Capítulo 8

Entrada/Saída em Microprocessadores

8.1 Introdução

A ENTRADA/SAÍDA (E/S) é a transferência de dados entre a lógica que faz parte do sistema de microcomputador e a lógica que está fora do sistema de microcomputador

Existem vários modos para realizar a transferência de dados entre o sistema de microcomputador e a lógica externa. As três categorias principais são:

E/S por Varredura: todas as transferências de dados entre o sistema de microcomputador e a lógica externa são completamente controladas pelo microprocessador, ou mais precisamente, por um programa que o microprocessador executa.

A característica chave de uma E/S programada é que a lógica externa atua de maneira que é comandada pelo programa que o microcomputador executa.

E/S por Interrupção: As interrupções são um meio que a lógica externa possui para forçar o sistema de microcomputador a suspender aquilo que estiver fazendo no momento para atender às necessidades delas (da lógica externa).

Acesso Direto à Memória: permite a movimentação de dados entre a memória do microcomputador e a lógica externa sem envolver o microprocessador nesta transferência. Os requerimentos físicos para cada tipo de E/S serão descritos adiante.

8.1.1 E/S POR VARREDURA

Na entrada e saída por varredura, os dados são transferidos entre um sistema de microcomputador e a lógica externa, em ambas direções, pela porta de E/S.

Quando um dispositivo da lógica externa transmite dados ao microcomputador, ele coloca o dado nos pinos da porta de E/S, onde, então, será armazenado no *Buffer* de E/S.

8.1.2 E/S PARALELA

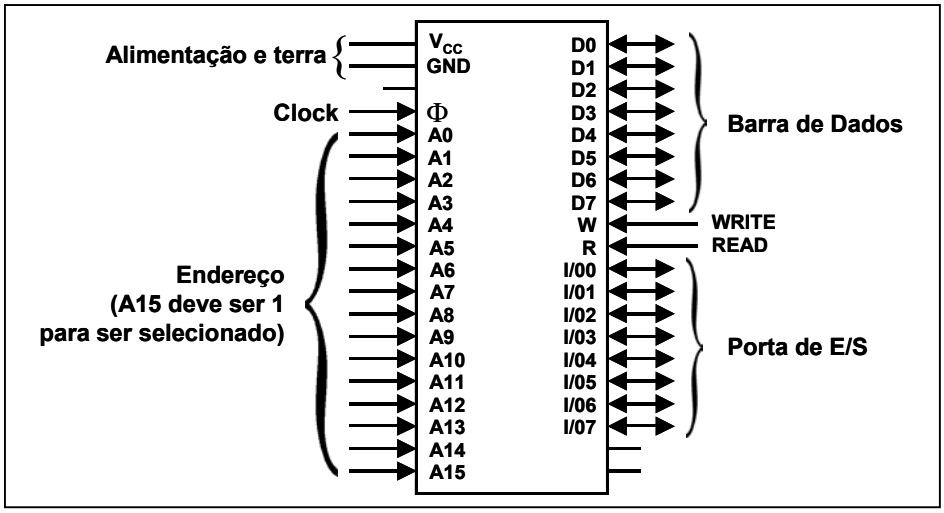
Na entrada e saída paralelas a porta Buffer de E/S não pode estar constantemente se comunicando com as linhas de dados da barra externa do sistema, uma vez que as linhas de dados da barra externa do sistemas podem estar carregando dados da (ou para a) memória. Chama-se isto de conflito de dados, ou seja, os dados que saem de um determinado dispositivo são colocados na barra de dados num instante em que dados vindos de um outro dispositivo já estavam caminhando na barra.

O microprocessador, portanto, deverá selecionar a porta de E/S e ler seu conteúdo (no Buffer de E/S), da mesma forma que um dado é lido da memória.

8.2 DISPOSITIVO DE E/S PARALELO SÍNCRONO

8.2.1 DISPOSITIVO DE E/S PARALELA

Nos dispositivos de E/S paralela os dados são escritos ou lidos em oitos unidades binárias paralelas, simultaneamente.

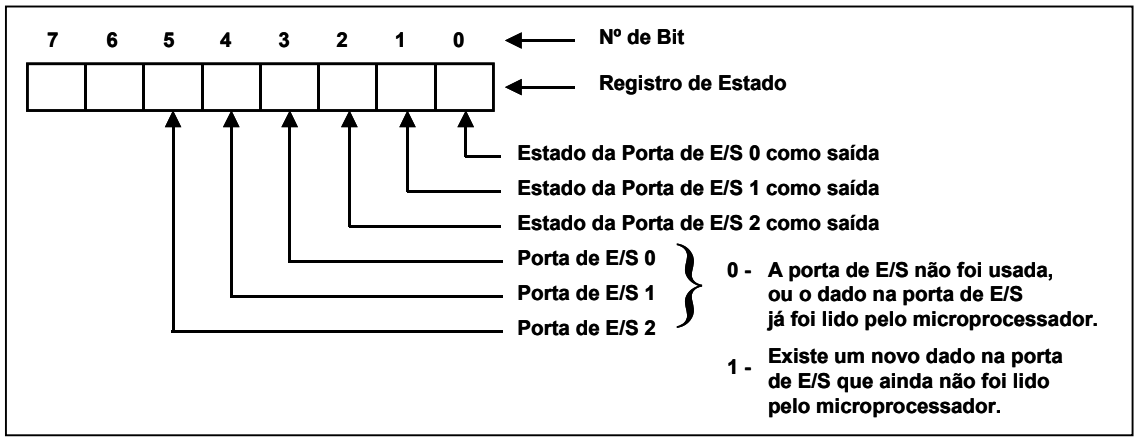


O número de portas que um dispositivo de E/S pode ter é função do número de pinos que são economicamente viáveis num encapsulamento DIP.

8.3 PORTA DE SAÍDA X ENTRADA

Quando o processador escreve dados numa porta de E/S paralela e a lógica externa lê este dado, a porta de E/S deve ser configurada como um porta de saída. Uma porta de E/S pode ser também configurada como porta de entrada, mas ,neste caso, a lógica externa transmite dados para a porta de E/S paralela, e o microprocessador lê este dado da porta de E/S.

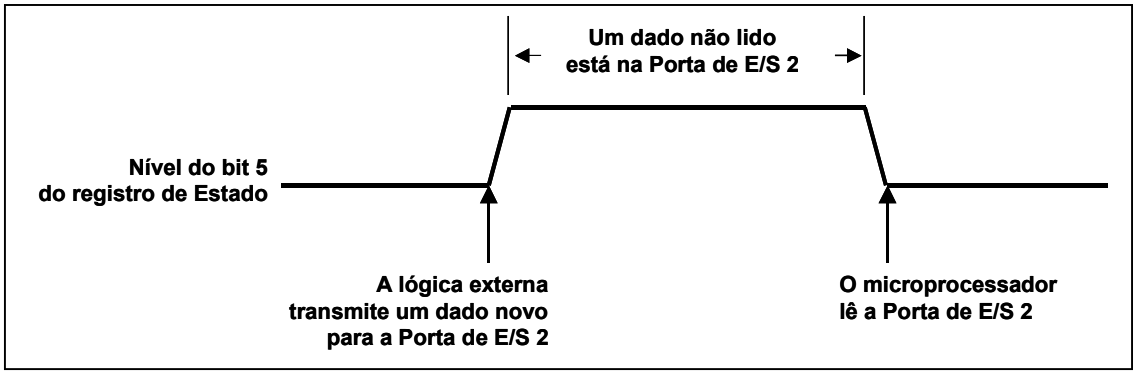
Um problema similar de estado existe quando uma porta de E/S paralela está configurada como porta de entrada. Após a leitura do dado de uma porta de entrada, o microprocessador necessita de um meio para saber que um novo dado foi escrito pela lógica externa na porta de entrada, antes de tentar ler desta porta de entrada outra vez. Poderemos usar mais 3 bits do registro de Estado para sanar este problema, como se segue:



Inicialmente os bits de registro do Estado 5, 4 e 3 serão 0. Quando a lógica externa transmitir um dado para uma porta de E/S, o bit de registro de Estado correspondente será setado em 1 pela lógica interna do dispositivo de E/S paralela. Por exemplo, se a lógica externa mandar um dado para porta de E/S 2, o bit 5 do registro de Estado será imediatamente colocado em 1. Quando o microprocessador ler este dado, o bit de registro de Estado (bit 5 para porta de E/S 2) será imediatamente colocado em 0 pela lógica interna do dispositivo de E/S paralela.

O microprocessador pode agora verificar o registro de Estado procurando por um 1 no bit apropriado deste registro como um indicador que um novo dado está disponível para leitura numa porta de E/S qualquer. Como o registro de Estado terá o bit da porta de E/S correspondente resetado imediatamente após a leitura do dado, a lógica do programa pode se assegurar que não se leu um mesmo dado duas vezes. Se o microprocessador ler o conteúdo de uma porta de E/S, enquanto seu bit de Estado correspondente for 0, então ele relerá um dado previamente lido.

8.3.1 Seqüência de eventos de uma porta de entrada



8.4 E/S POR INTERRUPÇÃO

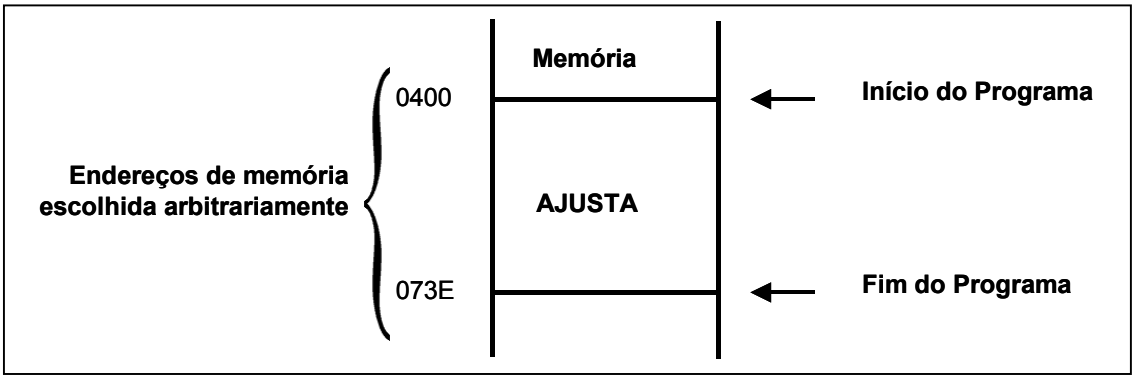
A maioria dos microprocessadores possui um sinal de controle por meio do qual a lógica externa pode exigir a atenção do microprocessador. Este sinal é chamado pedido por interrupção porque, efetivamente, a lógica externa está pedindo ao microprocessador para interromper tudo o que estiver fazendo naquele instante, para atender necessidades mais urgentes da lógica externa.

