB ; SAVE RECEIVE CONTROL BYTE HEXTI: MOV SIUST,#0EFH ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; FUSH-1 STATE MAIT2: CONE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE MAIT2: CONE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE MOVE APAT TO EXTERNAL RAM DINZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; VES, RESTORE THE CONTENTS OF RCB RETI ; RETURN TANSWITT TANSMIT SUBROUTINE ************************************</pre>	<pre>MoV STAD_#44H MoV 34H_#0H MoV 34H_#0H I: MOV STS_#11100000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETI NOV TLO_#0FEH ; DISABLE ALL INTERRUPTS MOV TLO_#0FEH ; INTERRUPT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB EA RETI 2292019-35 ************ RECEIVE INTERRUPT ROUTINE ************************************</pre>		MOV	ICD, R		
<pre>EETI: MOV STS, #11100000B ; ENABLE TRANSMISSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETT WETURN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO, #OFBH ; INTERRUPT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB EA RETI 20019-35 ***********************************</pre>	<pre>I: MOV STS_#11100000B ; ENABLE TRANSMITSION ACALL TRAN ; CALL TRANSMIT ROUTINE RETI URN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO,#OFBH ; INTERRUPT AFTER 8 COUNTS STB TRO ; TURN ON COUNTER 0 STB EA RETI 22019-35 ************************************</pre>					
ACALL TRAN ; CALL TRANSMIT ROUTINE RETI RETURN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO, #0FBH ; INTERRUPT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB TRA 292019-35 ;************************************	ACALL TRAN ; CALL TRANSMIT ROUTINE RETI RETI URN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO,40FBH ; INTERRUPT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB EA RETI CLR TRO ; TURN OF COUNTER 0 MOV DPTR,4400H ; EXT. RAM RECEIVE BUFFER START MOV R5,40FFH ; EXT. RAM RECEIVE BUFFER START MOV SLUST,WAIT1 ; WAIT FOR THE CONTROL BYTE T1: CJNE A, SLUST,WAIT1 ; WAIT FOR THE CONTROL BYTE T2: CJNE A, SLUST,WAIT2 ; WAIT FOR AN I-BYTE MOV A, 418H ; PUSH-1 STATE T2: CJNE A, SLUST,WAIT2 ; WAIT FOR AN I-BYTE MOV A, RCB ; MOVE RECEIVE INTO ACC. MOX & QEDTR,A ; NOVE DELTA TO EXT. RAM INC DFTR ; INCREMENT FUR TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ************************************	GETI:			: ENABLE TRANSMISSION	
<pre>SETURN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO, #OFBH ; INTERRUPT AFTER 8 COUNTS SETE TRO ; TURN ON COUNTER 0 SETE EA RETI 2292019-35 ;************* RECEIVE INTERRUPT ROUTINE ************************************</pre>	<pre>URN: CLR EA ; DISABLE ALL INTERRUPTS MOV TLO, #OFBH ; INTERRUPT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB EA RETI 2202019-35 ************************************</pre>			TRAN		
MOV TLO, #OFBH ; INTERRUPT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB EA RETI 222019-35 ;************** RECEIVE INTERRUPT ROUTINE ************************************	MOV TLO, #0FBH ; INTERRUPT AFTER 8 COUNTS SETB TRO ; TURN ON COUNTER 0 SETB EA RETI 2002019-35 ************************************	RETURN:		EA	DISABLE ALL INTERDUPTS	
SETS EA RETI 292019-35 ************************************	<pre>SETE EA RETI 202019-35 ************************************</pre>		MOV	TLO, #OFBH	; INTERRUPT AFTER 8 COUNTS	
RETI 292019-35 ;************************************	RETI292019-35***********************************				; TURN ON COUNTER 0	
<pre>XEC: CLR TR0 ; TURN OFF COUNTER 0 MOV DPTR,#400H ; EXT. RAM RECEIVE BUFFER START MOV R5,#0FFH ; EXT. RAM RECEIVE BUFFER START MOV A,#18H ; PUSH-1 STATE FUSH RCB ; SAVE RECEIVE CONTROL BYTE FUSH RCB ; SAVE RECEIVE CONTROL BYTE FUSH RCB ; SAVE RECEIVE CONTROL BYTE FUSH RCB ; SAVE RECEIVE CONTROL STATE(10H). MOV A,#18H ; PUSH-1 STATE MOV A,#18H ; PUSH-1 STATE MOV A,#18H ; PUSH-1 STATE MOV A,#18H ; PUSH-1 STATE MOV A,RCB ; MOVE DATA FOR AN I-BYTE MOV A,RCB ; MOVE DATA TO EXT. RAM INC DPTR, A ; MOVE DATA TO EXT. RAM INC DPTR ; INCREMENT FIR TO EXTERNAL RAM INC DPTR ; INCREMENT FIR TO EXTERNAL RAM INC DPTR ; INCREMENT FIR TO EXTERNAL RAM ON R5,#0FFH ; EXT. RAM TRANSMIT BUFFER START MOV R5,#0FFH ; EXT. RAM TRANSMIT BUFFER START MOV A,#0A8H ; CONTROL STATE VAIT: CJNE A,SIUST,WAIT ; WAIT FOR CTL BYTE XMISSION MOV A,#0A8H ; CONTROL STATE VAIT: CJNE A,SIUST,WAIT ; MOVE DATA FOR MEXT. RAM TA MA CC. MOV @R1,A ; MOVE DATA INTO INT. RAM @ (TBS) INC DPTR ; INCREMENT PITE ? MOV SIUST,#57H ; KEEP "BUFT INCONTROL STATE(A6H). MOV A,#0BH ; CONTROL STATE XET ; RETURN.