RawDIC

COLLABORATORS			
	TITLE : RawDIC		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 10, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

RawDIC

1.1 RawDIC documentation (14th Jun.99)

```
                RawDIC V1.7
© 1999 by John Selck
```


1.2 Disclaimer

This program is freely distributable unless no changes
are made to the archive.

The author is not liable for any damage/problems/loss of data
this program might directly or indirectly cause.

1.3 About RawDIC

RawDIC is a support tool for creating disk imagers and use them in ↩
own
installers. To create a disk image, a small file which holds basic
information about the disk layout and the MFM decoding must be created.
This file is very similar to the WHDLoad slaves, but they feature less code.
I will call these files "ISlaves" (Imager-Slaves).

Unlike the WHDLoad slaves, ISlaves are not that code-based. Much information
is provided in
structures
and
lists
. In most cases the only code featured
in an ISlave is the MFM decoder routine, sometimes you even don't need that
one.

1.4 Requirements

In most cases you will need an Amiga with atleast Kickstart 2.0 and
2 MB memory. (Depends on the ISlave and the size of the diskimage)

The memory requirements will be decreased in future versions.

1.5 Contact

E-Mail: graham@cruise.de

1.6 ToolTypes

For external control of RawDIC there are some ToolTypes defined:

SLAVE={filename} Defines the path where the ISlave can be found.

SOURCE={path} Defines the source of the MFM data, currently
only DF0: to DF3: are supported.

RETRIES={amount} The number of times RawDIC will try to read a
damaged track. When all retries failed, the

```
      error requester will appear.

IGNOREERRORS    Let's RawDIC continue to create the image even
      under error conditions. Sometimes this might
      be useful to save old damaged disks.
      Please try a high number of retries first!
```

1.7 Structures

```
                There are two different structures which are needed by the ISlaves ↩
                    .



The first one is the ISlave-header structure. This structure always follows
the slave header and is built up like this:

  UBYTE slv_Version
  UBYTE slv_Flags
  APTR  slv_FirstDisk
  APTR  slv_Text

slv_Version:  The version of the ISlave-header structure, since only V1.?
    of RawDIC exists, slv_Version also is 1.

slv_Flags:  The global flags for the ISlave. Currently only
                SFLG_DEBUG
                    is defined to enable RawDIC's debug output.

slv_FirstDisk:  Pointer to the first Disk-structure. Every ISlave must have
    atleast one Disk structure.

slv_Text: Pointer to a C-string which contains the text displayed in
    the RawDIC window (i.e. "XYZ-imager by Hans Wurst").

Example:

  dc.b  1,0 ; Version 1, no flags set.
  dc.l  DSK_1 ; First Disk-structure at label "DSK_1".
  dc.l  Text  ; ... at label "Text".



Now, for every disk of the game to be installed, you need to define a Disk-
structure. A 3-disk game will have 3 Disk-structures. A Disk-structure
contains no information about the disk layout itself, but defines the ISlaves
behaviour when reading a disk.

  APTR  dsk_NextDisk
  UWORD dsk_Version
  UWORD dsk_Flags
  APTR  dsk_TrackList
  APTR  dsk_TLExtension
  APTR  dsk_FileList
```

```
  APTR  dsk_CRCList
  APTR  dsk_AltDisk
  FPTR  dsk_InitCode
  FPTR  dsk_DiskCode

dsk_NextDisk: Pointer to the next Disk-structure or 0 when this is the last
    disk.

dsk_Version:  Version number of the Disk-structure, currently only 1 is
    supported.

dsk_Flags:
                Flags
                 for the disk.

dsk_TrackList:  Pointer to a
                TrackList
                .

dsk_TLExtension:Reserved for future use!

dsk_FileList: Pointer to a
                FileList
                . There are a few pre-
    defined FileLists: FL_DISKIMAGE to save the diskimage under
    the name "Disk.#" and FL_NULL to save no files at all.

dsk_CRCList:  Pointer to a
                CRCList
                . Needed for version
    checks on disks when different versions of a game have a
    different disk layout. 0 when no CRC check shall be done.

dsk_AltDisk:  This is a pointer to an alternative disk structure when the
    CRC check on the disk returned FALSE. 0 when no alternative
    Disk-structure is present.

dsk_InitCode: Pointer to code which is called BEFORE the disk image is
    created. Put your initialisations here. 0 when no InitCode
    needed.

dsk_DiskCode: Pointer to code which is called AFTER the disk image is
    created. Here you may save files. 0 when no DiskCode is
    needed.

Example:

  dc.l  0   ; This is the last disk.
  dc.w  1   ; Disk-structure version 1.
  dc.w  SFLG_SINGLESIDE ; The disk is single-sided.
  dc.l  TL_1    ; Track layout at label "TL_1".
  dc.l  0   ; Unused.
  dc.l  FL_DISKIMAGE  ; Create diskimage.
  dc.l  0   ; No CRC check.
  dc.l  0   ; No CRC check => no alternative disk.
  dc.l  0   ; No initialisation code.
  dc.l  0   ; No code.
```