8.5 EXEMPLO de E/S POR INTERRUPÇÃO

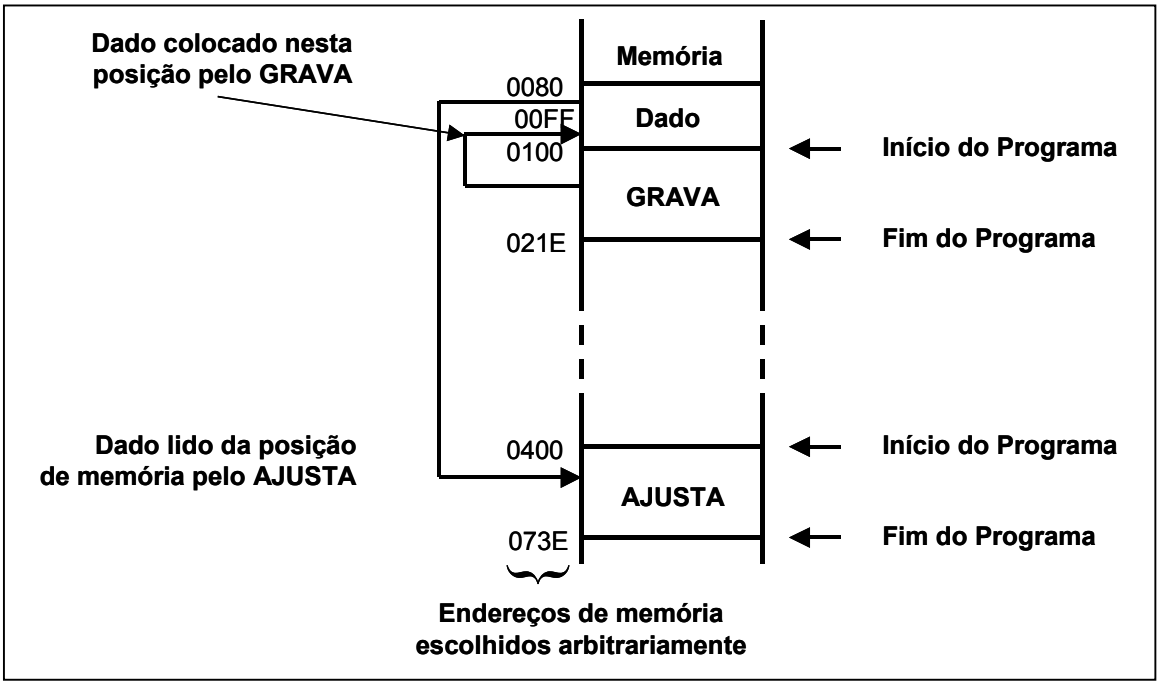
Suponhamos que um sistema de microcomputador esteja sendo usado para controlar a temperatura da água de um chuveiro, como ilustrado na Figura 5.8. Um termômetro mede a temperatura da água misturada quente e fria, na boca do chuveiro, e transmite esta temperatura como um sinal digital para o sistema de microcomputador. O sistema de microcomputador compara esta temperatura com a temperatura escolhida previamente num ponto de configuração, que é fornecido por um controle apropriado. Dependendo da diferenção entra temperatura real e a desejada da água do chuveiro. O sistema de microcomputador envia um dado que deve ser interpretado como um sinal de controle de válvula, fazendo a válvula aumentar ou diminuir o fluxo de água quente.

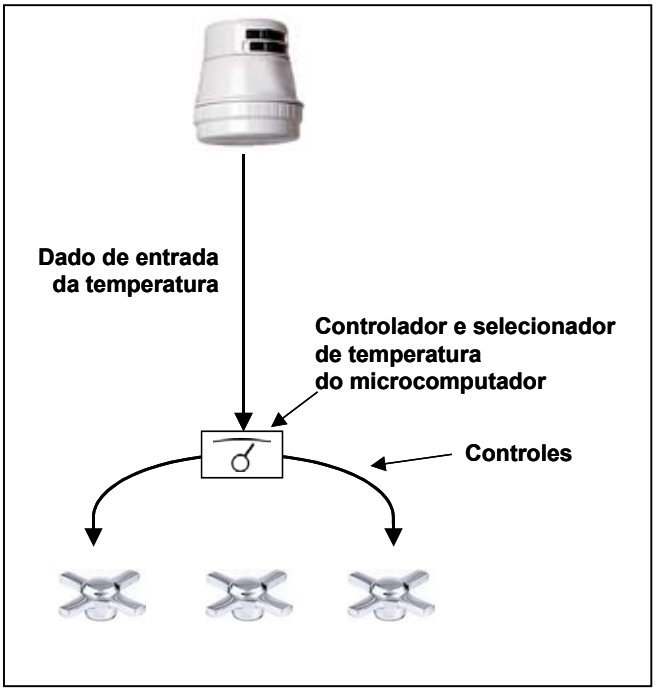
Complexidade:

Existe um atraso entre o tempo que se ajusta uma torneira de chuveiro e o tempo em que a água saindo da boca do chuveiro muda sua temperatura. Por esta razão, um programa não muito trivial terá que ser executado pelo microcomputador, para assegurar que ele não será acionado para fazer ajustes desnecessários. Chamaremos este programa de AJUDA e o ilustraremos armazenado na memória do programa, como se segue.



Um outro programa, chamado GRAVA, irá receber dados do sensor de temperatura, e interpretar corretamente estes dados para que representem leitura de temperatura. O único contato entre os programas GRAVA e AJUSTA é que o AJUSTA irá procurar um dado antecipadamente numa certa região da memória de dados, e o GRAVA irá colocar este dado na forma correta nesta posição da memória de dados. Nossa memória se parecerá então assim:


Dado colocado nesta posição pelo GRAVA
Memória
0080
00FF
Dado
0100
GRAVA
021E
Início do Programa
Fim do Programa
Dado lido da posição de memória pelo AJUSTA
0400
AJUSTA
073E
Início do Programa
Fim do Programa
Endereços de memória escolhidos arbitrariamente


Dado de entrada da temperatura
Controlador e selecionador de temperatura do microcomputador
Controles

8.6 Lógica Externa à UCP

8.6.1 Faltando um Pedido de Interrupção

A maneira pela qual a temperatura da boca do chuveiro é lida e transmitida ao sistema de microcomputador é uma outra característica deste problema que não é tão simples quanto possa parecer. Tomará aproximadamente meio segundo para um sensor de temperatura de baixo custo armazenar a temperatura. Meio segundo pode não parecer muito tempo, mas um microprocessador pode executar aproximadamente duzentos e cinqüenta mil instruções durante este período.

Como o microprocessador poderá saber quando o sensor de temperatura já tem um novo valor para transmitir? Se o sensor de temperatura simplesmente tentar enviar dados para uma porta de E/S, o microprocessador estará bem propício a não ler este dado. Uma leitura de temperatura pode se perder facilmente no meio de 250 mil instruções executadas. Se a lógica do programa simplesmente "tenta ao acaso" a leitura do dado de uma porta de E/S uma vez a cada 250 mil instruções, existe uma possibilidade mínima de pegar todas as leituras de temperatura, e ler apenas uma vez.

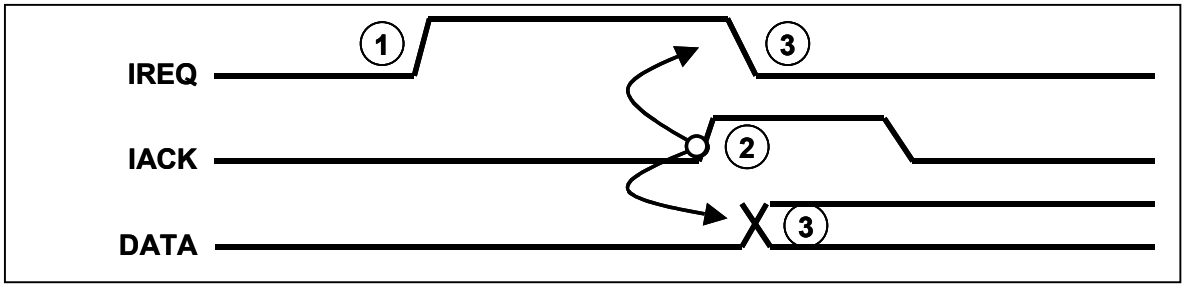
E a respeito da utilização do registro de Estado da porta de E/S? Sim, a lógica do programa poderia ler o estado da porta de E/S, e então ler o dado da porta de E/S quando o estado indicasse a presença de um novo dado. Mas se o programa tem outras coisas a fazer, a utilização do registro de Estado da porta de E/S irá desperdiçar, desta maneira, muito tempo. Na verdade, o microprocessador pode achar necessário ler o registro de Estado da porta de E/S centenas de vezes entre leitura de dados para garantir que nenhum dado de entrada foi perdido. Neste exemplo particular, o ponto importante a se observar é que a entrada de dados de E/S requer uma fração mínima do tempo do microprocessador; porém esta entrada é assíncrona. O microprocessador jamais poderá prever o intervalo de tempo que separa duas leituras de temperatura quaisquer.

