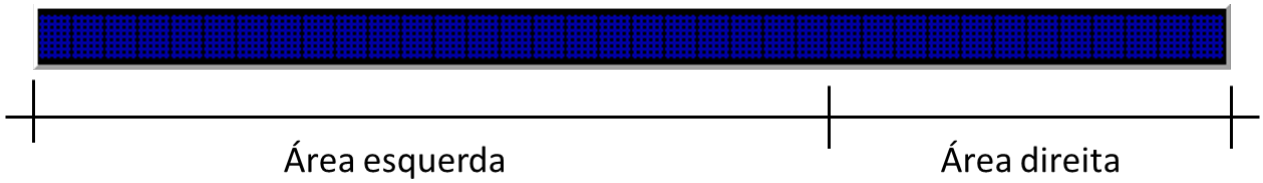
Kernel – 2023/1

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

O kernel deve gerenciar o visor de maneira que o usuário perceba duas áreas de apresentação de caracteres. A área esquerda é formada pelos 24 caracteres mais a esquerda do visor. A área direita é formada pelos 12 caracteres mais a direita do visor. Essa distribuição pode ser vista na figura abaixo:



Seu kernel vai receber caracteres para colocar na área esquerda e na área direita, através das funções “putchar” e “putmsg”. A função “puchar” permite colocar um caractere no visor enquanto que a função “putmsg” permite colocar um string no visor. Essas funções passam parâmetros para o kernel informando os caracteres a serem escritos e a área que deve recebe-los.

- Para colocar os caracteres na área da esquerda, seu kernel deverá manter a informação da posição de um cursor, que determinará onde colocar o caractere ou onde iniciar a colocar o string.
- Para colocar os caracteres na área da direita, seu kernel receberá a informação da posição onde colocar o caractere ou a posição onde iniciar a colocar o string, como um parâmetro adicional.

Caracteres visíveis (ASCII entre H20 e H7E) que ultrapassarem o tamanho da área deverão ser ignorados. Caracteres de controle devem ser interpretados e a função correspondente realizada.

Seu kernel deve apresentar no visor os caracteres ASCII visíveis (ASCII H20 até H7E). Além dos caracteres visíveis, seu kernel deve ser capaz de tratar alguns caracteres de controle. A definição dos caracteres de controle e suas ações estão definidos na descrição da função “putchar”.

Finalmente, quando o kernel terminar sua inicialização e passar à execução da aplicação, ambas as áreas devem ser limpas (preenchidas com caracteres SPACE ou 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 deverá implementar as funções “kbhit”, “getchar”, “startline” e “getline”.

- A função “kbhit” permitirá que a aplicação solicite ao kernel a informação se alguma tecla está disponível para leitura. Essa função apenas retorna esta informação, não ficando bloqueada aguardando ums tecla ser digitada.
- A função “getchar” tem operação semelhante ao “kbhit”, exceto que ela ficará bloqueada até que o usuário digite alguma tecla.

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.

As funções “startline” e “getline” operam em conjunto. A função “startline” coloca o kernel em um estado de espera pela entrada de um string pelo teclado e, opcionalmente, pode ser usada para encerrar a entrada do string. Durante esse estado do kernel, cada tecla digitada será colocada na posição do cursor, na área esquerda.

A função “getline” será usada pela aplicação para obter informações do kernel sobre a entrada do string. Através dessa função o kernek obterá as informações se a entrada do string terminou, qual é o string digitado e quanto tempo foi usado para isso.

Durante o período em que o kernel estiver no estado de entrada de um string as outras funções de acesso à área esquerda deverá estar bloqueadas (“putchar” e “putmsg”). Ou seja, a chamada dessas outras funções sobre a área esquerda será ignorada e devem retornar um código de erro. Entretanto, essas funções continuar operacionais para a área direita.

Funções do Kernel

A seguir são descritas as funções da API e a forma como serão chamadas pelos programas de aplicação, incluindo os parâmetros de entrada e valores a serem retornados como resultado.