</pre>	<ul> <li>CLR TRO ; TURN OFF COUNTER 0 MOV DFTR,#400H ; EXT. RAM RECEIVE BUFFER START MOV R5,#0FFH ; EXT. RAM RECEIVE BUFFER LENOTH MOV A,#18H ; FUSH-1 STATE</li> <li>TI: CJNE A,SIUST,WAIT1 ; WAIT FOR THE CONTROL BYTE FUSH RCB ; SAVE RECEIVE CONTROL BYTE</li> <li>TI: MOV SIUST,#0EFH ; FUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; FUSH-1 STATE</li> <li>T2: CJNE A,SIUST,WAIT2 ; WAIT FOR AN 1-BYTE MOV A,RCB ; MOVE RECEIVE D.TENTE INTO ACC. MOVW &amp; @DPTRA ; MOVE DATA TO EXTERNAL RAM INC DFTR ; INCREMENT FTR TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN</li> <li>************************************</li></ul>			LA		292019-35
<ul> <li>MOV DPTR,#400H</li> <li>EXT. RAM RECEIVE BUFFER SLART</li> <li>MOV A,#0FFH</li> <li>FXT. RAM RECEIVE BUFFER LENGTH</li> <li>MOV A,#18H</li> <li>FUSH-1 STATE</li> <li>VAITI: CONE A,SIUST,WAITI</li> <li>WAIT FOR THE CONTROL BYTE</li> <li>FUSH RCB</li> <li>SAVE RECEIVE CONTROL BYTE</li> <li>FUSH-1 STATE</li> <li>VAIT2: CUNE A,SIUST,WAIT2</li> <li>WAIT FOR AN I-BYTE INTO CONTROL STATE(10H).</li> <li>MOV A,#18H</li> <li>FUSH-1 STATE</li> <li>VAIT2: CUNE A,SIUST,WAIT2</li> <li>WAIT FOR AN I-BYTE</li> <li>MOV A,RCB</li> <li>MOV BDTR, A</li> <li>MOV BTR, A</li> <li>MOV BTTR, A</li> <li>MOV BTTR,</li></ul>	<ul> <li>MOV DPTR,#400H ; EXT. RAM RECEIVE BUFFER LENGTH MOV R5,#0FFH ; EXT. RAM RECEIVE BUFFER LENGTH MOV A,#18H ; PUSH-1 STATE</li> <li>TI: CJNE A,SIUST,WAIT1 ; WAIT FOR THE CONTROL BYTE FUSH RCB ; SAVE RECEIVE CONTROL BYTE</li> <li>TI: MOV SIUST,#0EFH ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; PUSH-1 STATE</li> <li>T2: CJNE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE MOV A,RCB ; MOVE RECEIVE OTTROL BYTE INTO ACC. MOVX &amp; @DPTRA ; MOVE DATA TO EXTERNAL RAM INC DFTR ; INCREMENT FTR TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE?</li> <li>POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN</li> <li>************************************</li></ul>	;*****	*****	** RECEIVE INTE	RRUPT ROUTINE ****************	
<pre>MOV DPTR,#400H ; EXT. RAM RECEIVE BUFFER LENGTH MOV R5,#0FFH ; EXT. RAM RECEIVE BUFFER LENGTH MOV A,#18H ; PUSH-1 STATE PUSH RCB ; SAVE RECEIVE CONTROL BYTE PUSH RCB ; SAVE RECEIVE CONTROL BYTE MEXTI: MOV SIUST,#0FFH ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; PUSH-1 STATE MOV A,#18H ; PUSH-1 STATE MOV A,RCB ; MOVE DI-BYTE INTO ACC. MOV A,RCB ; MOVE DATA TO EXT. RAM INC DFTR, ; INCREMENT PTR TO EXTERNAL RAM INC DFTR ; INCREMENT PTR TO EXTERNAL RAM INC DFTR ; INCREMENT PTR TO EXTERNAL RAM INC DFTR ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN TRAN: MOV DFTR,#200H ; EXT. RAM TRANSMIT BUFFER START MOV A, #0A8H ; CONTROL STATE MOV A, #0PTR ; MOVE DATA FROM EXT. RAM TANSMIT BUFFER START MOV A, #0A8H ; CONTROL STATE MOV A, #0PTR ; MOVE DATA FROM EXT. RAM TANSISION MOV A, #0PTR ; MOVE DATA FROM EXT. RAM TANSISION MOV A, #0PTR ; MOVE DATA FROM EXT. RAM TQUESSION MOV A, #0PTR ; MOVE DATA INTO INT. RAM @ (TBS) INC DPTR ; INCREMENT PONTER MOV SIST,#57H ; NO. MIT THE LAST I-BYTE ? MOV SIUST,#57H ; KEEP "BUF" IN CONTROL STATE(A6H). MOV A, #0PH ; DINT. ACOP STATE</pre>	<ul> <li>MOV DPTR,#400H ; EXT. RAM RECEIVE BUFFER START</li> <li>MOV R5,#0FFH ; EXT. RAM RECEIVE BUFFER LENGTH</li> <li>MOV A,#18H ; PUSH-1 STATE</li> <li>T1: CJNE A,SIUST,WAIT1 ; WAIT FOR THE CONTROL BYTE</li> <li>FUSH RCB ; SAVE RECEIVE CONTROL BYTE</li> <li>FUSH RCB ; PUSH "BYP" INTO CONTROL STATE(10H).</li> <li>MOV A,#18H ; PUSH-1 STATE</li> <li>T2: CJNE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE</li> <li>MOV A,#18H ; PUSH-1 STATE</li> <li>T2: CJNE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE</li> <li>MOV A, #18H ; PUSH-1 STATE</li> <li>T2: CJNE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE</li> <li>MOV A, RCB ; MOVE DATA TO EXT. RAM</li> <li>INC DPTR ; INCREMENT FTR TO EXTERNAL RAM</li> <li>DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE?</li> <li>POP RCB ; YES, RESTORE THE CONTENTS OF RCB</li> <li>RETI ; RETURN</li> <li>************************************</li></ul>	REC:			; TURN OFF COUNTER 0	