8.6.2 Pedido de Interrupção

Uma outra forma de se resolver este problema está ilustrado na Figura 5.9. Uma seqüência de três etapas permite que o sensor de temperatura atraia a atenção do microprocessador, da seguinte maneira:

1. O sensor de temperatura envia um sinal de pedido de interrupção (INTERRUPT REQUEST – IREQ) ao microprocessador por meio de uma linha de controle da barra externa do sistema.
2. O microprocessador tem a escolha de aceitar ou rejeitar um pedido de interrupção, gerando um sinal de reconhecimento de interrupção (Interrupt acknowledge – LACK) numa linha de controle da barra externa do sistema.
3. O dispositivo externa usa o sinal de reconhecimento de interrupção como um habilitador (ENABLE), fazendo-o transmitir dados para a porta de E/S A. O dispositivo externo deve retirar o seu sinal de pedido de interrupção ao receber um reconhecimento de interrupção, desde que o pedido de interrupção foi tratado, o dispositivo externo não estará mais pedindo uma outra interrupção. O microprocessador sabe que o dado está disponível em uma porta de E/S específica, logo a lógica do programa necessária para processar o dado recebido torna-se bem simples.

A temporização para esta seqüência de três etapas pode ser ilustrada da seguinte forma:



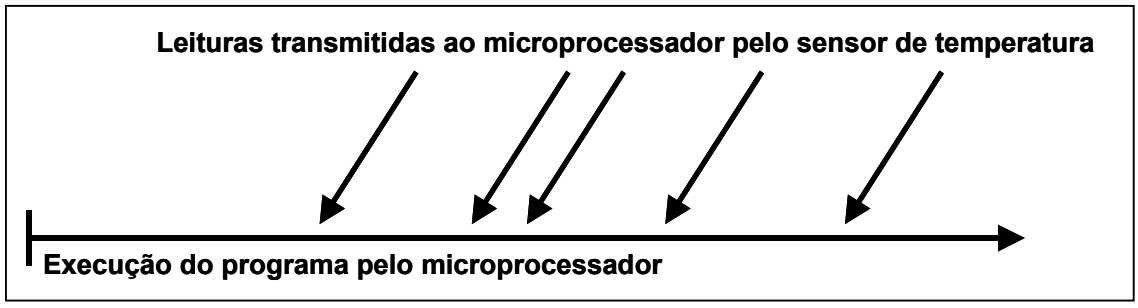
8.7 ACESSO DIRETO À MEMÓRIA

O sistema de microcomputador para controlar a temperatura do chuveiro irá desperdiçar um tempo enorme, simplesmente recebendo dados dos sensores de temperatura e armazenando-os num buffer de RAM. O programa GRAVA é executado com este objetivo.

O sensor de temperatura pode transmitir aproximadamente duas leituras de temperatura por segundo. Para um microprocessador isto equivale a receber uma leitura de temperatura a cada 250 mil instruções executadas, aproximadamente.

8.7.1 Eventos Assíncronos

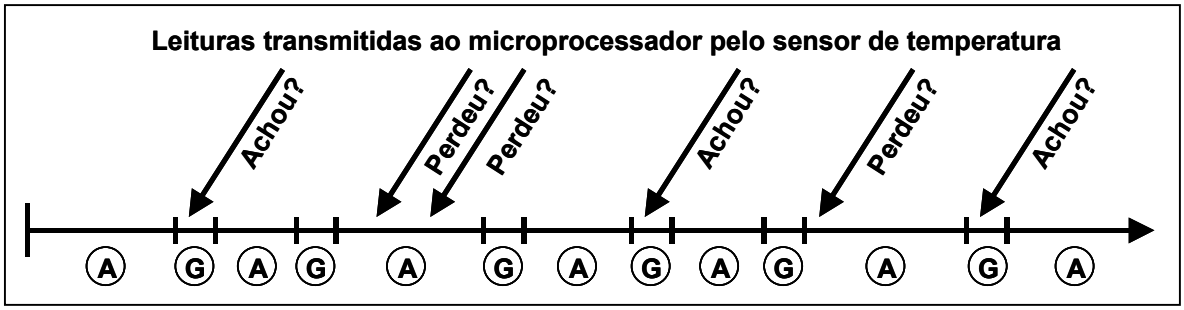
Um sensor de temperatura barato não irá transmitir exatamente duas leituras de temperatura por segundo. Na verdade, podem existir variações consideráveis no período de tempo entre as transmissões de temperatura. Como resultado, não poderemos prever, com nenhum grau de precisão, o atraso de tempo entre transmissões de dados consecutivas do sensor de temperatura para o sistema de microcomputador. Portanto, as transmissões de dados do sensor de temperatura para o sistema de microcomputador constituem-se em eventos assíncronos:



Devido as transmissões de dados do sensor para o sistema serem imprevisíveis (ou assíncronas), o programa GRAVA deverá ser executado cada vez que o sensor de temperatura transmite um item de dados. O programa GRAVA contém uma seqüência de instruções que move dados de uma porta de E/S para um byte de memória RAM; esta seqüência de instruções não pode fazer parte do programa AJUSTA, desde que a lógica deste programa não consegue detectar a chegada de um dado do sensor de temperatura.

Qualquer sistema que execute o programa GRAVA em intervalos de tempo fixos está sujeito a perder um grande número de transmissões de dados do sensor de temperatura.

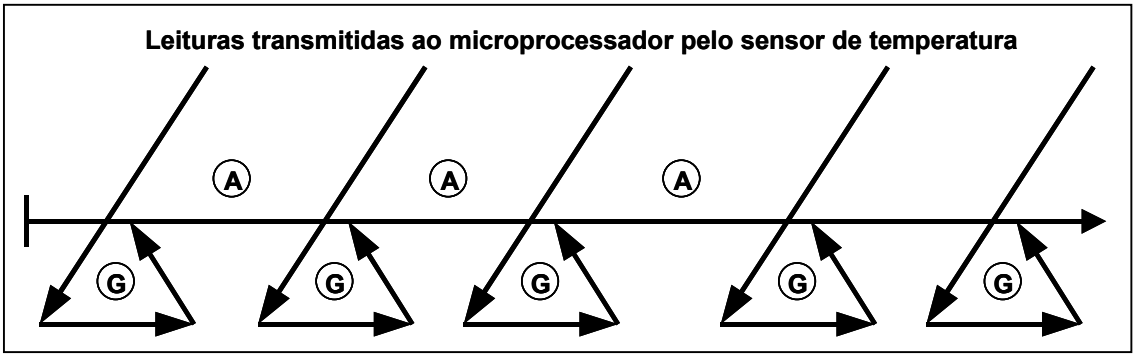
Eis um exemplo:



[A] representa a execução normal do programa AJUSTA.

[G] representa a execução periódica fixa da seqüência de instruções que grava dados transmitidos pelo sensor de temperatura para o sistema de microcomputador. Se o dado transmitido não chegar ao sistema de microcomputador durante [G], ele será perdido.

Em nossa discussão sobre interrupções, explicamos que o único processo segura de se pegar todos os dados transmitidos pelo sensor de temperatura é fazê-lo pedir uma interrupção quando estiver pronto para transmitir um item de dados. Em resposta ao pedido de interrupção do sensor de temperatura, o sistema de microcomputador executará a seqüência de instruções para pegar o dado, já caracterizado por [G]. A ilustração poderá ser agora modificada da seguinte maneira:



Cada vez que o sensor de temperatura estiver pronto para transmitir dados para o sistema, ele notifica ao microprocessador pedindo uma interrupção. Em resposta a este pedido, a lógica de programa suspende a execução do programa AJUSTA [A] e passa a executar o programa GRAVA [G]. O programa GRAVA lê a entrada de dados do sensor de temperatura, e o programa AJUSTA, então, continua sua execução (N.T. já de posse do dado) do ponto de suspensão.

Mesmo este método de gravação de dados transmitidos pelo sensor de temperatura não é muito eficiente. **Vejamos o que acontece quando a UCP aceita a interrupção e executa o programa GRAVA:**

1. A UCP está executando o programa AJUSTA [A] na ilustração citada. Quando a UCP sente um pedido de interrupção, ela executa uma seqüência de instruções que salva os registros e o estado dela; e então executa uma instrução para reconhecer a interrupção.

2. O programa GRAVA [G] na ilustração mostrada é executado. Este programa contém instruções que carregam um endereço de memória no contador de dados, lêem dados de uma porta de E/S para o acumulador, e então enviam este dado do acumulador para a palavra de memória endereçada pelo contador de dados. 3: O passo 1 é invertido. Os conteúdos dos registros e do estado são restaurados e o programa AJUSTA continua sua execução.

A única diferença entre todas as instruções que são executadas para implementar os três passos citados cada vez que o sensor de temperatura transmite uma leitura e [G] é re-executado, é o conteúdo do contador de dados no passo 2. O conteúdo do contador de dados será acrescido de um cada vez que o passo e é executado. Cinqüenta microssegundos, ou mais, serão necessários para processar repetidamente esta seqüência trivial de eventos. Antes de podermos decidir se isto é um problema sério ou inconseqüente, devemos fazer duas perguntas:

1. Estas operações são consideradas bem comuns, ou se trata de um caso especial e isolado? A resposta é que esta é uma das operações mais comuns realizadas por um microprocessador. Na verdade, não é apenas lendo dados de um dispositivo externo que um microprocessador perde muito tempo, ele perderá também quase tanto tempo ao transmitir dados rotineiramente entre *buffers* de RAM para dispositivos externos.
2. Se o microprocessador não gastar 50 microssegundos cada vez que tiver que receber dados de um dispositivo externo (ou enviar dados para um dispositivo externo), o que mais ele faria com este tempo? Em diversos casos de aplicações simples, a resposta seria nada, e nestes casos esta perda de tempo seria irrelevante.

Mas, claramente, quando uma aplicação de microcomputador começa a se tornar mais complexa, as perdas de tempo começam a se tornar mais críticas. Se levar 50 microssegundos para ler um dado de um dispositivo externo e mais 50 microssegundos para transmitir um dado para o mesmo dispositivo externo, então um sistema microcomputador poderia executar uma centena de transferências de dados por segundo, mas não sobraria tempo para fazer mais nada.

Devemos, portanto, concluir que existirão diversas aplicações onde o desperdício de tempo em processamento de interrupções se tornaria intolerável.

8.7.2 Roubo de Ciclo para Acesso Direto à Memória

O acesso direto à memória (ADM) permite uma forma mais rápida de mover dados entre as portas de E/S e a memória. Iremos criar um novo dispositivo para o nosso sistema de microcomputador, e neste dispositivo colocaremos um pouco de lógica parecida com a de uma UCP, dedicada à tarefa única de mover dados pela barra externa do sistema.

