embedded-os.de
a little world of RTOS and data-communication protocols

pC/MEM Reference

V1.30a

Haftungsausschluß
Der Autor übernimmt keinerlei Haftung für durch diesen Code entstandene oder entstehende Schäden an Hard- und Software. Er versichert lediglich, daß er den Code vielfältigen Test's auf unterschiedlicher Hardware unterzogen hat, um seinerseits keine Fehler bestehen zu wissen. Sollten dennoch Fehler auftauchen oder Vorschläge zur Verbesserung des Codes an den Autor weitergegeben werden, so ist dieser bestrebt, Fehler schnellstmöglich auszumerzen oder Vorschläge einzuarbeiten.

liability exclusion
The author takes over no liability for through this code originated or emerging damages to hardware and software. He assures merely that he subjected the code of diverse tests on different hardware, about for his part no mistakes to know exists. Mistakes nevertheless should appear or suggestions are passed on at the author to the improvement of the code, so this is striving, mistakes fastest to wipe out or to incorporate suggestions.

Der Heap-Manager benötigt bei der Verwaltung von ≥64kB 2(3)bytes und bis 4GB 4(5)bytes Overhead je Eintrag (je nachdem ob `MEM_CleanUp` eingebunden wurde). Desweiteren kann er RAM und EE-PROM parallel verwalten und stellt für den EE-PROM Speicher über das Port HW-unabhängige Zugriffswerkzeuge zur Verfügung.

User-Functions:

Memory-Manager:	
MEM_Init	Initialisierung des Heap-Managers
MEM_Alloc	Allokieren von Speicher
MEM_Free	Freigeben eines allokierten Speichers
MEM_Resize	Größe von allokiertem Speicher ändern
Heap_Write_EE	auf allokiertem EEPROM-Heap schreiben
Heap_Fill_EE	allokierten EEPROM-Heap füllen
optional	
MEM_CleanUp	Freigeben aller von einen Task allokiertem Speicherelemente

Error-Codes:

Name	Decimal_Value	Description
MEM_NO_ERR	0	no error
MEM_WR_PTR	120	Freigeben eines allokierten Speichers
MEM_OVF	121	Pointer is not in the storage area or not allocated or no valid entry into this address
MEM_ERR	122	error in the memory management

---

allgemeines

MEM_Init

```
U08 MEM_Init(void)
```

Initialisiert den Heap-Manager. Sollte der EE-Heap als unformatiert erkannt werden, so wird dieses durchgeführt.
Diese Funktion muß vor allen anderen Heapdiensten bei der Systeminitialisierung einmal aufgerufen werden.
Bei Nutzung des Linux-HOSTs wird vor dem EE-Formattest versucht ein bestehendes Heap-IMAGE für die EE-Simulation aus einer Linux-Datei zu laden.

Parameters

none

Return Value

MEM_NO_ERR	erfolgreich initialisiert
MEM_ERR	Fehler innerhalb der Speicherverwaltung
MEM_INVALID	EE-IMAGE ungültig (nur bei Linux-HOST)

Example

```
void main(void)
{
    U08  returnOk;

    .
    returnOk=MEM_Init();
    .
    .
}
```

MEM_Flush

```
U08 MEM_Flush(void)
```

Nur bei Nutzung des Linux_HOSTs.
Sichert den EE-Heap als IMAGE in eine Linux-Datei.

Parameters

none

Return Value

MEM_NO_ERR	*erfolgreich initialisiert*
...	*...*

Example

```
void main(void)
{
    U08  returnOk;

    .
    returnOk=MEM_Init();
    .
    .
    returnOk=MEM_Flush();
}
```

MEM_Alloc

```
U08 OS_HUGE *MEM_Alloc(MEM_LONG size, U08 type)
```

Allokiert den angegebenen Speicher im geforderten Speichertyp (RAM/EE) und gibt die Startadresse zurück.

Parameters

size	size of array in bytes
type	type of Memory (0=RAM / 1=EE)

Return Value

Ist die zurückgegebene Addresse gleich NULL, so kann mittels MEM_GetErrno() nachfolgend der FehlerCode abgeholt werden.

MEM_OVF	Speicher voll
MEM_ERR	Fehler innerhalb der Speicherverwaltung

Example

```
void main(void)
{
    U08  returnOk;
    U08  OS_HUGE *ptr;
    .
    returnOk=MEM_Init();
    .
    ptr=MEM_Alloc(100, MEM_RAM);
    if(ptr==NULL) {
        returnOk=MEM_GetErrno();
    }
    .
    .
}
```

MEM_Free

```
U08 MEM_Free(U08 OS_HUGE *ptr)
```

Gibt den allokierten Speicher wieder frei. Der Speichertyp (RAM/EE) wird dabei selbst ermittelt. Zur Freigabe wird versucht diesen frei werdenen Speicherbereich an einen freien Speicher direkt dahinter und wenn vorhanden, an einen freien Speicher direkt davor, zu hängen. (Defragmentierung)

Parameters

*ptr	pointer of array (from MEM_Alloc)

Return Value

MEM_NO_ERR	Speicher freigegeben
MEM_WR_PTR	Zeiger nicht im Speicherbereich oder nicht allokiert oder kein gültiger Eintrag unter dieser Adresse
MEM_ERR	Fehler innerhalb der Speicherverwaltung

Example

```
void main(void)
{
    U08  returnOk;
    U08  OS_HUGE *ptr;
    .
    returnOk=MEM_Init();
    .
    ptr=MEM_Alloc(100, 0);
    if(ptr==NULL) {
        returnOk=MEM_GetErrno();
    } else {
        .
        .
        returnOk=MEM_Free(ptr);
    }
    .
    .
}
```

MEM_CleanUp

```
U08 MEM_CleanUp(U08 prio)
```

Gibt allen allokierten Speicher des angegebenen Tasks der Priorität `prio` wieder frei. Die jeweiligen Speichertypen (RAM/EE) werden dabei selbst ermittelt.