<pre>MOV A, #18H ; PUSH-1 STATE WAIT1: CJNE A,SIUST,WAIT1 ; WAIT FOR THE CONTROL BYTE PUSH RCB ; SAVE RECEIVE CONTROL BYTE WEXTI: MOV SIUST,#0EFH ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; PUSH-1 STATE WAIT2: CJNE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE MOV A,ROB ; MOVE RECEIVED I-BYTE INTO ACC. MOVX @DPTR,A ; MOVE DATA TO EXT. RAM INC DPTR ; INCREMENT PTR TO EXTERNAL RAM DINC R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN TRAN: MOV DPTR,#200H ; EXT. RAM TRANSMIT BUFFER START MOV A, #0A8H ; CONTROL STATE WAIT: CJNE A,SIUST,WAIT ; WAIT FOR CTRL BYTE XMISSION MOVX A, @DPTR ; MOVE DATA FOM EXT. RAM TO ACC. MOV A, #0A8H ; CONTROL STATE WAIT: CJNE A,SIUST,WAIT ; WAIT FOR CTRL BYTE XMISSION MOV A, @OPTR ; INCREMENT POM EXT. RAM TO ACC. MOV @R1,A ; MOVE DATA FOM EXT. RAM (TBS) INC DPTR ; INCREMENT POM EXT. RAM (TBS) NOVE A,#0A8H ; CONTROL STATE WAIT: CJNE A,SIUST,WAIT ; WAIT FOR CTRL BYTE XMISSION MOV A, @OPTR ; INCREMENT POM EXT. RAM @ (TBS) INC DPTR ; INCREMENT POM TEXT. MOV SIUST,#57H ; NOX MAIT THE LAST I-BYTE ? MOV SIUST,#57H ; NOX MAIT THE LAST I-BYTE RET ; RETURN. XXTI: MOV SIUST,#57H ; KEEP "BUP" IN CONTROL STATE (A6H). MOV A, @OPH * DATA FOM EXT. RAM EACH. MOV SIUST,#57H ; KEEP "BUP" IN CONTROL STATE (A6H). MOV A, #00H ; DNA-LOOP STATE</pre>	<ul> <li>MOV A, #IBH ; PUSH-1 STATE</li> <li>CINE A, SIUST, WAIT1 ; WAIT FOR THE CONTROL BYTE PUSH RCB ; SAVE RECEIVE CONTROL BYTE</li> <li>TI: MOV SIUST, #OEFH ; PUSH "BYP" INTO CONTROL STATE(10H).</li> <li>MOV A, #18H ; PUSH-1 STATE</li> <li>T2: CJNE A, SIUST, WAIT2 ; WAIT FOR AN I-BYTE MOV A, #18H ; PUSH-1 STATE</li> <li>T2: CJNE A, SIUST, WAIT2 ; WAIT FOR AN I-BYTE</li> <li>MOV A, C DPTR ; MOVE DATA TO EXT. RAM</li> <li>INC DPTR ; INCREMENT PTR TO EXTERNAL RAM</li> <li>DJNZ R5, NEXTI ; IS IT THE LAST I-BYTE ?</li> <li>POP RCB ; YES, RESTORE THE CONTENTS OF RCB</li> <li>RETI ; RETURN</li> <li>************************************</li></ul>				; EXT. RAM RECEIVE BUFFER START	
<pre>AAIT1: CJME A,SIUST,WAIT1 ; WAIT FOR THE CONTROL BYTE PUSH RCB ; SAVE RECEIVE CONTROL BYTE NEXTI: MOV SIUST,#OEFH ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,RCB ; MOVE RECEIVED I-BYTE INTO ACC. MOV &amp; OPTR,A ; MOVE DATA TO EXT.RAM INC DFTR ; INCREMENT PTR TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ************************************</pre>	<pre>T1: CJNE A,SIUST,WAIT1 ; WAIT FOR THE CONTROL BYTE PUSH RCB : SAVE RECEIVE CONTROL BYTE T1: MOV A,#18H ; PUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; PUSH-1 STATE T2: CJNE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE MOV A,RCB ; MOVE RECEIVED I-BYTE INTO ACC. MOVY @DPTR,A ; MOVE RECEIVED I-BYTE INTO ACC. MOV @DPTR,A ; MOVE DATA TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ************************************</pre>			A,#18H	; DAT. RAM RECEIVE BUFFER LENGTH ; PUSH-1 STATE	
<pre>NEXTI: MOV SIUST,#0EFH ; FUSH "BYP" INTO CONTROL STATE(10H). MOV A,#18H ; FUSH "BYP" INTO CONTROL STATE(10H). MAIT2: CINE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE MOV A,RCB ; MOVE RECEIVED I-BYTE INTO ACC. MOV &amp; @DFTR,A ; MOVE DATA TO EXT. RAM INC DFTR ; INCREMENT FUR TO EXTERNAL RAM DINZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ***********************************</pre>	<pre>TI: MOV SIUST,#OEFH ; PUSH "BYD" INTO CONTROL STATE(10H). MOV A,#18H ; PUSH-1 STATE T2: CJNE A,SIUST,WAIT2 ; WAIT FOR AN I-BYTE MOV A,RCB ; MOVE RECEIVED I-BYTE INTO ACC. MOVW @DPTRA ; MOVE RECEIVED I-BYTE INTO ACC. MOVW @DPTRA ; INCREMENT PTR TO EXTERNAL RAM INC DPTR ; INCREMENT PTR TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ************************************</pre>	WAIT1:	CJNE	A,SIUST,WAIT1	; WAIT FOR THE CONTROL BYTE	
<pre>Movx @DTR,A ; Move DATA TO EXT. RAM INC DPTR ; INCREMENT PTR TO EXT. RAM INC DPTR ; INCREMENT PTR TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ;************************************</pre>	<ul> <li>MOV ROUTE A ; MOVE DATA TO EXT. RAM</li> <li>MOV ROUTE DATA TO EXT. RAM</li> <li>INC DPTR ; INCREMENT PTR TO EXTERNAL RAM</li> <li>DJNZ R5, NEXTI ; IS IT THE LAST I-BYTE?</li> <li>POP RCB ; YES, RESTORE THE CONTENTS OF RCB</li> <li>RETI ; RETURN</li> <li>************************************</li></ul>	VEXTT.		RCB STUST #AFFH	; SAVE RECEIVE CONTROL BYTE ; PUSH "BYP" INTO CONTROL STATE (1011)	