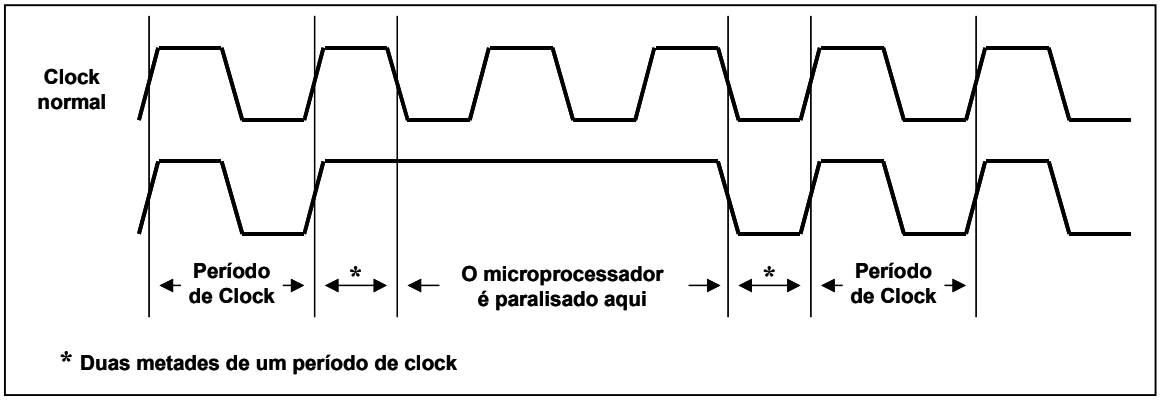
8.7.3 Dispositivo Controlador de ADM

Chamaremos este dispositivo de controlador de acesso direto à memória (AMD). Existem duas maneiras do controlador de ADM poder suprimir a lógica de UCP dentro do microprocessador; ele pode manipular o sinal de *clock* do microprocessador, ou poderá forçar o microprocessador a marcar tempo e flutuar suas conexões de sinais.

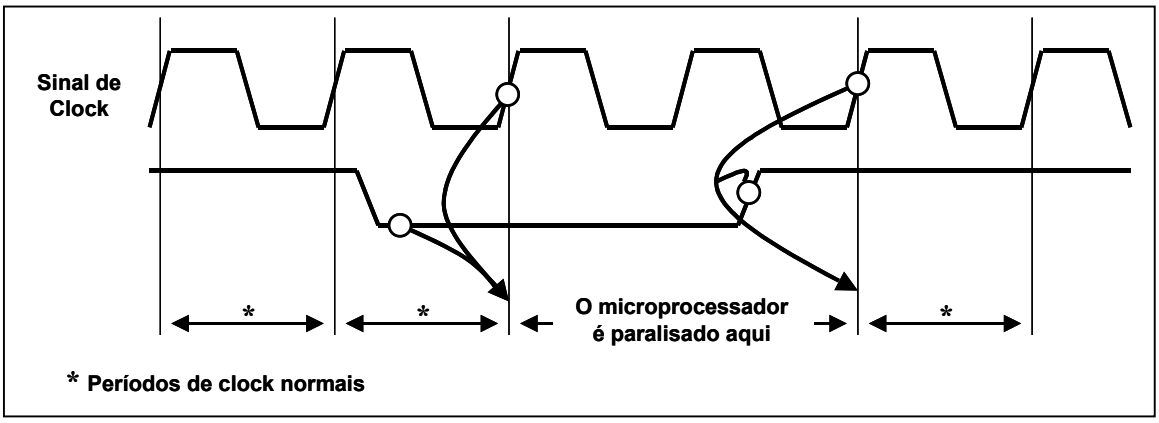
Sinal de Clock Alongado

Como toda UCP é excitada por um sinal de clock externo, se paralisarmos ou “alongarmos” os sinais de clock, pararemos o UCP.

Um sinal de clock “alongado” pode ser ilustrado assim:



O microprocessador pode ter uma entrada de controle que o faz colocar flutuando todas as conexões de pinos para barra externa do sistema, enquanto suspende as operações internas. Isto pode ser ilustrado assim:



Ambas as técnicas de desabilitação do microprocessador têm sido usadas por diferentes fabricantes de semicondutores para permitir o acesso direto à memória.

Inicialização do ADM

Um programa deve ser executado pela UCP para inicializar uma operação em ADM. O programa terá que carregar dados apropriados nos registros do controlador ADM. Não existe outra maneira dos dados chegarem aos registros de Endereço, Controlador e de Controle do dispositivo do ADM.

Para inicializar uma operação de ADM, o programa executado pelo microprocessador deverá realizar estes passos:

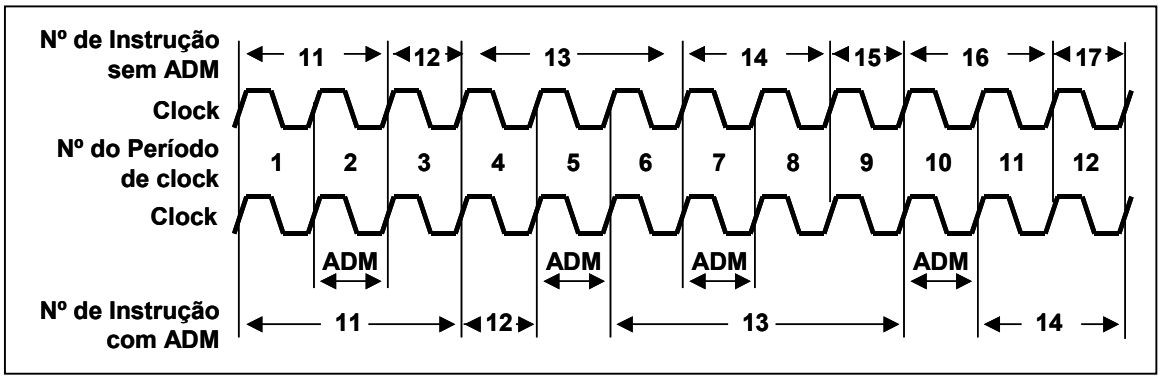
1: Transmitir um endereço de memória inicial para o registro de Endereço do controlador de ADM.

2: Transmitir uma contagem de palavras de memória para o registro Contador do controlador de ADM.

3: Transmitir um código de controle (03_{16} como já ilustrado) para o registro de Controle do dispositivo de ADM. O código de controle deve identificar a direção de transferência de dados e deve ligar o dispositivo de ADM.

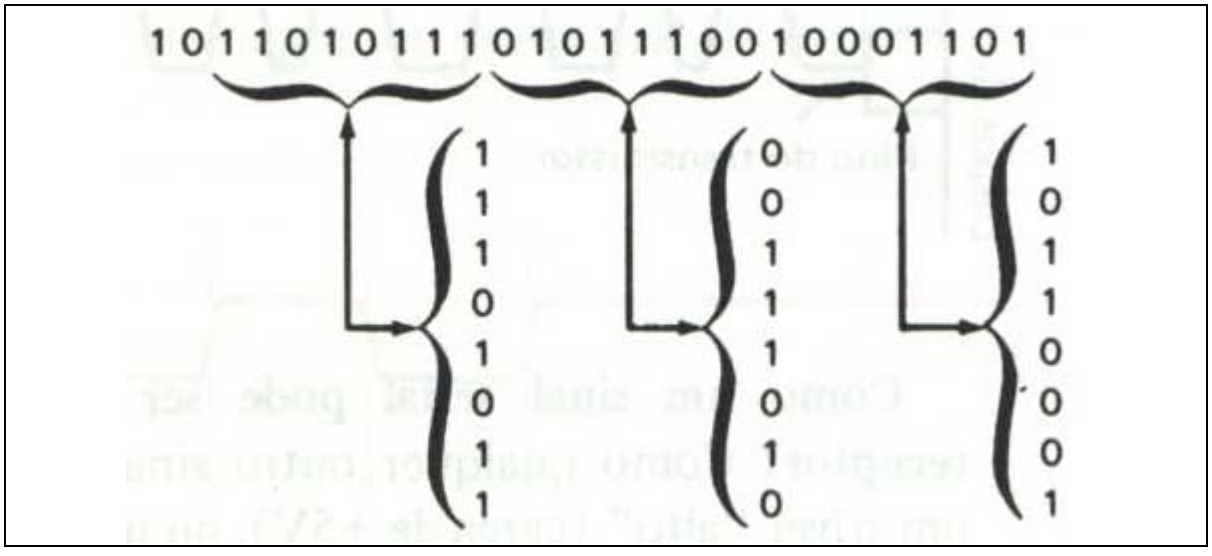
Observe que os sinais de controle de READ e WRITE do controlador de ADM e suas linhas de endereço são bidirecionais.

