in

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>in | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | June 25, 2022 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# in

## 1.1 C functions...

```
                    C functions...

    The following functions will help enormously when producing your own
effects.


struct EP_Port {
    struct MsgPort *our_mp;
    struct Ext_Proc_Msg message;
    ULONG button;
    BOOL lock;
    };

typedef struct EP_Port * EPP_HANDLE;

EPP_HANDLE
                EP_NewPort
                  (ULONG);
void
                EP_DeletePort
                  (EPP_HANDLE);
BOOL
                EP_SendMsg
                  (EPP_HANDLE, ULONG, void *);
BOOL
                EP_SendIDMsg
                  (EPP_HANDLE, ULONG, struct Task *, ULONG);
BOOL
                EP_Quit
                  (EPP_HANDLE);
BYTE
                EP_AllocQSig
                  (EPP_HANDLE);
void
                EP_DeallocQSig
                  (BYTE);
BOOL
                EP_Lock
```

```
                  (EPP_HANDLE);
BOOL
                EP_Unlock
                  (EPP_HANDLE);
BOOL
                EP_GetSample
                  (EPP_HANDLE, struct sam_info **, struct sam_info **, ULONG *);

void
                Set_Level
                  (BYTE *, ULONG, LONG, ULONG, UBYTE);
WORD
                Get_Level
                  (BYTE *, ULONG, ULONG, UBYTE);
ULONG
                Sams_To_Bytes
                  (ULONG, ULONG);
ULONG
                Bytes_To_Sams
                  (ULONG, ULONG);
void
                CopyWave
                  (BYTE *, ULONG, BYTE *, ULONG, ULONG);
void
                set#_lev
                  (BYTE *, ULONG, LONG, UBYTE);
LONG
                get#_lev
                  (BYTE *, ULONG, UBYTE);

#define
                GET_LEVEL
                (start, pos, type, chan) (*Get_Level_Function[(type)])((start), ( ↩
                   pos), (chan))
#define
                SET_LEVELN
                (start, pos, lev, type, chan) (*Set_Level_Function[(type)])((start ↩
                   ), (pos), (lev), (chan))
#define
                SET_LEVEL
                (start, pos, lev, type, chan) LONG macro_level = lev;\
        if (macro_level > 32767) macro_level = 32767;\
        else if (macro_level < -32768) macro_level = -32768;\
        (*Set_Level_Function[(type)])((start), (pos), macro_level, (chan))
```

## 1.2 epp_han = EP_NewPort (button_num)

```
    NAME
        EP_NewPort -- Create a new message port.

    SYNOPSIS
        epp_han = EP_NewPort (button_num);

        EPP_HANDLE = EP_NewPort (ULONG);
```

```
FUNCTION
    Create a new message port and initialise. An EPP_HANDLE is returned
    for use with other EP_ functions. The EP_ functions take care of many
    things, for example your button number and whether you hold the lock.

INPUTS
    Your button number (ULONG).

RESULT
    A new port will be created and a handle returned if sucessful else
    NULL.

SEE ALSO
    EP_#(); EP_DeletePort().
```

## 1.3 EP_DeletePort (epp_han)

```
NAME
    EP_DeletePort -- Delete a message port.

SYNOPSIS
    EP_DeletePort (epp_han);

    void EP_DeletePort (EPP_HANDLE);

FUNCTION
    Deletes a previously created port, created using EP_NewPort. This may
    be performed on a NULL pointer. The EPP_HANDLE passed will then be
    invalid.

INPUTS
    EPP_HANDLE as returned by EP_NewPort().

RESULT
    The port will be deallocated as well as the memory it occupied.

SEE ALSO
    EP_NewPort().
```

## 1.4 result = EP_SendMsg (epp_han, command, data)

```
NAME
    EP_SendMsg -- Send a command to SamEd.

SYNOPSIS
    result = EP_SendMsg (epp_han, command, data);

    BOOL = EP_SendMsg (EPP_HANDLE, ULONG, void *);

FUNCTION
    Sends a message to SamEd, with the command, and any relevent data.
```

```
        You must check the result and check for errors, ie. once sent check
        epp_han->message->epm_Error.

    INPUTS
        EPP_HANDLE returned by EP_NewPort();
        ULONG command (EPC_#);
        void * pointer to command specific data or NULL;

    RESULT
        TRUE if the message was sent, then check epp_han->message->epm_Error
        for errors, (= NULL for no error).

    SEE ALSO
        EP_NewPort(); EP_SendIDMsg.
```

## 1.5 "

```
    NAME
    SYNOPSIS
    FUNCTION
    INPUTS
    RESULT
    BUGS
    SEE ALSO
```

## 1.6 "

```
    NAME
    SYNOPSIS
    FUNCTION
    INPUTS
    RESULT
    BUGS
    SEE ALSO
```

## 1.7 "

```
    NAME
    SYNOPSIS
    FUNCTION
    INPUTS
    RESULT
    BUGS
    SEE ALSO
```

## 1.8 "

```
NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO
```

## 1.9 "

```
NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO
```

## 1.10 "

```
NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO
```

## 1.11 result = EP_GetSample (epp_han, &sample_ptr, &copybuf_ptr, &ranges)

```
NAME
    EP_GetSample -- Get pointers to sam_info structs, and range data.

SYNOPSIS
    result = EP_GetSample (epp_han, &sample_ptr, &copybuf_ptr, &ranges);

    BOOL = EP_GetSample (EPP_HANDLE, struct sam_info **, struct sam_info **,  ↩
       ULONG *);

FUNCTION
    Sets given pointers to point at sample data. &ranges should be an
    array of 2 ULONGs. If a pointer is not given then that data is not
    retrieved. If you did not have the lock then it will be temporarily
    aquired. With no lock the data is read only.

INPUTS
    epp_han         = EPP_HANDLE returned by EP_NewPort();
    &sample_ptr     = address of a pointer to a sam_info struct.
    &copybuf_ptr    = address of a pointer to a sam_info struct for the
```

```
                          copy buffer.
    &ranges         = address of 2 ULONG's for range start & end.

RESULT
    result only returns TRUE when all data was retrieved. If only some
    was retrieved then the 'failed pointers' = NULL, other pointers are
    valid.

NOTES
    The data from SamEd is not copied, just pointers to that data. The
    actual range data is copied.

SEE ALSO
    EP_NewPort().
```

## 1.12 "

```
NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO
```

## 1.13 "

```
NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO
```

## 1.14 "

```
NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO
```

## 1.15 "

```
NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO
```

## 1.16 CopyWave (from, from_type, to, to_type, length)

```
NAME
    CopyWave -- Copy waveform data, and change type.

SYNOPSIS
    void = CopyWave (BYTE *, ULONG, BYTE *, ULONG, ULONG);

    CopyWave (from, from_type, to, to_type, length);

FUNCTION
    Copy waveform data, and change type. Stereo samples may be mixed to
    mono (using mean average), mono samples will be expanded to stereo.

INPUTS
    from        - pointer to source data.
    from_type   - source type.
    to          - pointer to a destination buffer.
    to_type     - destination type.
    length      - length of copy in sample frames.

RESULT
    Copied wave data, in new sample format.

NOTES
    Make sure the destination buffer is large enough. Use Sams_To_Bytes
    etc.

SEE ALSO
    Sams_To_Bytes(); Bytes_To_Sams().
```

## 1.17 "

```
NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO
```

## 1.18 "

NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO

## 1.19 "

NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO

## 1.20 "

NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO

## 1.21 "

NAME
SYNOPSIS
FUNCTION
INPUTS
RESULT
BUGS
SEE ALSO