Parameters

prio	priority of task to free

Return Value

MEM_NO_ERR	Speicher freigegeben
OS_PRIO_INVALID	die Priorität ist größer OS_MAX_TASK
OS_TASK_SUSP_PRIO	unter dieser Priorität ist kein Task eingetragen
MEM_ERR	Fehler innerhalb der Speicherverwaltung

Example

```
void OS_TaskDelete(void)
{
    U08  returnOk, prio;

    OS_Lock();
    prio = OSTCBCur->OSTCBPrio;
    returnOk=MEM_CleanUp(prio);
    OS_ENTER_CRITICAL();
    OS_Unlock();
    .
    .
}
```

MEM_Resize

```
U08 OS_HUGE *MEM_Resize(U08 OS_HUGE *ptr, MEM_LONG newsize)
```

Vergrößert oder Verkleinert den allokierten Speicher und gibt die neue Startadresse zurück. Dabei wird beim Vergrößern versucht, einen möglichen freien Speicher direkt hinter diesem Eintrag, und wenn dieser noch nicht ausreicht, einen möglichen freien Speicher genau vor diesem Eintrag zu verwenden. (Defragmentierung)
Nur wenn dieses nicht ausreicht, wird mittels `MEM_Alloc` ein neuer Bereich allokiert und der alte nach erfolgtem Datenkopierens mittels `MEM_Free` freigegeben.

Parameters

*ptr	pointer of array (from MEM_Alloc)
newsize	new size of array in bytes

Return Value

Ist die zurückgegebene Addresse gleich NULL, so kann mittels MEM_GetErrno() nachfolgend der FehlerCode abgeholt werden.

MEM_OVF	Speicher voll
MEM_WR_PTR	Zeiger nicht im Speicherbereich oder nicht allokiert oder kein gültiger Eintrag unter dieser Adresse
MEM_ERR	Fehler innerhalb der Speicherverwaltung

Example

```
void main(void)
{
    U08  returnOk;
    U08  OS_HUGE *ptr;
    U08  OS_HUGE *ptr_new;
    .
    returnOk=MEM_Init();
    .
    ptr=MEM_Alloc(100, 0);
    if(ptr==NULL) {
        returnOk=MEM_GetErrno();
    } else {
        ptr_new=MEM_Resize(ptr, 150);
        if(ptr_new==NULL) {
            returnOk=MEM_GetErrno();
            .
            returnOk=MEM_Free(ptr);
        } else {
            .
            .
            returnOk=MEM_Free(ptr_new);
        }
    }
    .
    .
}
```

EE-Heap Access

Heap_Write_EE

```
U08 Heap_Write_EE(U08 OS_HUGE *source, U08 OS_HUGE *dest, MEM_LONG length)
```

Schreibt `length` Bytes von `*source` nach `*dest` ins EE-PROM unter Beachtung der HW-Spezifika.

Parameters

*source	source pointer (not in EE-PROM)
*dest	destination pointer into EE-PROM
length	bytes to write

Return Value

MEM_NO_ERR	Speicher geschrieben
MEM_WR_PTR	source-pointer zeigt ins EE-PROM
MEM_ERR	Fehler innerhalb der Speicherverwaltung

Example

```
void main(void)
{
    U08  returnOk;
    U16  var1;
    U08  OS_HUGE *ptr;
    .
    returnOk=MEM_Init();
    .
    ptr=MEM_Alloc(sizeof(U16), MEM_EE);          // alloc EE-PROM
    if(ptr==NULL) {
        returnOk=MEM_GetErrno();
    } else {
        var1 = 4125;
        returnOk=Heap_Write_EE(&var1, ptr, sizeof(U16));
    }
    .
    .
}
```

Heap_Fill_EE

```
U08 Heap_Fill_EE(U08 OS_HUGE *dest, MEM_LONG length, U08 value)
```

Füllt den angegebenen EE-PROM Bereich mit `value` unter Beachtung der HW-Spezifika.

Parameters

*dest	destination pointer into EE-PROM
length	bytes to fill
value	byte to fill with it

Return Value

MEM_NO_ERR	Speicher geschrieben
MEM_WR_PTR	dest-pointer zeigt nicht ins EE-PROM
MEM_ERR	Fehler innerhalb der Speicherverwaltung

Example

```
void main(void)
{
    U08  returnOk;
    U08  OS_HUGE *ptr;
    .
    returnOk=MEM_Init();
    .
    ptr=MEM_Alloc(20, MEM_EE);          // alloc EE-PROM
    if(ptr==NULL) {
        returnOk=MEM_GetErrno();
    } else {
        returnOk=Heap_Fill_EE(ptr, 20, 0x55);
        .
        .
    }
    .
}
```

Comments

Comments

Comments