Devido à maneira pela qual ocorrem as operações em ADM, qualquer programa executado pelo microprocessador nem percebe que uma operação em ADM está ocorrendo – excetuando-se que a execução deste programa será ligeiramente mais lenta. Conceitualmente isto pode ser ilustrado da seguinte forma:

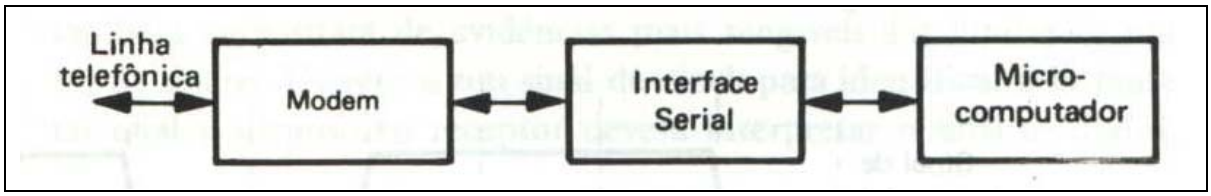


8.8 ENTRADA / SAÍDA SERIAL

Os dados são transferidos por meio de linhas telefônicas, serialmente. Existem também alguns dispositivos de E/S lentos, tais como o conhecido teletipo e os cassetes de fita magnética, que transmitem e recebem dados serialmente. Caso um microprocessador tiver que transmitir ou receber dados serialmente, então deverá possuir uma lógica de interfaceamento capaz de converter dados seriais em dados paralelos ou dados paralelos em dados seriais:



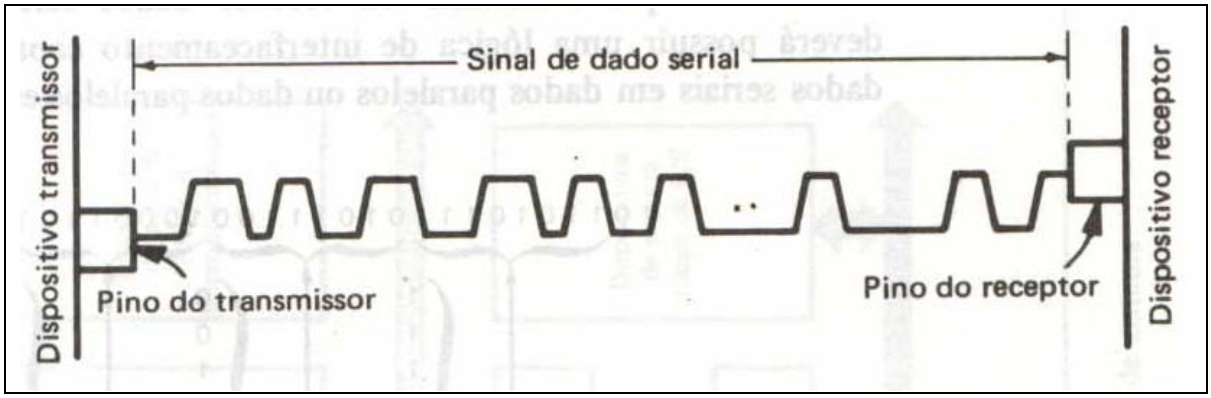
Esta é a configuração necessária para transferir dados de uma linha telefônica para um sistema de microcomputador.



Um modem é um dispositivo que pode transformar sinais de linha telefônica em níveis lógicos digitais, ou níveis lógicos em sinais de linha telefônica.

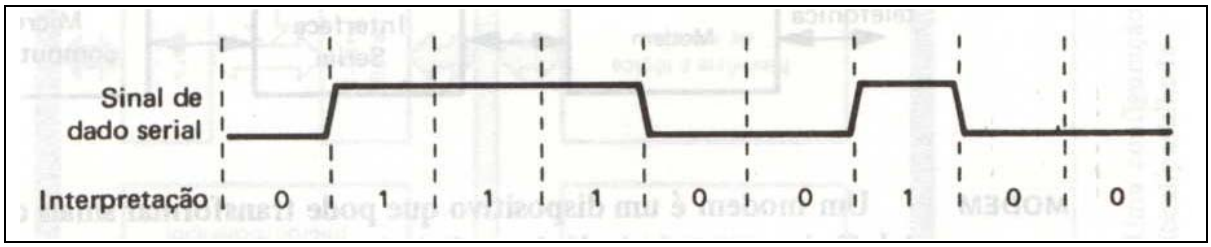
8.9 IDENTIFICAÇÃO DE BITS DE DADOS SERIAIS

A única propriedade de uma corrente de dados seriais é que os dados são transmitidos e recebidos como um único sinal, por um único pino de um dispositivo:



Como um sinal serial pode ser interpretado pelo dispositivo receptor? Como qualquer outro sinal digital, o sinal de dados terá um nível “alto” (cerca de +5V), ou um nível “baixo” (cerca de 0V).

Existem duas maneiras de se interceptar um dado serial. A primeira, e a mais óbvia, trata um sinal de nível baixo como um bit 0, e um sinal de nível alto como um bit 1; a seqüência de dígitos binários 011100100 seria decodificada de um sinal de dado serial, na seguinte forma:

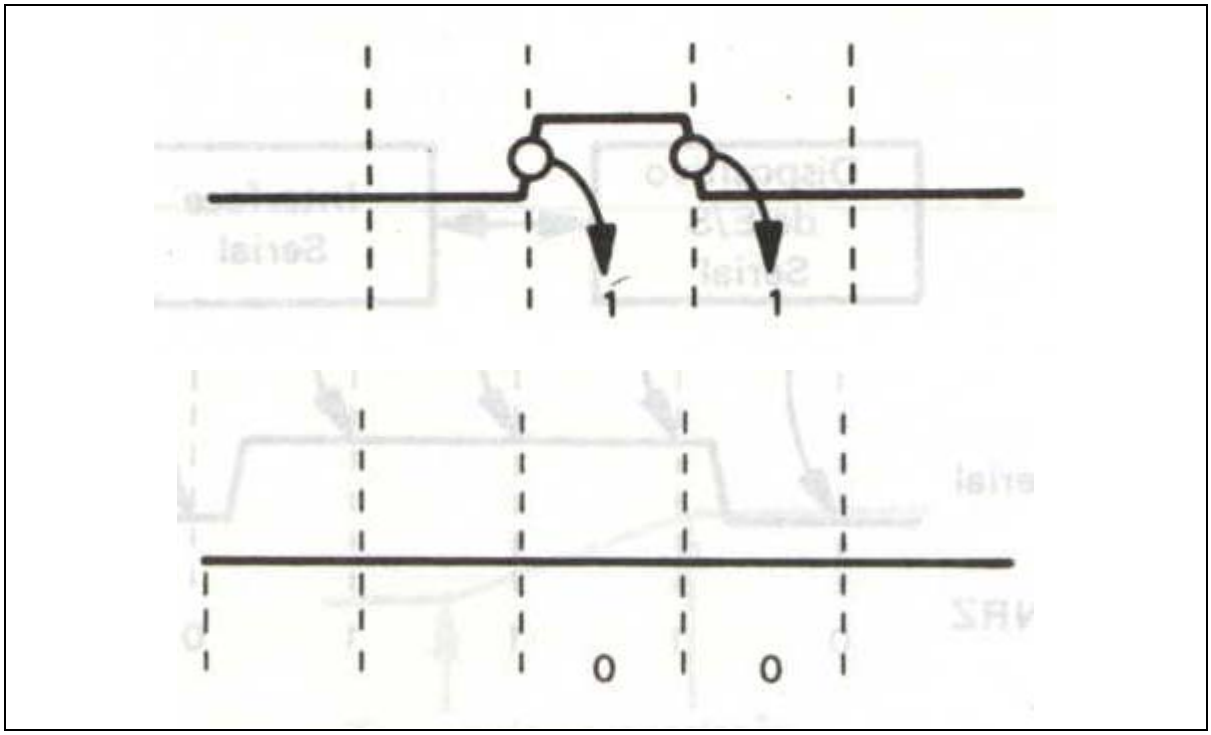


8.9.1 Dado Serial NRZ

A interpretação de dados seriais ilustrada é chamada dado serial NRZ (“non-return to zero” – não volta a zero).

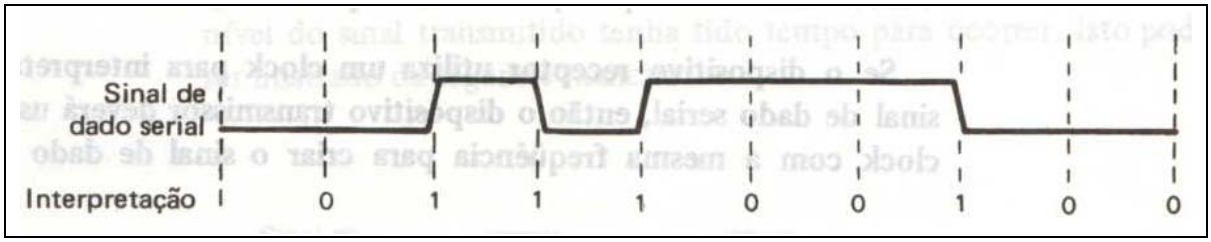
Dados Serial NRZI

NRZI (*non-return to zero inverted* – não volta a zero invertida) representa uma outra interpretação comum para dados seriais. O NRZI interpreta as transições de nível do sinal, em vez de interpretar os próprios níveis. Uma transição de nível representa um bit 1:

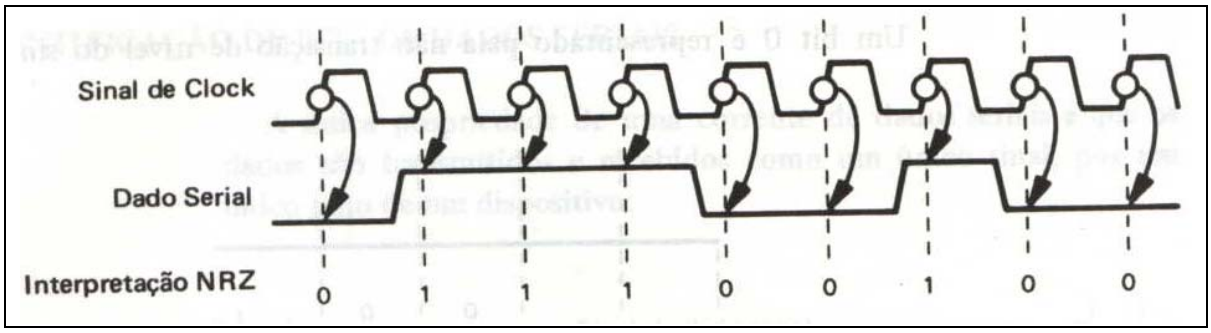


Sinal de Clock

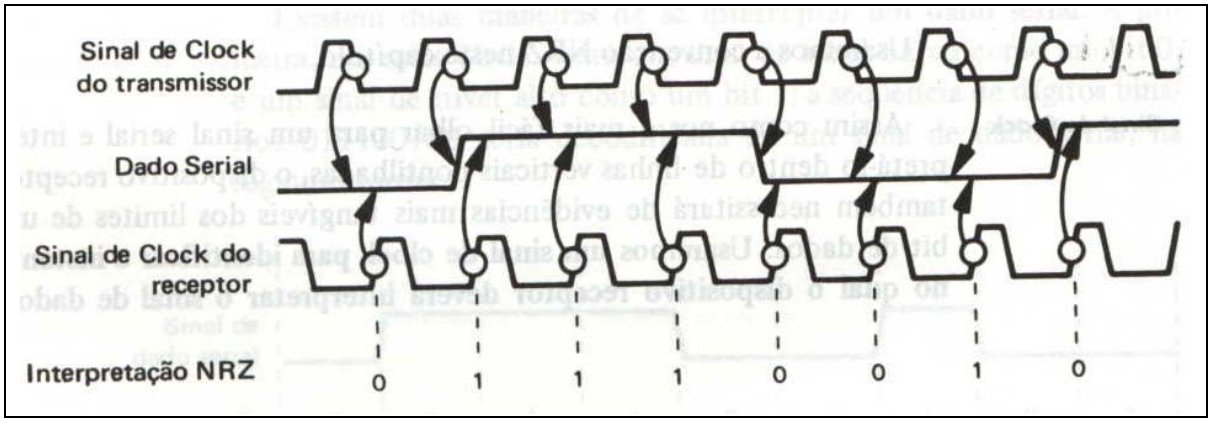
Assim como nos é mais fácil olhar para um sinal serial e interpretá-lo dentro de linhas verticais pontilhadas, o dispositivo receptor também necessitará de evidências mais tangíveis dos limites de bit de dados. Usaremos um sinal de clock para identificar o instante no qual o dispositivo receptor deverá interpretar o sinal de dados.