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 memória, na área reservada para a tabela de vetores do kernel. Esses vetores devem ser colocados, adequadamente, na ordem definida neste documento. Cada vetor (ponteiro com 2 bytes) dessa tabela deve conter o endereço, no kernel, onde inicia a implementação da função correspondente ao vetor.

Assim, a ordem desses vetores deve ser obedecida rigorosamente. Os vetores, ordenados segundo seu número de ordem, e suas funções correspondentes estão indicados abaixo:

Vetor	Função
[0]	kbhit
[1]	getchar
[2]	putchar
[3]	putmsg
[4]	startline
[5]	getline

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 "kbhit"
WORD kbhit(void)

Função através da qual a aplicação solicita ao kernel a informação da disponibilidade 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 da existência de tecla digitada.

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

1. Função "getchar"
WORD 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 a tecla, a função deve retornar o código ASCII da mesma.

- *Parâmetros de entrada*: nenhum.
- *Parâmetro de saída*: registrador R0 com a tecla digitada ou código de erro
    - Se não houver erro, retorna a tecla digitada no byte menos significativa de R0. O byte mais significativo de R0 deve ser zero (H00).
    - Caso ocorra algum erro, a função deve retornar HFFFF.

A função só retorna (só termina) se houver uma tecla já digitada ou quando o usuário digitar alguma tecla ou se houver algum erro. O código ASCII da tecla digitada deve ser retornado na parte menos significativa do registrador R0.

2. Função “putchar”
WORD putchar(WORD area, BYTE caractere)
WORD putchar(WORD area, BYTE caractere, WORD pos)

A aplicação usa essa função para solicitar ao kernel que seja colocado um caractere na área esquerda ou na área direita do visor. Essa função admite duas formas de chamada, dependendo da área para a qual se destina o caractere. Além disso, essa função deve ignorar todos os caracteres ASCII de controle, exceto os listados a seguir, junto com os procedimentos esperados:

- A área esquerda deve ser capaz de tratar os caracteres ASCII de controle: BS (back space), CR (carriage return) e LF (line feed);
- A área direita deve ser capaz de tratar o caractere ASCII de controle FF (form feed).

A funcionalidade esperada para cada um destes caracteres de controle é a seguinte:

- O *back space* (ASCII H08) move o cursor uma posição para a esquerda. Caso o cursor esteja na posição mais a esquerda da área, o cursor não deve ser alterado;
- O *carriage return* (ASCII H0D) posiciona o cursos na posição mais a esquerda da área;
- O *line feed* (ASCII H0A) limpa a área (preenche com SPACE). A posição do cursor não deve ser alterado.
- O *form feed* (ASCII H0C) limpa a área (preenche com SPACE).

O parâmetro “área” identifica a área para a qual está destinado o caractere. Se o destino for a área esquerda, esse valor será “0” (zero); se o destino for a área direita, esse valor será diferente de zero.

O parâmetro “caractere” informa o caractere a ser escrito no visor. Só são aceitos valores entre H20 (SPACE) e H7A (“z”) e os caracteres de controle previstos para a área à qual se destinam.

Se o destino for a área esquerda, então o caractere será escrito na posição indicada pelo cursor e, então, incrementado. Quando o cursor chegar no final da área (quando o cursor sair da área), qualquer caractere visível adicional será ignorado, com exceção dos caracteres de controle, que devem continuar a serem interpretados.

Por outro lado, se o destino for a área direita, o caractere será escrito na posição indicada pelo parâmetro “pos”. Observar que o parâmetro “pos” indica a posição apenas na área da direita, sendo que o primeiro caractere da esquerda corresponde à posição “0” (zero). Portanto, só serão aceitos valores entre 0 (zero) e 11 (onze).

Se os parâmetros de entrada estiverem corretos e a função puder ser realizada corretamente, deverá ser retornado o valor “0” (zero). Caso a função não possa ser realizada por qualquer motivo, a chamada da função deve ser completamente ignorada e deve ser retornado um código de erro (qualquer valor diferente de zero).