1.8 Lists

Since it is not very senseful to define everything in structures
, there are
a few tables which might hold information about the disk and the ISlaves behaviour.

```
#################
###       ###
### TrackList ###
###       ###
#################
```

The most important table is the TrackList. I won't describe the structures of the single TrackList entry, there are macros defined which keep you as the ISlave programmer away from them:

TLENTRY: A macro for a TrackList entry. Such an entry contains of a starttrack, an endtrack, a tracklength (the length of the decoded trackdata), a sync signal and finally a pointer to a routine which does the MFM to raw conversion.

Usage:

```
TLENTRY firsttrack,lasttrack,length,sync,decoder
```

TLEND: This terminates a TrackList so RawDIC will know that no more TLENTRYs follow. TLEND has no parameters.

Usage:

```
TLEND
```

Example for a TrackList:

```
  TLENTRY 0,19,$1600,SYNC_STD,DMFM_STD
  TLENTRY 40,159,$1800,$4A84,DMFM_CUSTOM
  TLEND
```

Pre-defined values to use for TLENTRY:

SYNC_INDEX: The track will be read with indexsync only.
SYNC_STD: Syncword $4489.

DMFM_NULL: The diskimage will contain zeros on this track.
DMFM_STD: Standard Amiga track decoder, will automatically decode tracklength/512 sectors.

Please notice that DMFM_CUSTOM is a routine YOU must add to the ISlave. (Look at the code examples.)

```
RESTRICTIONS: RawDIC will check your TrackLists if they follow these
    restricions:

    1. Tracks with a higher track number never come
       before tracks with a lower track number.

    2. Tracks only appear once in the TrackList.

    3. The first track of a TrackList entry is always
       lower/same than the last track of the entry.

    To disable the check for these restrictions, use the
    DFLG_NORESTRICTIONS flag.



################
###      ###
### FileList ###
###      ###
################

There is another quite important list used for ISlaves, it's the FileList
which describes the files to be saved by RawDIC. Since in most cases this
list only is used to save a diskimage or to save no files (since the
files are saved in dsk_DiskCode), there are two pre-defined filelist:

FL_NULL:  Save no files automatically.

FL_DISKIMAGE: Save diskimage with the name "Disk.#".

In case you still have a very static disk layout (no directories etc) and
yet you want to split the diskimage into different files, you might define
your own FileList:

FLENTRY:  A filelist entry, very simple:

    FLENTRY name,offset,length

FLEND:    This terminates the FileList. No parameters needed.

Example for a FileList:

  FLENTRY FL_DISKNAME,0,$1600*37
  FLENTRY FL_HSNAME,$1600*37,$C8
  FLEND

FL_HSNAME:  ds.b  "HighScore",0


Pre-defined values to use with FLENTRY:

FL_DISKNAME:  Pointer to "Disk.#".

FL_DISKLENGTH:  The length of the diskimage.
```

```
DO NOT USE THE FILELIST IF THE DISK CONTAINS A DIRECTORY, RATHER USE SOME
CODE AT
                dsk_DiskCode
                 TO PARSE THE DIRECTORY!



###############
###     ###
### CRCList ###
###     ###
###############

This table is used for version checks. Every CRCList entry contains a
tracknumber and a CRC16 checksum for the track. When a CRCList is defined,
RawDIC will automatically read these tracks and calculate the checksums,
and if there is one checksum which is not the same as the CRCList entry
checksum the actual
                Disk-structure
                 will be discarded
and the dsk_AltDisk Disk-structure will be used.

To get hold of the CRC16 values RawDIC calculates on the tracks of a disk
you may set SFLG_DEBUG in the slave structure. The debug output will
carry all CRC16 checksums.

CRCENTRY: A CRCList entry.

    CRCENTRY track,checksum

CRCEND:   This terminates the CRCList. No parameters needed.

Example for a CRCList:

  CRCENTRY 19,$B25A
  CRCENTRY 20,$757E
  CRCEND
```

