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

pC/LZFG Reference

V1.20a

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.

Die meisten Entpacker benötigen reichlich (bis zu 256kB) RAM um z.B. über die Huffman-Table die Daten wieder zu entpacken oder die Packrate erreicht nicht einmal 20%.
Das LZFG-Decompressor Modul benötigt gerade einmal 4kB RAM, die in Laufzeit dynamisch allokiert werden. Desweiteren ist der Code sehr kompakt und der Packer erreicht erstaunliche 40..55% beim Packen einer Applikation. Als Übergabeparameter benötigt der Decompressor die Adresse des gepackten Datensatzes, die Länge dieses Datensatzes sowie zwei Zeiger für Adresse entpackter Datensatz (wird intern dynamisch allokiert) und Länge des Selben. Innerhalb des Modules wird geprüft, ob es sich um einen kompatiblen LZFG-Datensatz handelt und ob die Länge und die CRC16 der entpackten Daten identisch den im LZFG-Datensatz hinterlegten Werten sind.

User-Functions:

LZFG-Decompressor:	
LZFG_Decomp	Dekomprimiert den Datensatz

Error-Codes:

Name	Decimal_Value	Description
LZFG_SUCCESS	0	decompress successfully
LZFG_noLZFG	130	Data are not in the necessary format
LZFG_lenERR	131	length error
LZFG_crcERR	132	CRC error
LZFG_memERR	133	error in allocation of the window or destination memory

pC/LZFG - Decompressor

LZFG_Decomp

```
U08 LZFG_Decomp(U08 OS_HUGE *source, U32 source_l, U08 OS_HUGE **dest_pp, U32 *dest_l)
```

Dekomprimiert den übergebenen Datensatz nach dem LZFG-Verfahren inclusive Prüfung der Erweiterungen CRC-16 und Länge des Originals.

Parameters

*source	pointer to compressed data
source_l	lenght of compressed data
**dest_pp	pointer to get the position of decompressed data
*dest_l	pointer to get the lenght of decompressed data

Return Value

LZFG_SUCCESS	erfolgreich dekomprimiert
LZFG_noLZFG	Daten sind nicht im erforderlichen Format
LZFG_lenERR	Längenfehler
LZFG_crcERR	CRC-Fehler
LZFG_memERR	Fehler bei Allokation des Window- oder Zielspeichers

Example

```
void OS_FAR Task2(void *data)
{
    U08  OS_HUGE *source_p;
    U08  OS_HUGE *decomp_p;

    U08  state;
    U32  rx_lenght;
    U32  decomp_lenght;

    .
    .
    while(1)
    {
        .
        state=OS_MemAlloc(&source_p, 46000);
        if(state == OS_NO_ERR)
        {
            rx_lenght=UART1_receive(source_p, .........
            .
            state=LZFG_Decomp(source_p, rx_lenght, &decomp_p, &decomp_lenght);
            if(state == LZFG_SUCCESS)
            {
                .
                state=FLSH_Update(decomp_p, decomp_lenght ..
                .
                .
                state=OS_MemFree((OS_MEM OS_HUGE *) decomp_p);
                .
```

```
            }
            .
            state=OS_MemFree((OS_MEM OS_HUGE *) source_p);
        }
        .
    }
}
```

Comments

Comments

Comments