# MoSynC

## Resource Compiler

Version 0.5

28/07/2006

Authors:

Antony Hartley

Frederik Eldh

# Contents

## 1 Disclaimer

This document is intended for internal use by Mobile Sorcery AB.
If for some reason you are external to Mobile Sorcery, be warned that the information in this document my change or may not be up to date.

## 2 General

A MoSynC application have three distinct sections: code , data and resource objects. This document contains information about creating resource objects.

The resource compiler is invoked on the command line or within a make environment. See buildres.bat for example usage.

## 3 Resource Basics

### 3.1 General

The resource list file is a standard text files that contain commands. These are described below.

A single resource object is typically structured like this:

| | |
|---|---|
| .res<name> | *New resource* |
| .sprite parameters | *(or .bin / .image / .sound etc)* |
| ... | |

The next .res command will automatically close the previous resource and start a new resource. Resource files are terminated with a .end command.

### 3.2 Resource order (numbering)

Resources are numbered incrementally, starting at 1. Resource ID 0 is reserved.

### 3.3 Output files

The resource compiler outputs two files. Firstly, the binary file of the resources. Secondly, a header file which should be included in the C program. This file contains #defines that symbolically link the C program to its resources.

### 3.4 Comments

Standard C/C++ comments are allowed in the resource file.

Examples:

```
// C++ Line comments work
/* C style comments too */
```

# 4 Resource Commands

## 4.1 .res

**Syntax**: .res <symbol>

**Description**:
Initializes a new resource with the symbol name of <symbol>, although a symbol need not be defined.

**Example**:

```
.res myimage
.image “myimage.png”
```

## 4.2 .image

**Syntax:** .image <imagefile>

**Description**:

Declares a resource as an image and then loads and stores a PNG image file into a resource.

**example**:

```
.image “myimage.png”                    (centerpoint is 0,0)
```

## 4.3 .bin

**Syntax:** .bin

**Description:**

Declares a resource  as binary. A binary resource is created and has 0 length.

**Example:**

```
.bin
```

## 4.4 .ubin

**Syntax:** .ubin

**Description:**

Declares a resource as unloaded binary. A binary resource is created and has 0 length.
Note: At runtime this resource will not be memory resident, but is accessed from the file system directly.

**Example:**
.ubin

## 4.5 .media

**Syntax:** .media “MIME type string”, “MediaFile”

**Description:**

Creates a resource for media files, these may be any type, these are of course platform specific, so a specific device may or may not support 'mp3' for example.

**Example:**
.media “audio/mp3”, “mysound.mp3”

## 4.6 .umedia

**Syntax:** .umedia “MIME type string”, “MediaFile”

**Description:**

Creates a unloaded resource for media files, these may be any type, these are of course platform specific, so a specific device may or may not support 'mp3' for example.

Note: At runtime this resource will not be memory resident, but is accessed from the file system directly.

**Example:**
.umedia “audio/mp3”, “mysound.mp3”

## 4.7 .sprite

**Syntax**: .sprite image_reference , start_x, start_y, size_x, size_y

**Description**:
declares a resource as a sprite object, it requires an image reference for a previously loaded image, the sprite is cut out from this image at start_x & start_y the size of the sprite is defined by size_x & size_y.

**Example**:

```
.res image1                          (load image)
.image "myimage.png"

.res.sprite  image1, 0,0, 10,10          (cut out a sprite from 0,0 to 10,10)
```

## 4.8 .tileset

**Syntax**: tileset "tileset.png", tilesize_x, tilesize_y

**Description**: Declares a tileset image, the images contains tiles of the specified tilesize.

**Example**:
.tileset "mytiles.png", 16, 16

## 4.9 .tilemap

**Syntax**: tileset "tilemap.bin", mapsize_x, mapsize_y

**Description**: Declares a tilemap, the tilemap binary file contains mapsize_x*mapsize_y 16bit indices's that refer to a tileset. Although the actual connection between tilemap and tileset is created at runtime.

**Example**:
.tilemap "mytilemap.bin", 64, 64

## 4.10 .dispose

**Syntax**: .dispose

**Description**:
Marks a resource as disposable, when the resource loader has finished loading all resources, it deletes all those resources marked for disposal.

**Example**:

```
.res image1                          (load image)
.dispose                             (force image to dispose after loading)
.image "myimage.png"
```

## 4.11 .placeholder

**Syntax**: .placeholder

**Description**:
Creates an empty resource that can be filled with something at runtime.

**Example**:
```
.res myspace
.placeholder
```

## 4.12 .skip

**Syntax**: .skip

**Description**:
Skips a resource when loading, this is used when loading new resources over the top of existing resources.

**Example**:
```
.skip
```

## 4.13 .Label

**Syntax**: .label “name”

**Description**:
Creates a marker label resource entry, so the application can search for the resource symbolically
at runtime, this allows libraries to find there resouces.

## 4.14 .enum

**Syntax**: .enum { var <=expression><,>}

**Description**:
Define an enumerated set of variables that can be used in expressions.

**Example:**
```
.enum
{
    abc                         =                         0,
    xyz,          (xyz                                   =1)
    qrs                         =                         99
}
```

## 4.15 .string

**Syntax:** .string “string”

**Description**:
Used only in binary resources to insert ASCII strings.
Note: These are nonull terminated strings.

**Example**:
.string “hullo”

## 4.16 .cstring

**Syntax:** .cstring “string”

**Description**:
Used only in binary resources to insert ASCII null terminated strings.

**Example**:
.cstring “hullo”

## 4.17 .pstring

**Syntax:** .pstring “string”

**Description**:
Used only in binary resources to insert pascal strings.

**Example**:
.pstring “hullo”

## 4.18 .fill

**Syntax**: fill size, filler

**Description**:
Used only in binary resources, this command fills the resource with size bytes of the filler byte. The data will be inserted at the current data position.

E**xample**:
.fill 8, '?' (Insert '?' 8 times)

## 4.19 .byte

**Syntax**: .byte n1<,n2,...>

**Description**:
Used only in binary resources, this command inserts bytes into the resource.
The data will be inserted at the current data position.

**Example**:

```
.byte 1,2,3,4                  (writes 1,2,3,4 to binary resource)
```

---

## 4.20 .half

**Syntax**: .half  n1<,n2,...>

**Description**:
Used only in binary resources, this command inserts half words (16 bits) into the resource.
The data will be inserted at the current data position.

**Example**:

```
.half 1,2,3,4                  (writes shorts 1,2,3,4 to binary resource)
```

---

## 4.21 .word

**Syntax**: .word  n1<,n2,...>

**Description**:
Used only in binary resources, this command inserts words (32 bits) into the resource.
The data will be inserted at the current data position.

**Example**:

```
.half 1,2,3,4                  (writes ints 1,2,3,4 to binary resource)
```

---

## 4.22 .include

**Syntax**: ..include “file”

**Description**:
Used only in binary resources, this command inserts a binary file into the resource.
The data will be inserted at the current data position.

**Example**:

```
.include “test.bin”
```

---

### 4.23 .extension

**Syntax**:

**Description**:
Reserved type used by mobile sorcery.

**example**:

---

### 4.24 .varint

**Syntax**:

**Description**:
Reserved type used by mobile sorcery.

**example**:

---

### 4.25 .varsint

**Syntax**:

**Description**:
Reserved type used by mobile sorcery.

**example**:

---

### 4.26 .end

**Syntax**: .end

**Description**:
Marks the end of a resource file.

**example**:
.end

---

### 4.27 .eof

**Syntax**: .end

**Description**:
Reserved type used by mobile sorcery.

---

### 4.28.index

**Syntax**: .index symbol_name

**Description**:
Used to create indexed resources.
Adds an index inside binary resources, so a single resource can contain sub-indices's.

A resource with indices will contain an index table, which can be read by the users program code with the resource index reading functions.

**example**:
.index “MySym”

---

### 4.29 .wideindex

**Syntax**: .wideindex

**Description**:
Forces a indexed resource to use 32 bit indices's, so an index table may contain data pointers greater than 64K.

**example**:
.wideindex

---

### 4.30 .end

**Syntax**: .end

**Description**:
Marks the end of a resource file.

**example**:
.end

---

### 4.31 .set

**Syntax**: .set variable = <expression>

**Description**:
Sets a script variable with the value of expression.

**example**:
.set hello = 1

---

### 4.32 Other examples

<u>`Create an Image resource`</u>

```
.res image1
.dispose
```

```
.image "testimage.png", 99, 99
```

Create an File resource

```
.res afile
.bin
.include "testfile.dat"
```

Create an sprite resource

```
.res sprite1
.sprite image1, /* XY */ 0, 0, 10, 10,  5,  5
```

Create an binary resource

```
.res                        // note resource has no name
.bin                        // say binary
.string "The buck"
.fill 8, '?'             // fill memory
.string "stops here!!!"        // strings with esc codes
.byte 1,2,3,4               // bytes
.half 5,6,7,8               // shorts
.word 9,10,11,12            // ints
.include "randomdata.bin"   // Include file
```