1.9 Flags

```
                Global flags for an ISlave (slv_Flags):

SFLG_DEBUG: Enables debug output.



Global flags for a
                Disk-structure
                 (dsk_Flags):

DFLG_SINGLESIDE: Only one diskside contains data.
    Attention! When setting this flag the tracknumber
    is equal to the cylindernumber.
```

```
DFLG_SWAPSIDES: The disksides will be swapped.

DFLG_ERRORSWAP: (Only for single sided disks!) On error, use other side for
    retry. Use this when both sides of the disk contain the
    same data so when the first side is bugged, the second
    most likely is still ok.
    This flag will only have affect when DFLG_SINGLESIDE is set.

DFLG_ERRORS:  Function calls will return on errors.

DFLG_RAWREADONLY: RawDIC will not use CMD_READ for standard amiga tracks, but
    TD_RAWREAD and it's own custom decoder.
    Senseful for disks with a standard Amiga track format but
    changed block headers so Amiga DOS will not be able to
    read these blocks, but RawDIC's custom Amiga format decoder
    might be (i.e. Simulcra, Outrun).
    When not set, CMD_READ will only be used on the first
    try, then RawDIC will use it's own routines anyway. This
    flag only will speed up this process.

DFLG_NORESTRICTIONS: RawDIC will not check the TrackList to follow
    the TrackList restrictions. You are now allowed to let
    lower tracks follow higher tracks, the disk image may
    contain a track more than once, even negative track
    increments are allowed (first track > last track).

DFLG_DOUBLEINC: RawDIC will use a track increment of 2 instead of 1.
    This has been introduced for doublesided disks with
    NO interleave of the disksides. Normally after a
    track on side 0 follows a track on side 1 but I
    encountered a format where the first 80 tracks are
    on side 0 and the next 80 tracks are on side 1.
    For an example look up OutRun.islave.asm.
    DO NOT USE FOR SINGLE SIDED DISKS!
    (That's what DFLG_SINGLESIDE is for...)
```

1.10 Functions

```
To allow more flexible ISlaves, you have various possibilities to add own
code to your ISlave. Since there are some functions which are quite often
needed when creating own ISlaves, they have been added to the RawDIC
function library.

When ISlave code is called, A5 will always carry the RawDIC library base.

All register except for registers which contain return values remain
unchanged.

rawdic_ReadTrack:

  Reads a track into the trackbuffer and automatically calls the
  decoder routine which is defined for it in the TrackList.

  D0.w=track
```

```
  => D0.l=errorcode
  => A1=trackbuffer

rawdic_NextSync:

  Moves the bitoffset in the MFM data to the end of the next sync.
  This function will first search a syncword and then skip all
  words which are equal to the syncword, so the pointer to the
  MFM data will be positioned at the first MFM word different to
  the sync.
  The 4 words BEFORE the MFM buffer will contain 4 syncwords, you
  may use these (i.e. for checksum calculation).

  => D0.l=errorcode
  => A0=MFM data buffer

rawdic_NextMFMword:

  Same as rawdic_NextSync, but with a syncword in D0 and not the sync
  defined in the TrackList.

  D0.w=bitpattern
  => D0.l=errorcode
  => A0=MFM data buffer

rawdic_SaveFile:

  Stores a memory block as file, an existing file will be overwritten.

  A0=filename
  A1=memory adress
  D0.l=length
  => D0.l=errorcode

rawdic_SaveDiskFile:

  Stores a part of the diskimage as file, an existing file will be
  overwritten.

  A0=filename
  D0.l=offset in diskimage
  D1.l=length
  => D0.l=errorcode

rawdic_AppendFile:

  Appends a memory block to an existing file, if the file does not
  exist, it will be created.

  A0=filename
  A1=memory adress
  D0.l=length
  => D0.l=errorcode

rawdic_DMFM_STANDARD:

  Standard Amiga track decoder.
```

```
  D0.b=sectors per track (normally 11)
  => D0.l=errorcode
```

1.11 ISlave code

There are various possibilities to add own code to an ISlave.

The most important one is the TrackList decoder. Here you must put some code which is able to convert the MFMBuffer into raw data and put it into the TrackBuffer.

When a TrackList decoder is called, some registers already contain values and pointers needed for track decoding:

```
  D0.w=Tracknumber
  A0=MFMBuffer
  A1=TrackBuffer (empty, filled with zeros)
  A5=RawDIC function library base

  => D0.l=errorcode
```

D0 must contain an errorcode when leaving the decoder again. IERR_OK when no error occured, IERR_CHECKSUM when a checksum check failed.

The next both possibilities are dsk_InitCode and dsk_DiskCode, both are called only one time per disk, but dsk_InitCode is called BEFORE a diskimage is created and dsk_DiskCode is called AFTER a diskimage is created.

Both have the same register configuration when called and on exit:

```
  D0.w=Disknumber
  A0=Pointer to the current Disk-structure
  A5=RawDIC function library base

  => D0.l=errorcode
```

Using RawDIC's library functions (example):

```
  moveq #0,d0     ; track 0
  jsr rawdic_ReadTrack(a5)  ; read & decode into trackbuffer
  bne.b .error      ; on error, exit

  ...

  moveq #IERR_OK,d0
.error
  rts
```

The "bne.b .error" is only needed when DFLG_ERRORS is set, otherwise all library functions will not return on error. The only exception is

```
rawdic_DMFM_STANDARD which is just a normal track decoder.

Note: You do not need to save any registers. The only register you need
      to take care of when returning to RawDIC is D0, it must contain
      an errorcode.
```

1.12 RawDIC's behaviour on Errors

```
Since not everything can always be ok, RawDIC has a built-in error
handling. The only thing you (as the ISlave coder) have to do is to
always return an errorcode so RawDIC has a chance to react.

At first I will describe the errors which you may indicate via
errorcodes in your own ISlaves.

The ok-returncode is 0, all other values are ~= 0. The zero flag of
the SR is set according to this when returning from RawDIC's library
functions.

IERR_OK   The name says it, everything went ok and RawDIC will
    continue with it's current task.

IERR_CHECKSUM Checksum error. Use this if a checksum test failed.

IERR_NOSYNC You don't need to care of this, as long es you use
    either a TrackList sync, rawdic_NextSync or
    rawdic_NextMFMword. Normally this will be handled
    automatically.

IERR_NOSECTOR For track formats which have multiple sectors on
    one track. It may happen that a sector is missing.
    In such a case return this error code.

IERR_OUTOFMEM Memory could not be allocated.


Now I will describe some errorcodes which you should NOT use as
errorcodes, but they are used as returncodes to some RawDIC library
functions, so you might be interested in them.
Please note: The functions will only return on error when DFLG_ERRORS
       is set, otherwise RawDIC will handle errors automatically.

IERR_NOWFILE  A file could not be written. (Reason is not described)

IERR_DISKRANGE  The DiskFile exceeds the diskimage (rawdic_SaveDiskFile).

IERR_NOTRACK  Not existant track (rawdic_ReadTrack)
    (a track MUST be defined in the TrackList)

IERR_NOFUNCTION A function may not be called at this point of the ISlave.
    Example: never call rawdic_ReadTrack in a track decoder.

All other defined errorcodes are for internal use in RawDIC only.
```

Behaviour on errors:

Most errors will simply terminate the image creation and an error
message will be displayed.

Due to the fact that read errors in many cases can be corrected by
simply re-reading a track, RawDIC will handle errors indicated by the
TrackList decoders different. This means RawDIC will try to repeat
reading a track and decoding it with the decoder until the decoder
replies IERR_OK. If this is not successful for a number of times,
an error requester will be opened and the user will be asked if
he wants RawDIC to continue to retry or to stop the whole action.

1.13 Glossary

ISlave: The slave which has to be created to use RawDIC as a
diskimager for a specific game.

Header: Every ISlave needs a header (SLAVE_HEADER).

Header-struct: The header follows this structure.

Disk-structure: Every disk of a game must have an own Disk-structure in
order to define RawDIC's behaviour.

TrackList: A table where the disk layout is defined.

FileList: A table where files which RawDIC automatically shall store
to HD are defined.

CRCList: A table which is needed for version checks.

Decoder: The routine which converts the MFMBuffer into raw data and
stores it into the TrackBuffer.

MFMBuffer: Every track will be read into the MFMBuffer.
The MFMBuffer has a size of $7c00 MFM words.

TrackBuffer: The TrackList decoder must convert the MFMBuffer into
the TrackBuffer.
The size of a TrackBuffer has the size defined in the
TrackList.

1.14 History

1.7 (Dark Angel) Using rawdic_ReadTrack in the InitCode caused
strange error messages after the first disk, this is fixed.
(Bored Seal) Calling rawdic_ReadTrack with a tracknumber
resulted in a freezed RawDIC. Now an error message is
displayed.

```
1.6 Released on 8.5.1999. (Together with WHDLoad 10.0)
  RawDIC behaved like DFLG_DOUBLEINC was set when
  DFLG_NORESTRICTIONS was set. This is fixed.
  The TrackList will be checked a second time after InitCode
  so you may use InitCode to change the TrackList.
  Pressing STOP and START again no longer causes a Software
  Failure.
  The library function rawdic_ReadTrack had major bugs and is
  fixed now.

1.5 Released on 16.3.1999. (Together with WHDLoad 9.2)
  SFLG_DEBUG now also disables the error requester.

1.4 Released on 10.3.1999. (A fast update to fix some nasties)
  Before reading a disk RawDIC will seek track 0 to avoid problems
  with the head positioning.
  Replaced all RectFill() calls. This graphics.library function
  doesn't work properbly on Picasso IV graphic cards and made
  the system crash when drawing the RawDIC progress bars.

1.3 Released on 8.3.1999.
  DFLG_NORESTRICTIONS introduced to allow more flexible TrackLists.
  Another new flag is DFLG_DOUBLEINC to use a trackincrement of 2
  instead of 1.
  In previous versions rawdic_DMFM_STANDARD cancelled sector
  decoding when a checksum error appeared. This caused "missing"
  sectors. Now all sectors are decoded and if one of them is
  bugged a checksum error will be returned.

1.2 Minor changes only: (This version wasn't released to the public)
  RawDIC will now exit automatically when "Cancel" is pressed
  in an error requester.
  Some fonts caused graphical bugs. Fixed.
  Improved ToolType parser.
  New ToolType: IGNOREERRORS

1.1 Released on 3.3.1999.
  Added detection for standard Amiga tracks in the tracklist.
  This detection can be disabled with DFLG_RAWREADONLY.

1.0 First version of RawDIC. Never released to the public.
```