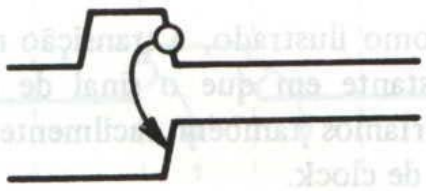
Como ilustrado, a transição negativa do sinal de clock identifica o instante em que o sinal de dados serial deverá ser amostrado. Poderíamos também facilmente amostrar na transição positiva do sinal de clock.



Um sinal de dado serial deve ser criado pelo dispositivo transmissor antes de poder ser interpretado pelo dispositivo receptor. Vamos analisar as implicações desta simples necessidade.
Se o dispositivo receptor utiliza um clock para interpretar um sinal de dado serial, então o dispositivo transmissor deverá usar um clock com a mesma freqüência para criar o sinal de dado serial:



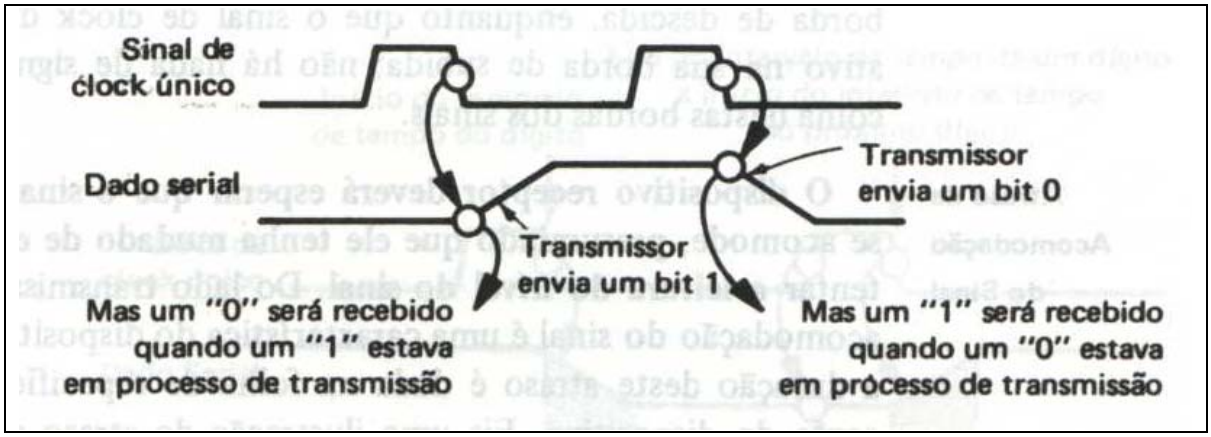
Os sinais de clock do transmissor e do receptor não podem ser idênticos; isto, porque, na verdade, um sinal leva um certo tempo finito para mudar de estado. As figuras ficam mais simples de entender quando as transições de sinais são desenhadas como ondas quadradas límpidas:



8.9.2 Tempo de Acomodação do Sinal

Mas na verdade, cada sinal que muda de estado requer um tempo de acomodação finito (*settling time*).

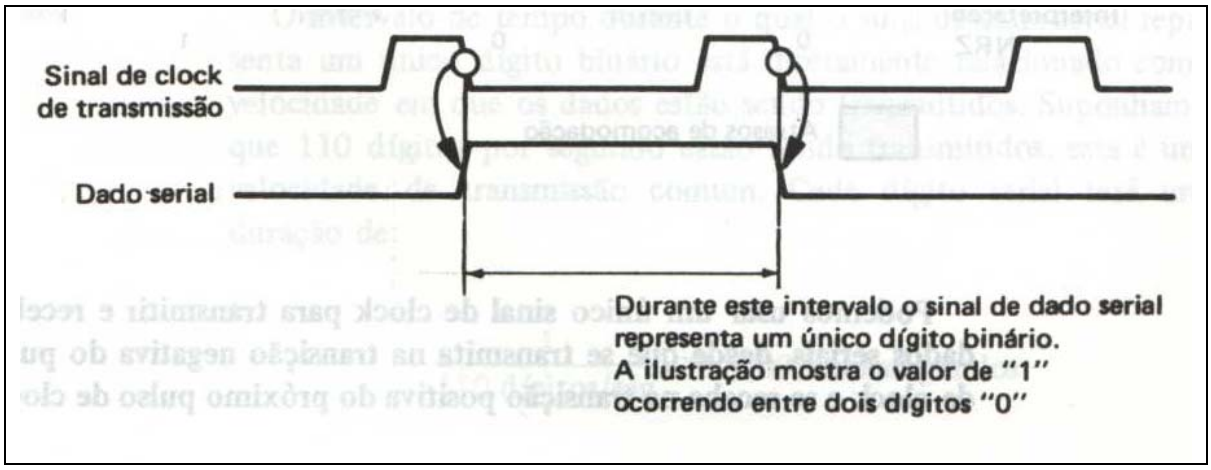
Se utilizarmos a mesma transição de *clock* para transmitir e receber dados seriais, então o tempo de acomodação do sinal fará com que a amostragem do sinal recebido se dê antes que a transição de nível do sinal transmitido tenha tido tempo para ocorrer. Isto pode ser ilustrado da seguinte maneira:



8.9.3 Sinal de *Clock* de Transmissão de Dados Seriais

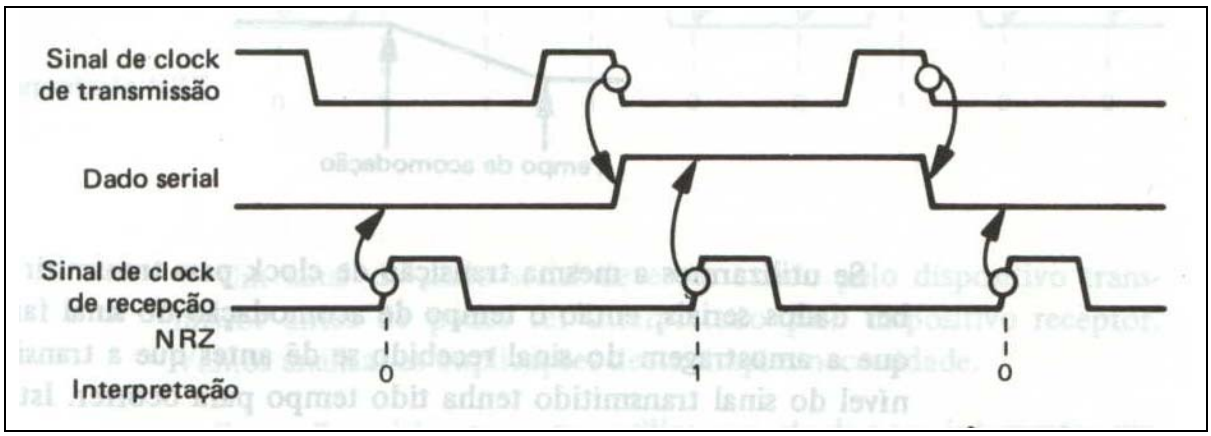
Concluímos, então que, os sinais de *clock* de transmissão e de recepção, embora tenham muito em comum, não podem ser um único sinal, sujeito a interpretações idênticas.

O sinal de *clock* de transmissão irá identificar a duração de um dígito binário:



8.9.4 Sinal de *Clock* de Recepção de Dados Seriais

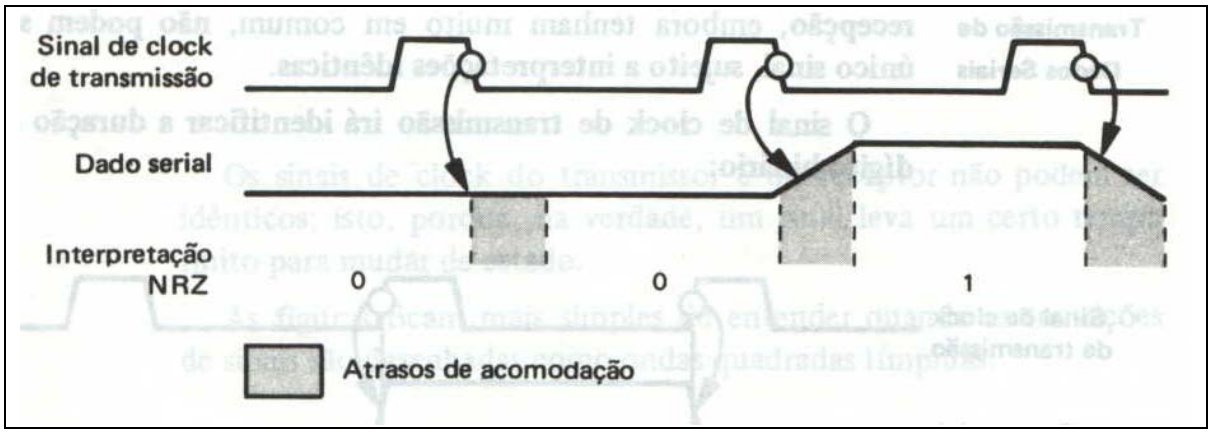
Em algum ponto dentro do intervalo de tempo de um único dígito, um sinal de clock de recepção irá identificar o nível do sinal de dado serial:



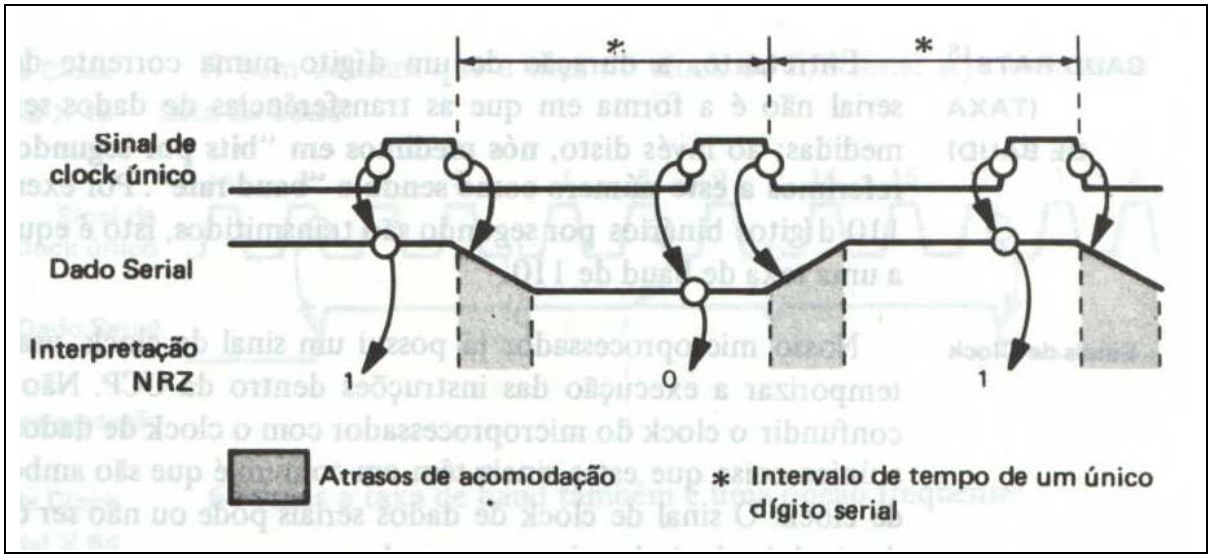
A ilustração mostra o sinal de *clock* de transmissão ativo na borda de descida, enquanto que o sinal de *clock* de recepção está ativo na sua borda de subida; não há nada de significativo na escolha destas bordas dos sinais.

8.9.5 Atraso de Acomodação do Sinal

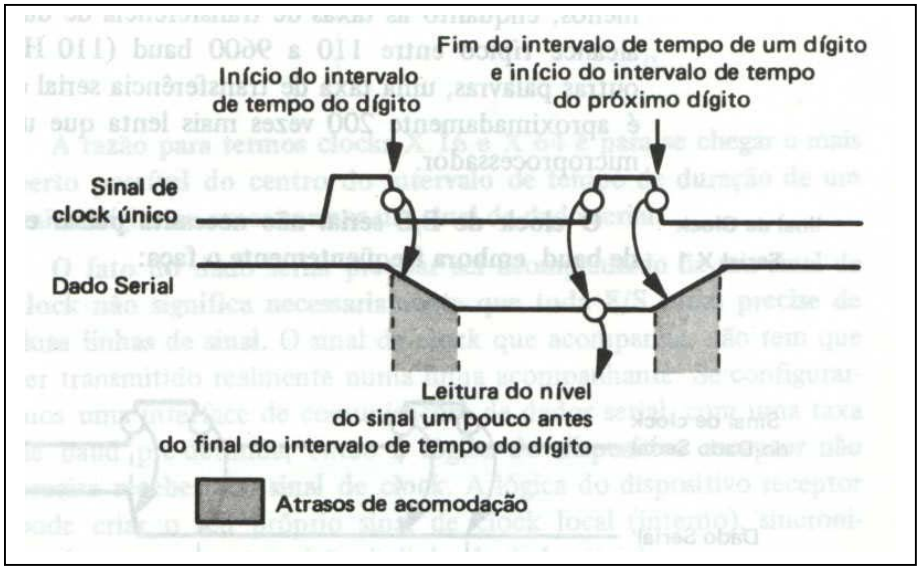
O dispositivo receptor deverá esperar que o sinal de dados serial se acomode, presumindo que ele tenha mudado de estado, antes de tentar a leitura do nível do sinal. Do lado transmissor, o atraso de acomodação do sinal é uma característica do dispositivo transmissor; a duração deste atraso é dada na folha de especificações do fabricante do dispositivo. Eis uma ilustração do atraso de acomodação:



Podemos usar um único sinal de clock para transmitir e receber dados seriais, desde que se transmita na transição negativa do pulso de clock e se receba na transição positiva do próximo pulso de clock:



Observemos cuidadosamente como o nível do sinal transmitido será recebido:



O intervalo de tempo durante o qual o sinal de dado serial representa um único dígito binário está diretamente relacionado com a velocidade em que os dados estão sendo transmitidos. Suponhamos que 110 dígitos por segundo estão sendo transmitidos; esta é uma velocidade de transmissão comum. Cada dígito serial terá uma duração de:

$$\frac{1}{\text{110 dígitos/seg}} = 9{,}091\text{ms}$$

8.10 *BAUD RATE* (TAXA DE BAUD)

Entretanto, a duração de um dígito numa corrente de dados serial não é a forma em que as transferências de dados seriais são medidas; ao invés disto, nós medimos em "bits por segundo" e nos referimos a este número como sendo o "baud rate". Por exemplo, se 110 dígitos binários por segundo são transmitidos, isto é equivalente a uma taxa de baud de 110.

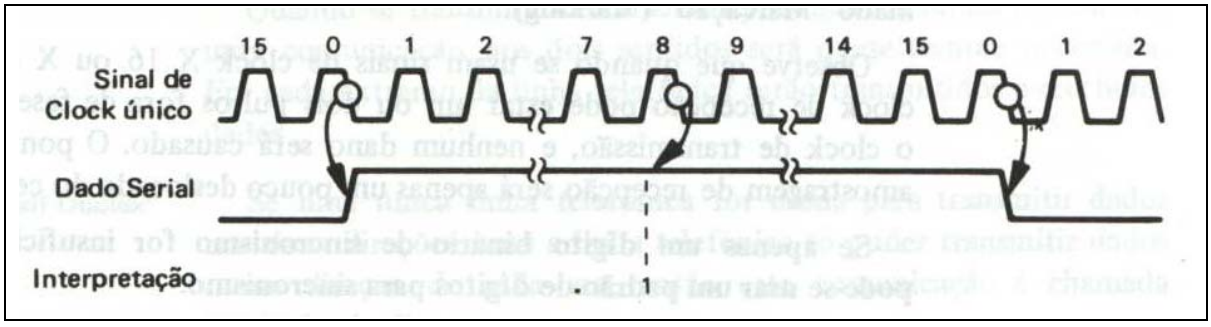
8.10.1 Sinais de *Clock*

Nosso microprocessador já possui um sinal de *clock* usado para temporizar a execução das instruções dentro da UCP. Não se deve confundir o *clock* do microprocessador com o *clock* de dados seriais, a única coisa que estes sinais têm em comum é que são ambos sinais de *clock*. O sinal de *clock* de dados seriais pode ou não ser derivado do sinal de *clock* do microprocessador.

Do ponto de vista de um usuário de um microprocessador, a velocidade é a diferença mais chamativa entre o *clock* do microprocessador e o clock de dados seriais. Um *clock* de microprocessador típico pode ter um período de 500ns (2MHz) ou menos, enquanto as taxas de transferência de dados seriais têm um alcance típico entre 110 a 9600 baud (110Hz a 9,6 KHz) é aproximadamente 200 vezes mais lenta que um clock típico de microprocessador.

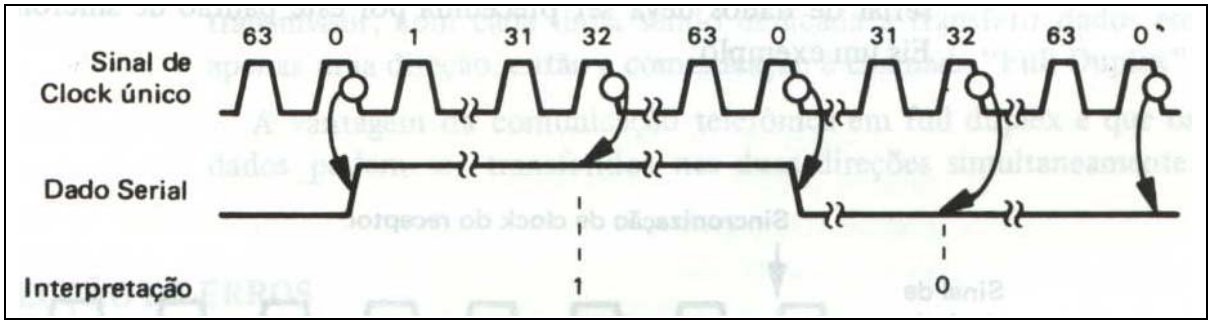
8.10.2 Sinal de *Clock* Serial X 1

O clock de E/S serial não necessita pulsar exatamente na taxa de baud, embora freqüentemente o faça:FIGURA – PAGINA 282Sinal de Clock Serial X 16
É bem comum que a taxa do clock de E/S serial seja 16 vezes a taxa de baud:



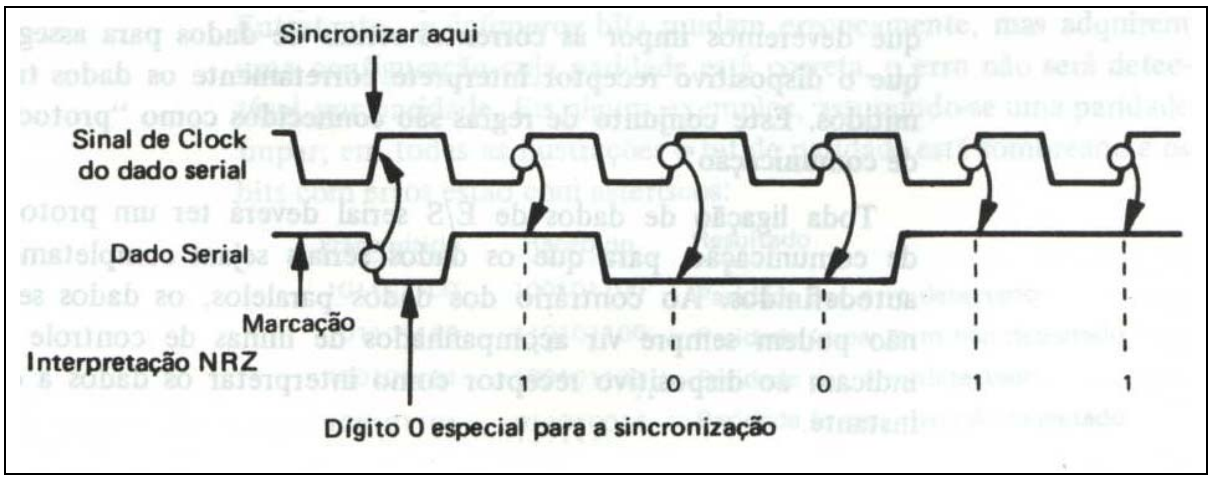
8.10.3 Sinal de Clock Serial X 64

Também é uma opção freqüente 64 vezes a taxa de baud:



A razão para termos clock X 16 e X 64 é para se chegar o mais rápido possível do centro do intervalo de tempo de duração de um único dígito ao amostrarmos um sinal de dado serial.

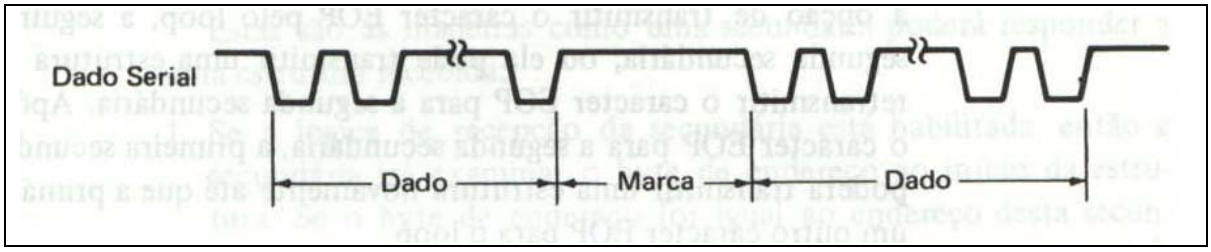
O fato do dado serial precisar ser acompanhado de um sinal de clock não significa necessariamente que toda E/S serial precise de duas linhas de sinal. O sinal de clock que acompanha, não tem que ser transmitido realmente numa linha acompanhante. Se configurarmos uma interface de comunicações de dados serial, com uma taxa de baud pré-definida, então a lógica do dispositivo receptor não precisa receber um sinal de clock. A lógica do dispositivo receptor pode criar o seu próprio sinal de clock local (interno) sincronizando-o com uma transição da linha de dados serial:



8.11 TRANSFERÊNCIA DE DADOS SERIAIS ASSÍNCRONOS

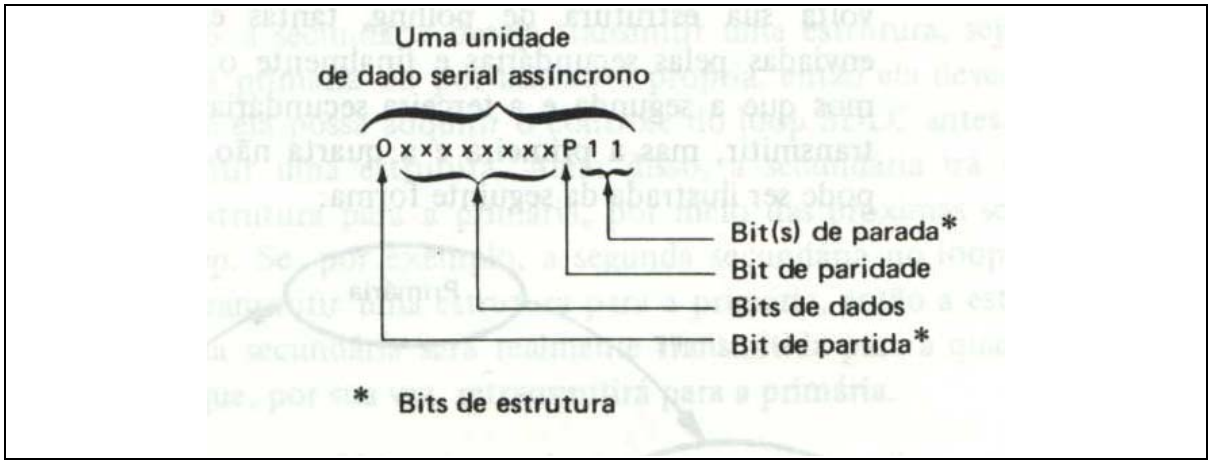
8.11.1 Sinal de Marca

Quando dados seriais estão sendo transferidos assincronamente, o dispositivo transmissor apenas transmite um caractere quando ele tiver um caractere pronto a transmitir. Entre caracteres, um sinal de marca contínuo (geralmente um nível alto) é enviado:



8.11.2 Estrutura

Todas as unidades de dados numa corrente de dados assíncrona devem levar sua própria informação de sincronismo. Uma unidade de dados assíncronos é portanto "estruturada" por um único bit de partida, em um, um e meio, ou dois bits de parada:

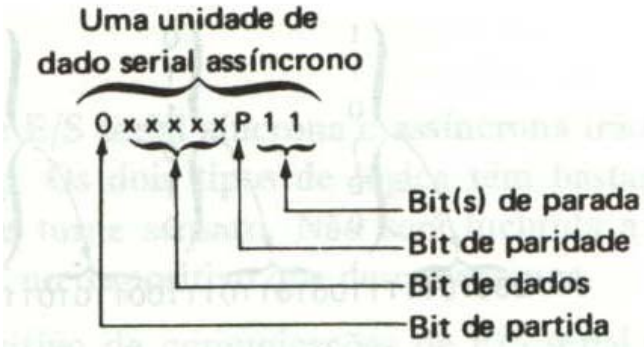


Ter um único bit de partida 0 é universalmente aceito no mundo dos microcomputadores. Existe uma semelhança entre os caracteres SYNC da corrente de dados síncrona, e os bits de estrutura de uma corrente de dados assíncrona.

Os caracteres SYNC estruturam um bloco de caracteres de dados síncronos. Os bits de partida e de parada estruturam cada caracter de dados numa corrente de dados assíncrona.

O protocolo assíncrono permite que um caractere tenha cinco, seis, sete ou oito bit de dados, assim como os dados seriais síncronos também. Por exemplo, se um protocolo estipular que só existirão cinco bits de dados em cada palavra assíncrona transmitida,

então o dispositivo receptor só receberá cinco bits de dados, e interpretará cada palavra recebida da seguinte maneira:

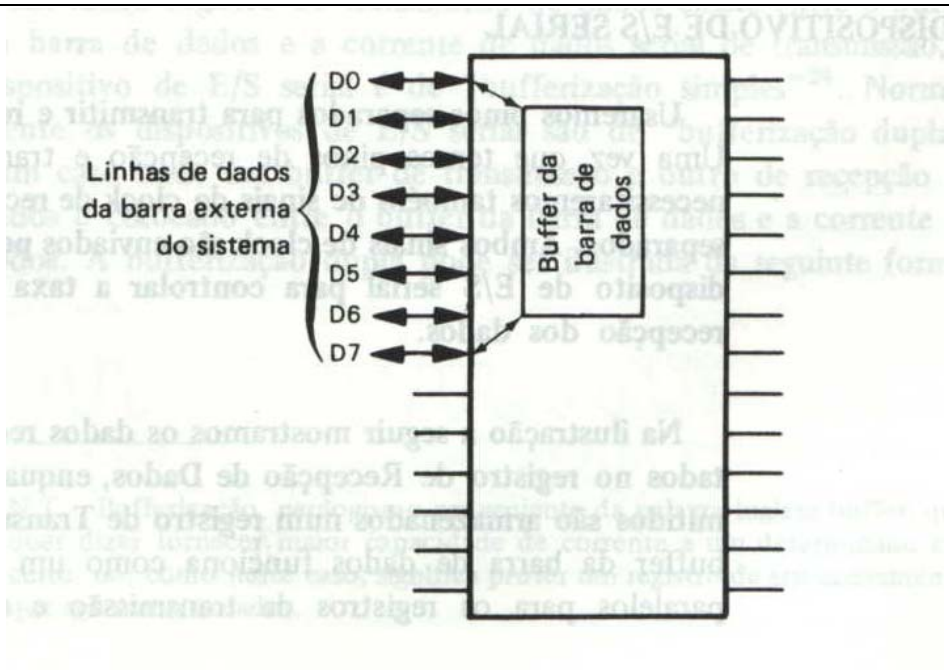


Logo, uma unidade de 9 bits será realmente transmitida.

8.12 INTERFACE MICROPROCESSADOR / DISPOSITIVO DE E/S SERIAL

Como a interface com o microprocessador é comum para as E/S síncrona e assíncrona, começaremos por ela.

O dispositivo de E/S serial irá se comunicar em paralelo com o microprocessador por meio das linhas de dados da barra externa do sistema. Devemos, portanto, prover oito pinos de dados, resguardados por um buffer da barra de dados:



Os outros sinais requiridos pela interface do microprocessador não são diferentes daqueles incluídos no dispositivo de E/S paralela. Usaremos o Select e o IORW. O *Select* é decodificado da barra de endereços para selecionar o dispositivo de E/S serial; IORW seleciona tanto uma “leitura para” o microprocessador quanto uma “escrita do”

microprocessador. Assumiremos arbitrariamente que o IORW alto especifica uma leitura, enquanto baixo especifica uma escrita.

Acrescentando o *clock*, alimentação e terra, o nosso dispositivo de E/S serial terá o seguinte aspecto: