Kernel

Este documento descreve as estruturas a serem gerenciadas pelo Kernel assim como as funções da API de acesso ao Kernel, que deverão ser implementadas no trabalho com o Computador CESAR16i.

Gerência do Visor

Seu kernel vai receber caracteres através das funções “putchar” e “putmsg”, e terá como responsabilidade coloca-los, corretamente, na posição do visor indicada nessas funções. Para a gestão do visor, seu kernel deve implementar a função “clr_visor”, que deve limpar, completamente, o visor. Limpar o visor significa preenche-lo caracteres SPACE (H20).

Quando o kernel passar para a execução da aplicação, o visor deve estar “limpo”. Ou seja, preenchido com caracteres SPACE (H20).

Gerência do Teclado

Seu kernel deverá ser capaz de informar para a aplicação que o usuário do programa digitou alguma tecla. Para isso, seu kernel deve implementar as funções “kbhit” e “getchar”. Notar que a função “getchar” não coloca os caracteres digitados no visor. A única função do “getchar” é devolver para a aplicação os caracteres digitados. Ainda, a função “kbhit” informa para a aplicação se algo foi digitado. Mas, esta função não retorna nenhuma tecla nem remove a tecla digitada.

Gerência do Timer

O kernel desenvolvido deverá dar suporte para um *timer*. A gestão desse timer será realizada pelas funções “clr_timer” (que zera o *timer*); “get_timer” (que fornece o valor atual do *timer*); “set_timer_on” (que faz o *timer* parar ou avançar); “get_timer_on” (que informa se o *timer* está parado ou avançando).

O *timer* deve ser implementado no kernel como um valor de 16 bits, sem sinal, que representa a passagem do tempo. O *timer* contém o valor em milissegundos (ms). O valor desse timer inicia em 0 e avança até 65535, quando deve retornar para 0. Portanto, o maior valor possível nesse timer corresponde a 65,535 segundos.

A resolução do timer deve ser de, no mínimo, 10 milisegundos. Portanto, por exemplo, se o temporizador do CESAR tiver sido programado para interromper a cada 10ms, o timer será somado com o calor 10, a cada interrupção. Outras possibilidades de valores estão apresentados na tabela abaixo.

Periodicidade da interrupção (ms)	Valor a ser somado no timer, a cada interrupção
10	10
5	5
2	2
1	1

Gerência de Velocidade

O seu kernel deve implementar a função “get_speed”, que tem por função fornecer a velocidade de deslocamento do veículo controlado pelo seu programa. Essa velocidade é fornecida em metros por segundo (m/s) e deve ser um valor entre 0 (zero) e 100 m/s (cem metros por segundo).

O controle da velocidade é feito pelo usuário, através das teclas “+” (que aumenta a velocidade) e “-“ (que diminui a velocidade). Essas teclas devem ser capturadas pelo seu kernel e usadas para aumentar ou diminuir a velocidade. Além disso, essas teclas jamais deverão ser retornadas pelas funções “kbhit” nem “getchar”.

Funções do Kernel

A seguir são descritas as funções da API e a forma como devem ser chamadas pelos programas de aplicação, incluindo os parâmetros de entrada e valores resultantes de saída.

Essas funções permitem que o programa de aplicação possa utilizar os periféricos disponíveis no CESAR16i (teclado, visor e *timer*), sem a necessidade de conhecer o funcionamento do hardware e das portas de acesso a esses periféricos.

As funções a serem implementadas deverão ser colocadas na área de memória reservada para o kernel.

Iniciando no endereço H0100, você deve definir a Tabela de Vetores do Kernel. Cada vetor (ponteiro com 2 bytes) dessa tabela deve conter o endereço, no kernel, onde inicia a implementação da função correspondente ao vetor.

Observar que a ordem desses vetores deve ser obedecida, rigorosamente. Os vetores e suas funções correspondentes estão indicados abaixo:

Vetor	Função
[0]	getchar
[1]	putchar
[2]	putmsg
[3]	clr_visor
[4]	Kbhit
[5]	get_timer
[6]	clr_timer
[7]	get_timer_on
[8]	set_timer_on
[9]	get_speed

A seguir, você encontra a descrição das funções a serem implementadas. No título de cada função está indicado o protótipo em "C" da função. Nesse protótipo o tipo "WORD" indica um valor com 16 bits sem sinal e "BYTE" indica um valor com 8 bits sem sinal.

0. Função: "getchar" – BYTE getchar(void)

Função através da qual a aplicação solicita ao kernel que informe a tecla digitada. Caso não tenha sido digitada uma tecla, a função deve aguardar até que seja digitada uma tecla (a função é "*bloqueante*"). Ao ser digitada uma tecla, a função deve retornar o código ASCII dessa tecla.

- *Parâmetros de entrada*: nenhum.
- *Parâmetro de saída*: registrador R0, com a tecla digitada.

A função só retorna (só termina) se houver uma tecla já digitada ou quando o usuário digitar alguma tecla. O código ASCII da tecla digitada deve ser retornado no registrador R0.

1. Função: "putchar" – void putchar(BYTE c, WORD posicao)

A aplicação usa essa função para solicitar que seja colocado um caractere em determinada posição do visor.

- *Parâmetros de entrada*:
  - Registrador R5, com o caractere a ser colocado no visor, codificado em ASCII. Só são aceitos valores entre H20 (SPACE) e H7A ("z").
  - Registrador R4, com a posição no visor onde colocar o caractere. Só são aceitos valores entre 0 (zero) e 35.

Caso algum dos parâmetros seja inválido, a função não deve apresentar qualquer informação no visor.

2. Função: "putmsg" – void putmsg(BYTE *msg, WORD posicao)

A aplicação usa essa função para solicitar que seja colocado no visor um string de caracteres (bytes) terminado por um byte H00 (o mesmo delimitador usado em string "C"). Os caracteres ASCII (visíveis e de controle) que formam o string devem ser tratados conforme definido na função "_putchar".

- *Parâmetros de entrada*:
  - Registrador R5, com o endereço de memória onde inicia o string.
  - Registrador R4, com a posição no visor onde iniciar a colocar o string. Só são aceitos valores entre 0 (zero) e 35.

3. Função: "clr_visor" – void clr_visor(void)

A aplicação vai chamar essa função sempre que desejar limpar o visor. O kernel deve preencher o visor com SPACE (H20).

Essa função não tem parâmetros de entrada nem de saída.

4. Função: "kbhit" – WORD kbhit(void)

Função através da qual a aplicação solicita ao kernel a informação da existência de alguma tecla digitada. A função deve retornar com a informação da existência de tecla. Essa função retorna imediatamente, sem aguardar pela digitação de qualquer tecla.

- *Parâmetro de saída*: registrador R0, com a informação da existência de tecla.

A função retorna no registrador R0 a informação se existe tecla ou não.

- **Se há tecla, o valor em R0 deverá ser zero**;
- Se não há tecla, o valor em R0 será um valor qualquer diferente de zero.

5. Função: "get_timer" – WORD get_timer(void)

A aplicação chama essa função sempre que desejar saber qual é o valor atual do timer. O valor retornado representa o tempo em milissegundos.

- *Parâmetro de saída*: registrador R0, com a informação do valor atual do timer.

6. Função: "clr_timer" – void clr_timer(void)

A aplicação chama essa função sempre que desejar ZERAR o valor do timer.

Essa função não possui parâmetros de entrada nem de saída.

7. Função: "get_timer_on" – WORD get_timer_on(void)

A aplicação chama essa função sempre que desejar saber se o timer está parado ou avançando.

- *Parâmetro de saída*: registrador R0, com a informação sobre o estado atual do timer (PARADO / AVANÇANDO).

Se o timer estiver PARADO, deve ser retornado o valor 0 (zero). Se o timer estiver avançando, deve ser retornado um valor diferente de 0 (zero).

8. Função: "set_timer_on" – void set_timer_on(WORD on)

A aplicação chama essa função sempre que desejar definir o estado do timer. Esse estado pode ser PARADO ou AVANÇANDO.

- *Parâmetros de entrada*: Registrador R5, com o estado a ser colocado no timer

Se o valor de R5 for 0 (zero), o timer deve ser PARADO. Se o valor de R5 for diferente de zero, o timer deve passar para o estado AVANÇANDO.

9. Função: “get_speed” – WORD get_speed(void)

A aplicação chama essa função para obter informações sobre a velocidade de deslocamento do veículo controlado pelo programa.

- *Parâmetro de saída*: registrador R0, com a informação atual da velocidade.

O valor retornado pode ser qualquer número entre 0 (zero) e 100, e representa a velocidade em metros por segundo.

A informação retornada por essa função representa o valor da velocidade controlado pelo usuário, através das teclas “+” e “-“.