<pre>Movx @DTR,A ; Move DATA TO EXT. RAM INC DPTR ; INCREMENT PTR TO EXT. RAM INC DPTR ; INCREMENT PTR TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ;************************************</pre>	<ul> <li>MOV ROUTE A ; MOVE DATA TO EXT. RAM</li> <li>MOV ROUTE DATA TO EXT. RAM</li> <li>INC DPTR ; INCREMENT PTR TO EXTERNAL RAM</li> <li>DJNZ R5, NEXTI ; IS IT THE LAST I-BYTE?</li> <li>POP RCB ; YES, RESTORE THE CONTENTS OF RCB</li> <li>RETI ; RETURN</li> <li>************************************</li></ul>		MOV	A,#18H	; PUSH-1 STATE	
<pre>Movx @DTR,A ; Move DATA TO EXT. RAM INC DPTR ; INCREMENT PTR TO EXT. RAM INC DPTR ; INCREMENT PTR TO EXTERNAL RAM DJNZ R5,NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ;************************************</pre>	<ul> <li>MOV ROUTE A ; MOVE DATA TO EXT. RAM</li> <li>MOV ROUTE DATA TO EXT. RAM</li> <li>INC DPTR ; INCREMENT PTR TO EXTERNAL RAM</li> <li>DJNZ R5, NEXTI ; IS IT THE LAST I-BYTE?</li> <li>POP RCB ; YES, RESTORE THE CONTENTS OF RCB</li> <li>RETI ; RETURN</li> <li>************************************</li></ul>	WAIT2:	CJNE	A,SIUST,WAIT2	; WAIT FOR AN I-BYTE	
DVAZ K5,NEXTI ; IS IT THE LAST I-FYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ************************************	DUNZ K5, NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ************************************			@DPTR,A	; MOVE RECEIVED 1-BYTE INTO ACC. ; MOVE DATA TO EXT. RAM	
DVAZ K5,NEXTI ; IS IT THE LAST I-FYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ************************************	DUNZ K5, NEXTI ; IS IT THE LAST I-BYTE? POP RCB ; YES, RESTORE THE CONTENTS OF RCB RETI ; RETURN ************************************		INC	DPTR	; INCREMENT PTR TO EXTERNAL RAM	
RETI ; RETURN ;************************************	RETI     ; RETURN       ************************************			R5,NEXT1	; IS IT THE LAST I-BYTE?	
<pre>TRAN: MOV DFTR,#200H ; EXT. RAM TRANSMIT BUFFER START MOV R5,#0FFH ; EXT. RAM TRANSMIT BUFFER LENGTH MOV A,#0A8H ; CONTROL STATE WAIT: CJNE A,SLUST,WAIT ; WAIT FOR CTRL BYTE XMISSION MOVX A, 0DFTR ; MOVE DATA FROM EXT. RAM TO ACC. MOV 0R1,A ; MOVE DATA INTO INT. RAM 0 (TBS) INC DFTR ; INCREMENT FOINTER DJNZ R5,NXTI ; IS IT THE LAST I-BYTE ? MOV SIUST,#57H ; NOX XMIT THE LAST I-BYTE RET ; RETURN. NXTI: MOV SIUST,#57H ; EXEP "BYP" IN CONTROL STATE(A8H). MOV A,#00H ; DMA-LOOP STATE</pre>	<ul> <li>N: MOV DPTR, #200H ; EXT. RAM TRANSMIT BUFFER START MOV R5, #0FFH ; EXT. RAM TRANSMIT BUFFER LENGTH MOV A, #0A8H ; CONTROL STATE</li> <li>T. CJNE A, SIUST, WAIT ; WAIT FOR CTRL BYTE XMISSION MOVX A, @DPTR ; MOVE DATA FROM EXT. RAM TO ACC. MOV @R1,A ; MOVE DATA INTO INT. RAM @ (TBS) INC DPTR ; INCREMENT POINTER DJNZ R5, NYTI ; IS IT THE LAST I-BYTE ? MOV SIUST, #57H ; NO. XMIT THE LAST I-BYTE RET ; RETURN.</li> <li>INC SIUST, #57H ; KEEP "BYP" IN CONTROL STATE(A8H). MOV A, #0B0H ; DMA-LOOP STATE JMP WAIT ; TRANSMIT THE NEXT BYTE</li> </ul>					
MOV       R5,#0FFH       ; EXT. RAM TRANSMIT BUFFER LENGTH         MOV       A,#0A8H       ; CONTROL STATE         MAIT:       CJNE A,SIUST,WAIT       ; WAIT FOR CTRL BYTE XMISSION         MOV       A,#0PTR       ; WOVE DATA FROM EXT. RAM TO ACC.         MOV       QR1,A       ; MOVE DATA INTO INT. RAM @ (TBS)         INC       DPTR       ; INCREMENT POINTER         DJNZ       R5,NXTI       ; IS IT THE LAST I-BYTE ?         MOV       SIUST,#57H       ; RETURN.         XXII:       MOV       SIUST,#57H         MOV       SIUST,#57H       ; DMALOOF STATE	MOV       R5, #0FFH       ; EXT. RAM TRANSMIT BUFFER LENGTH         MOV       A, #0A8H       ; CONTROL STATE         T:       CJNE A, SIUST, WAIT       ; WAIT FOR CTRL BYTE XMISSION         MOVX       A, @DPTR       ; WOVE DATA FROM EXT. RAM TO ACC.         MOV       GRI, A       ; MOVE DATA INTO INT. RAM @ (TBS)         INC       DPTR       ; INCREMENT POINTER         DJNZ       R5,NXTI       ; IS IT THE LAST I-BYTE ?         MOV       SIUST, #57H       ; NO. XMIT THE LAST I-BYTE         RET       ; RETURN.         I:       MOV       SIUST, #57H         MOV       A, #080H       ; DMA-LOOP STATE         JMP       WAIT       ; TRANSMIT THE NEXT BYTE	*****	*****	***** TRANSMIT	SUBROUTINE ************************************	