- *Parâmetros de entrada*:
  - Registrador R5, com o parâmetro “área”;
  - Registrador R4, com o parâmetro “caractere”;
  - Registrador R3, com o parâmetro “pos”.
- *Parâmetro de saída*:
  - Registrador R0, com o código de retorno.

3. Função “putmsg”
WORD putmsg(WORD area, BYTE *msg)
WORD putmsg(WORD area, BYTE *msg, WORD pos)

A aplicação usa essa função para solicitar ao kernel que seja colocado um string na área esquerda ou na área direita do visor. Essa função admite duas formas de chamada, dependendo da área para a qual se destina o string. Além disso, essa função deve ignorar todos os caracteres ASCII de controle, exceto aqueles listados para a função “putchar”.

O parâmetro “área” identifica a área para a qual está destinado o caractere. Se o destino for a área esquerda, esse valor será “0” (zero); se o destino for a área direita, esse valor será diferente de zero.

O parâmetro “msg” informa o endereço na memória onde inicia o string a ser escrito no visor. O final do string é identificado pelo byte “\0” (ou H00). As restrições sobre os valores aceitos para os caracteres são as mesmas da função “putchar”.

Se o destino for a área esquerda, então o caractere será escrito na posição indicada pelo cursor. Por outro lado, se o destino for a área direita, o caractere será escrito na posição indicada pelo parâmetro “pos”. Observar que o parâmetro “pos” indica a posição apenas na área da direita, sendo que o primeiro caractere da esquerda corresponde à posição “0” (zero). Portanto, só serão aceitos valores entre 0 (zero) e 11 (onze).

Caso o string seja maior do que o espaço disponível na área de destino no visor, os caracteres visíveis adicionais (aqueles que ultrapassam o limite da área do visor) devem ser ignorados. Isso não se aplica aos caracteres de controle, que devem ser processados em qualquer lugar do string de entrada.

Caso seja encontrado no string algum caractere que não possa ser processado (não é um caractere visível nem caractere de controle aceito na área de destino), ele deverá ser ignorado e o processamento deve ser mantido para os caracteres restantes.

Se os parâmetros de entrada estiverem corretos e a função puder ser realizada corretamente, deverá ser retornado o valor “0” (zero). Caso a função não possa ser realizada por qualquer motivo, a chamada da função deve ser completamente ignorada e deve ser retornado um código de erro (qualquer valor diferente de zero).

- *Parâmetros de entrada*:
  - Registrador R5, com o parâmetro “área”;
  - Registrador R4, com o parâmetro “msg”;
  - Registrador R3, com o parâmetro “pos”.
- *Parâmetro de saída*:
  - Registrador R0, com o código de retorno.

4. Função “startline” WORD startline(BYTE *buffer, WORD max)

A aplicação usa essa função para informa ao kernel que deve ser iniciada a entrada de um string de caracteres fornecidos pelo usuário através do teclado. O kernel, após realizar os procedimentos necessários para possibilitar a entrada do string de teclas, deverá retornar para a aplicação.

A cada tecla digitada, o kernel deve colocar o caractere correspondente na posição do cursor, na área esquerda. Além disso, essa tecla deve ser salva na memória, a partir do endereço informado pelo parâmetro “buffer”.

Essa operação do kernel deve ser mantida até que ocorra uma das seguintes situações:

- O número de caracteres no buffer atingir o valor do parâmetro “max”;
- O usuário tenha digitado a tecla “ENTER” (código ASCII H0D);
- O programa de aplicação efetue uma nova chamada de “startline”. Nesse caso, encerra-se a operação em curso e se inicia nova operação de entrada de string;
- O programa de aplicação faz uma chamada de “getline” com “param = 0” (ver função “getline” a seguir).

Além do caractere de controle ENTER, usado para que o usuário informe que encerrou a entrada de teclas, o kernel deve tratar o caractere de controle BS (back-space) adequadamente. Ou seja, quando for digitado um “BS” no teclado, o cursor deve ser movido uma posição para a esquerda e o caractere que existia nessa posição deve ser apagado (escrever H20).