MOV       R5,#0FFH       ; EXT. RAM TRANSMIT BUFFER LENGTH         MOV       A,#0A8H       ; CONTROL STATE         VAIT:       CJNE A,SUUST,WAIT       ; WAIT FOR CTRL BYTE XMISSION         MOV       A,0PDTR       ; MOVE DATA FROM EXT. RAM TO ACC.         MOV       QR1,A       ; MOVE DATA INTO INT. RAM @ (TBS)         INC       DPTR       ; INCREMENT POINTER         DJNZ       R5,NXTI       ; IS IT THE LAST I-BYTE ?         MOV       SIUST,#57H       ; RETURN.         XXTI:       MOV       SIUST,#57H         MOV       SIUST,#57H       ; DMALOOF STATE	MOV       R5, #0FFH       ; EXT. RAM TRANSMIT BUFFER LENGTH         MOV       A, #0A8H       ; CONTROL STATE         T:       CJNE A, SIUST, WAIT       ; WAIT FOR CTRL BYTE XMISSION         MOVX       A, @DPTR       ; WOVE DATA FROM EXT. RAM TO ACC.         MOV       GRI, A       ; MOVE DATA INTO INT. RAM @ (TBS)         INC       DPTR       ; INCREMENT POINTER         DJNZ       R5,NXTI       ; IS IT THE LAST I-BYTE ?         MOV       SIUST, #57H       ; NO. XMIT THE LAST I-BYTE         RET       ; RETURN.         I:       MOV       SIUST, #57H         MOV       A, #080H       ; DMA-LOOP STATE         JMP       WAIT       ; TRANSMIT THE NEXT BYTE	TRAN:	MOV	DPTR.#200H	; EXT. RAM TRANSMIT BUFFER START	
<ul> <li>VAIT: CJNE A,SIUST,WAIT ; WAIT FOR CTRL BYTE XMISSION MOVX A,@DPTR ; MOVE DATA FROM EXT. RAM TO ACC. MOV @R1,A ; MOVE DATA INTO INT. RAM @ (TBS) INC DPTR ; INCREMENT POINTER DJNZ R5,NXTI ; IS IT THE LAST I-BYTE ? MOV SIUST,#57H ; NO.XMIT THE LAST I-BYTE RET ; RETURN.</li> <li>XXTI: MOV SIUST,#57H ; KEEP "BYP" IN CONTROL STATE (A8H). MOV A,#050H ; DMA-LOOP STATE</li> </ul>	<pre>T: CINE A,SIUST,WAIT ; WAIT FOR CTRL BYTE XMISSION MOVX A,@DFTR ; MOVE DATA FROM EXT. RAM TO ACC. MOV @R1,A ; MOVE DATA INTO INT. RAM @ (TBS) INC DFTR ; INCREMENT POINTER DJNZ R5,NXTI ; IS IT THE LAST I-BYTE ? MOV SIUST,#57H ; NO. XMIT THE LAST I-BYTE ? RET ; RETURN. I: MOV SIUST,#57H ; KEEP "BYP" IN CONTROL STATE(A8H). MOV A,#080H ; DMA-LOOP STATE JMP WAIT ; TRANSMIT THE NEXT BYTE</pre>		MOV	R5,#OFFH	; EXT. RAM TRANSMIT BUFFER LENGTH	
MOVX       A, @DPTR'       ; MOVE DATA FROM EXT. RAM TO ACC.         MOV       @R1,A       ; MOVE DATA INTO INT. RAM @ (TBS)         INC       DPTR       ; INCREMENT FOINTER         DJNZ       R5,NXTI       ; IS IT THE LAST I-BYTE ?         MOV       SIUST, #57H       ; NO.X MIT THE LAST I-BYTE         RET       ; RETURN.         VXTI:       ; NO.X MIT THE LAST I-BYTE         MOV       SIUST, #57H       ; RETURN.         VXTI:       MOV       ; DATA LOOP STATE	MOVX       A (DPTR)       MOVE DATA FROM EXT. RAM TO ACC.         MOV       (R1, A)       ; MOVE DATA INTO INT. RAM (CTBS)         INC       DPTR       ; INCREMENT POINTER         DJNZ       R5,NXTI       ; IS IT THE LAST I-BYTE ?         MOV       SIUST, #57H       ; NO. XMIT THE LAST I-BYTE         RET       ; RETURN.         I:       MOV       SIUST, #57H         MOV       SIUST, #57H       ; KEEP "BYP" IN CONTROL STATE (A8H).         MOV       A, #060H       ; DMA-LOOP STATE         JMP       WAIT       ; TRANSMIT THE NEXT BYTE					
MOV     @Rl,A     ; MOVE DATA INTO INT. RAM @ (TBS)       INC     DPTR     ; INCREMENT POINTER       DJNZ     R5,NXTI     ; IS IT THE LAST I-BYTE ?       MOV     SIUST, #57H     ; NO. XMIT THE LAST I-BYTE       RT     ; RETURN.       XXTI:     MOV SIUST, #57H     ; KEEP "BYP" IN CONTROL STATE(A8H).       MOV     A, #060H     ; DMA-LOOP STATE	MOV       @R1,A       ; MOVE DATA INTO INT. RAM @ (TBS)         INC OPTR       ; INCREMENT POINTER         DINZ       R5,NXTI       ; IS IT THE LAST I-BYTE ?         MOV       SIUST,#57H       ; NO. XMIT THE LAST I-BYTE ?         RET       ; RETURN.         IMOV       SIUST,#57H       ; NO. XMIT THE LAST I-BYTE ?         MOV       SIUST,#57H       ; RETURN.         I       MOV       SIUST,#57H       ; KEEP "BYP" IN CONTROL STATE(A8H).         MOV       A,#080H       ; DMA-LOOP STATE         JMP       WAIT       ; TRANSMIT THE NEXT BYTE		MOVX		; MOVE DATA FROM EXT. RAM TO ACC.	
DJNZ R5,NXTI ; IS IT THE LAST I-BYTE ? MOV SIUST,#57H ; NO. XMIT THE LAST I-BYTE RET ; RETURN. XXTI: MOV SIUST,#57H ; KEEP "BYP" IN CONTROL STATE(A6H). MOV A,#00H ; DMA-LOOP STATE	DJNZ R5,NXTI ; IS IT THE LAST I-BYTE ? MOV SIUST,#57H ; NO. XMIT THE LAST I-BYTE ? RET ; RETURN. I: MOV SIUST,#57H ; KEEP "BYP" IN CONTROL STATE(A8H). MOV A,#080H ; DMA-LOOP STATE JMP WAIT ; TRANSMIT THE NEXT BYTE		MOV	ØRL,A	; MOVE DATA INTO INT. RAM @ (TBS)	
MOV SIUST,#57H ; NO. XMIT THE LAST I-BYTE RET ; RETURN. IXTI: MOV SIUST,#57H ; KEEP "BYP" IN CONTROL STATE(A8H). MOV A,#0B0H ; DMA-LOOP STATE	MOV       SIUST,#57H       ; NO. XMIT THE LAST I-BYTE         RET       ; RETURN.         I:       MOV       SIUST,#57H       ; KEEP "BYP" IN CONTROL STATE(A8H).         MOV       A,#0B0H       ; DMA-LOOP STATE         JMP       WAIT       ; TRANSMIT THE NEXT BYTE			DFTR R5.NXTI	; INCREMENT POINTER : IS IT THE LAST I-BYTE ?	
<pre>XXI: MOV SIUST,#57H ; KEEP "BYP" IN CONTROL STATE(A8H). MOV A,#0B0H ; DMA-LOOP STATE</pre>	I: MOV SIUST,#57H ; KEEP "BYP" IN CONTROL STATE(A8H). MOV A,#0B0H ; DMA-LOOP STATE JMP WAIT ; TRANSMIT THE NEXT BYTE		MOV	SIUST,#57H	; NO. XMIT THE LAST I-BYTE	
MOV A, #0B0H ; DMA-LOOP STATE	MOV A,#0B0H ; DMA-LOOP STATE JMP WAIT ; TRANSMIT THE NEXT BYTE			GTUCA #F71	; RETURN.	
	JMP WAIT ; TRANSMIT THE NEXT BYTE			A,#0B0H	; NELF "BIP" IN CONTROL STATE(A8H). ; DMA-LOOP STATE	
JMP WAIT ; TRANSMIT THE NEXT BYTE END	292019–36					

B-2

		44.PDF)		
; ASSEI ; AUTO	MBLY CO MODE;	DE FOR SECONDARY FCS OPTION	STATIONS (MULTIPOINT)	
	ORG SJMP	OOH INIT		
	ORG	OBH	; VECTOR ADDRESS FOR TIMERO INT.	
	JMP	REC		
	ORG JMP	13H XINT1	; VECTOR ADDRESS FOR EXT. INT. 1	
	ORG JMP	23H SIINT	; VECTOR ADDRESS FOR SIU INTERRUPT	
			ATION *****	
;			ATION STATES	
INIT:	ORG MOV	26H SMD.#11010100B	; INT. CLKED @ 375K;NRZI=1;PFS=1	
	MOV	STAD, #55H	; STATION ADDRESS; STAD=44H FOR THE	
			; OTHER STATION	
	MOV	RBS,#10H RBL,#00H	; INT. RAM RECEIVE BUFFER START ; INT. RAM RECEIVE BUFFER LENGTH	
	MOV	R1,#20H		
	MOV MOV	TBS,R1	; INT. RAM XMIT BUFFER START ; INT. RAM XMIT BUFFER LENGTH	
	MOV	TBL,#01H NSNR,#00H	; INT. RAM XMIT BOFFER LENGTH ; NS=NR=0	
	MOV	TCON. #00000100B	* EXT. INT.: EDGE TRIGGERED	
	MOV	IE,#00010110B	; SI=1; ET0=1; EX0=1	
	MOV MOV	IP,#00000010B	; TIMER 0: PRIORITY 1 ; COUNTER FUNCTION: MODE 3	
	MOV		; RECEIVE I-FRAME.	
	MOV	TLO, #OF8H	; SET COUNTER TO OVERFLOW	
		TRO	; AFTER 8 COUNTS	
	SETB SETB	EA	; TURN ON COUNTER ; ENABLE ALL INTERRUPTS	
DOT:	SIMP	DOT	• WATT FOR AN INTERDIDT	292019-37
; CPU	IS INTE	RRUPTED AT THE	END OF RECEPTION (SI SET), AND AT*	232013-37
	END OF	LONG-FRAME TRANS	MISSION (EXO SET). *	
		***EXTERNAL INTE	RRUPT ************************************	
			RRUPT ************************************	
;***** XINT1:	SETB RETI	P1.7	; DISABLE CTS PIN	
;***** XINT1:	SETB RETI	P1.7	; DISABLE CTS PIN ; RETURN.	
;***** XINT1: ;****	******* SETB RETI ******* CLR JB	P1.7 *** SERIAL INTER SI AM,HOP	; DISABLE CTS PIN ; RETURN.	
;***** XINT1: ;****	******** SETB RETI ******** CLR JB CLR	Pl.7 *** SERIAL INTER SI AM,HOP EA	; DISABLE CTS PIN ; RETURN. PRUPT ROUTINE ************************************	
;***** XINT1: ;****	******* SETB RETI ******** CLR JB CLR MOV	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B	; DISABLE CTS PIN ; RETURN. PRUPT ROUTINE ************************************	
;***** XINT1: ;****	******** SETB RETI ******** CLR JB CLR	Pl.7 *** SERIAL INTER SI AM,HOP EA	; DISABLE CTS PIN ; RETURN. PRUPT ROUTINE ************************************	
;***** XINT1: ;****	******* RETI ******* CLR JB CLR MOV MOV SETB SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H	; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************	
;***** XINT1: ;****	******* RETI ******* CLR JB CLR MOV MOV SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,≹01000010B TL0,∛0F8H TR0	; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************	
;***** XINT1: ;****	******* RETI ******* CLR JB CLR MOV MOV SETB SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	
;***** XINT1: ;***** SIINT: ;	SETB RETI CLR JB CLR MOV MOV SETB SETB RETI	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA	<pre>; DISABLE CTS PIN ; RETURN. :RUPT ROUTINE ************************************</pre>	
;***** XINT1: ;***** SIINT: ;	SETB RETI JB CLR MOV SETB SETB RETI JB SETB SETB RETI JB SETB CLR	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA TBF,GETI TBF,GETI TBF,J7	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	
;***** XINT1: ;***** SIINT: SIINT: ; HOP:	******* SETB RETI ******* CLR JB CLR MOV SETB RETI JB SETB RETI JB SETB CLR ACALL	P1.7 *** SERIAL INTER SI AM,HOP EA STS,≹01000010B TL0,∛0F8H TR0 EA TBF,GETI TBF P1.7 TRAN	<pre>; DISABLE CTS PIN ; RETURN. :RUPT ROUTINE ************************************</pre>	
;***** XINT1: ;***** SIINT: ;	SETB RETI CLR JB CLR MOV MOV MOV SETB SETB RETI JB SETB CLR ACALL JB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA TBF,GETI TBF P1.7 TRAN RBE,RETURN	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	
;***** XINT1: ;***** SIINT: SIINT: HOP:	******* SETB RETI ******* CLR JB CLR MOV MOV SETB SETB RETI JB SETB CLR ACALL JB CLR	P1.7 *** SERIAL INTER SI AM,HOP EA STS,\$01000010B TLO,*078H TRO EA TBF,GETI TBF,GETI TBF,GETI TBF,GETI TBF,GETI TBF,CETU TBF,CETU	<pre>; DISABLE CTS PIN ; RETURN. :RUPT ROUTINE ************************************</pre>	
;***** XINT1: ;***** SIINT: SIINT: HOP:	SETB RETI JB CLR JB CLR MOV SETB SETB RETI JB CLR ACALL JB CLR SETB MOV	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TRO EA TBF,GETI TBF,GETI TBF,GETI TBF,17 TRAN RBE,RETURN EA RBE RETURN EA TL0,#0F8H	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	
;***** XINT1: ;***** SIINT: SIINT: HOP:	******* SETB RETI JB CLR MOV SETB SETB CLR ACALL JB SETB CLR SETB MOV SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS, #01000010B TL0, #0F8H TR0 EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA RBE TL0, #0F8H TR0	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	
;***** XINT1: ;***** SIINT: SIINT: HOP:	******* SETB RETI JB CLR JB CLR SETB SETB SETB CLR ACALL JB CLR SETB SETB SETB SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS, #01000010B TL0, #0F8H TR0 EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA RBE TL0, #0F8H TR0	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019_38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN	******* SETB RETI JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB CLR SETB CLR SETB SETB SETB SETB SETB SETB SETB	P1.7 *** SERIAL INTER AM,HOP EA STS,#01000010B TL0,#0F8H TRO EA TBF,GETI TBF,GETI TBF,GETI TBF,GETI TBF,GETI TBF,CETURN EA RDE RDE TL0,#0F8H TRO EA	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN	******* SETB RETI JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB CLR SETB CLR SETB SETB SETB SETB SETB SETB SETB	P1.7 *** SERIAL INTER AM,HOP EA STS,#01000010B TL0,#0F8H TRO EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA RBE TL0,#0F8H TRO EA ***** TRANSMIT S	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN ;****	******* SETB RETI JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB CLR SETB SETB SETB SETB SETB SETB SETB SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA TL0,#0F8H TR0 PTR,#200H R5,#0FFH	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN: ;*****	SETB RETI CLR JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB SETB CLR ACALL JB SETB CLR SETB SETB SETB SETB SETB SETB MOV MOV	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA RBE TL0,#0F8H TR0 EA ***** TRANSMIT S DPTR,#200H R5,#0FFH A,#008H	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN ;****	******* SETB RETI JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB CLR SETB SETB SETB SETB SETB SETB SETB SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TRO EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA RBE TL0,#0F8H TRO EA ***** TRANSMIT S DPTR,#200H R5,#0FFH A,#0A8H A,SIUST,WAIT	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN: ;*****	SETB RETI CLR JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB SETB CLR ACALL JB SETB CLR SETB SETB SETB SETB SETB SETB MOV MOV	P1.7 *** SERIAL INTER SI AM,HOP EA STS,\$01000010B TL0,#0F8H TRO EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA ***** TRANSMIT S DPTR,#200H R5,#0FFH A,#0A8H A,SIUST,WAIT A,@DFFR	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN: ;*****	******* SETB RETI JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB CLR SETB SETB SETB SETB SETB SETB SETB SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,\$01000010B TL0,#0F8H TRO EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA ***** TRANSMIT S DPTR,#200H R5,#0FFH A,\$1057,WAIT A,@DPTR @R1,A DPTR	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN: ;*****	SETB RETI CLR JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB CLR SETB CLR ACALL JB SETB CLR ACALL SETB MOV SETB SETB SETB SETB SETB SETB MOV MOV SETB SETB DJNZ	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA RBE TL0,#0F8H TR0 EA ***** TRANSMIT S DPTR,#200H R5,#0FFH A,#0ABH	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN: ;*****	******* SETB RETI JB CLR MOV SETB RETI JB SETB CLR ACALL JB CLR SETB CLR ACALL JB SETB SETB SETB SETB SETB SETB MOV SETB SETB SETB MOV SETB SETB SETB SETB SETB SETB SETB SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,\$01000010B TL0,#0F8H TRO EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA ***** TRANSMIT S DPTR,#200H R5,#0FFH A,\$1057,WAIT A,@DPTR @R1,A DPTR	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: HOP: GETI: GETI: TRAN: WAIT:	SETB RETI CLR JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB CLR SETB CLR ACALL JB SETB CLR ACALL SETB MOV SETB SETB SETB SETB SETB SETB MOV MOV SETB SETB DJNZ	<pre>Pl.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA TBF,GETI TBF Pl.7 TRAN RBE,RETURN EA RBE TI0,#0F8H TR0 EA ***** TRANSMIT S DPTR,#200H RS,#0FFH A,#0A8H A,SUUST,WAIT A,0DPTR EA,PTH SIUST,WAIT SIUST,#57H</pre>	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: ; HOP: GETI: RETURN: ;*****	******* SETB RETI ******* CLR JB CLR WOV MOV SETB RETI JB SETB CLR ACALL JB CLR SETB SETB SETB SETB SETB SETB SETB SETB	P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA RBE TL0,#0F8H TR0 EA ***** TRANSMIT S DPTR,#200H R5,#0FFH A,#0ABH	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38
;***** XINT1: ;***** SIINT: HOP: GETI: GETI: TRAN: WAIT:	******* SETB RETI JB CLR MOV SETB SETB RETI JB SETB CLR ACALL JB CLR SETB SETB SETB SETB SETB SETB SETB SETB	<pre>P1.7 *** SERIAL INTER SI AM,HOP EA STS,#01000010B TL0,#0F8H TR0 EA TBF,GETI TBF P1.7 TRAN RBE,RETURN EA RBE TL0,#0F8H TR0 EA ***** TRANSMIT S DPTR,#200H RS,#0FFH A,#10A8H A,SIUST,WAIT A,@DPTR R5,NXTI SIUST,#57H SIUST,#57H</pre>	<pre>; DISABLE CTS PIN ; RETURN. RUPT ROUTINE ************************************</pre>	292019–38



REC:	CLR MOV MOV MOV	TRO DPTR,#200H R5,#OFFH	; TURN OFF COUNTER 0 ; EXT. RAM RECEIVE BUFFER START ; EXT. RAM RECEIVE BUFFER LENGTH	
HOLD:		A,#08H A,SIUST,HOLD SIUST,#0EFH	; ADDRESS STATE ; WAIT FOR ADDRESS BYTE ; MOVE "BYP" INTO CONTROL STATE ; SKIP THE ADDRESS STATE	
WAIT1:	MOV CJNE	A,SIUST,WAIT1 A,RCB A,STAD,WAIT2	; PUSH-1 STATE ; WAIT FOR THE ADDRESS BYTE ; MOVE THE RECEIVED ADDRESS BYTE TO ACC. ; ADDRESS MATCHED?	
WAIT2:	SJMP MOV MOV RETI	WAIT3 RCB,#00010000B SIUST,#0CFH	; YES. ; MOVE INFO. CONTROL BYTE TO RCB ; MOVE "BYP" INTO BOV-LOOP STATE ; RETURN	
WAIT3:	MOV MOV	SIUST,#OEFH A,#18H	; MOVE "BYP" INTO CONTROL STATE ; FUSH-1 STATE	
WAIT4:		A,SIUST,WAIT4 A,RCB ACC.0,RTRN	; WAIT FOR THE CONTROL BYTE ; MOVE RECEIVE CONTROL BYTE INTO ACC. ; IF NOT AN I-FRAME RETURN ; SAVE RECEIVE CONTROL BYTE	
NEXTI:	MOV		; PUSH "BYP" INTO CONTROL STATE(10H). ; PUSH-1 STATE	
WAIT5:	CJNE MOV MOVX INC	A,SIUST,WAIT5 A,RCB @DPTR,A DPTR	; WAIT FOR AN I-BYTE ; WAVE RECEIVED I-BYTE INTO ACC. ; MOVE DATA TO EXT. RAM ; INCREMENT PTR TO EXTERNAL RAM ; IS IT THE LAST I-BYTE? ; YES. RESTORE THE CONTENTS OF RCB	
RTRN: END	RETI		; RETURN	

INTEL, SUPPLY FILLER



INTEL CORPORATION, 2200 Mission College Blvd., Santa Clara, CA 95052; Tel. (408) 765-8080 INTEL CORPORATION (U.K.) Ltd., Swindon, United Kingdom; Tel. (0793) 696 000 INTEL JAPAN k.k., Ibaraki-ken; Tel. 029747-8511

Printed in U.S.A./xxxx/1296/B10M/xx xx