Observar que essa operação é diferente daquela prevista para o BS, quando enviado através de “putchar” ou “putmsg”. No caso dessas funções, o caractere não deve ser apagado.

Notar ainda que, enquanto o kernel estiver realizando a entrada do string de teclado, a aplicação poderá estar executando outras operações.

Portanto, durante o período em que o kernel estiver realizando a entrada do string, as funções “kbhit” e “getchar” serão ignoradas pelo kernel. Além disso, as funções “putchar” e “putmsg” dirigidas para a área esquerda do visor também deverão ser ignoradas pelo kernel. Essas duas funções só serão realizadas se forem dirigidas para a área direita do visor.

Caso a função tenha sido realizada corretamente, a função deve retornar em R0 o valor “zero”. Caso ocorra algum erro e a função não possa ser realizada, o kernel deverá retornar um código de retorno diferente de zero.

- *Parâmetros de entrada*:
  - Registrador R5, com o parâmetro “buffer”;
  - Registrador R4, com o parâmetro “max”;
- *Parâmetro de saída*:
  - Registrador R0, com o código de retorno.

5. Função “getline” WORD getline(WORD param)

A aplicação usa essa função para solicitar ao kernel informações sobre a entrada de strings de teclado, que pode ter sido iniciada por uma chamada da função “startline”, ou para controlar a entrada do string de teclado.

O parâmetro “param” identifica qual parâmetro o kernel deve informar ou o comando a ser executado. Os valores de “param” e a função a ser executada são os seguintes:

- PARAM_RESET (0): comanda o kernel para encerrar a entrada do string de teclado. Retorna:
  - 0 (zero), caso a operação tenha sido realizada corretamente;
  - 65535, caso o kernel não esteja com a entrada de string ativada ou outras situações de erro.
- PARAM_FINISHED (1): solicita ao kernel a informação sobre o término da entrada do string de teclado. Retorna:
  - 0 (zero), caso a entrada do string de teclado tenha terminado;
  - 65535, caso a entrada do string de teclado NÃO tenha terminado.
- PARAM_TIME (2): solicita ao kernel a informação de quanto tempo transcorreu desde a chamada da função “startline”.
  - O valor do tempo retornado deve ser informado em dezenas de milissegundos. Os valores possíveis de retorno vão desde 0 até o máximo de 65500, que corresponderá ao tempo de 655000 milissegundos (ou, aproximadamente, 10 minutos);
  - Caso a entrada do string de teclado já tenha encerrado, a função retorna o tempo deste a chamada da função “startline” até o encerramento da entrada do string de teclado;
  - Em outras situações não previstas, o valor retornado deve ser “0” (zero).
- PARAM_BUFFER (3): solicita ao kernel a informação do endereço de início do buffer de memória onde os caracteres estão sendo salvos. Retorna:
  - O endereço de início do buffer programado para salvar as teclas digitadas;
  - 65535, caso ocorra algum erro.
- PARAM_SIZE (4): solicita ao kernel a informação de quantos caracteres já foram teclados desde a chamada da função”startline”. Retorna:
  - Número de caracteres que existem no buffer de caracteres;
  - Caso a entrada do string de teclado já tenha encerrado, a função retorna o número de caracteres presentes no buffer no encerramento da entrada do string de teclado;
  - 65535, caso ocorra algum erro.
- Outros valores deste parâmetro devem ser ignorados e retornado código de erro 65535.

Se os parâmetros de entrada estiverem corretos e a função puder ser realizada corretamente, deverá ser retornado o valor previsto, conforme o valor do “param” usado na sua chamada. Caso a função não possa ser realizada por qualquer motivo, o kernel deve ignorar sua chamada e retornar o código de erro 65535.

- *Parâmetros de entrada*:
  - Registrador R5, com o parâmetro “param”;
- *Parâmetro de saída*:
  - Registrador R0, com o código de retorno (valor do parâmetro solicitado ou código de erro).