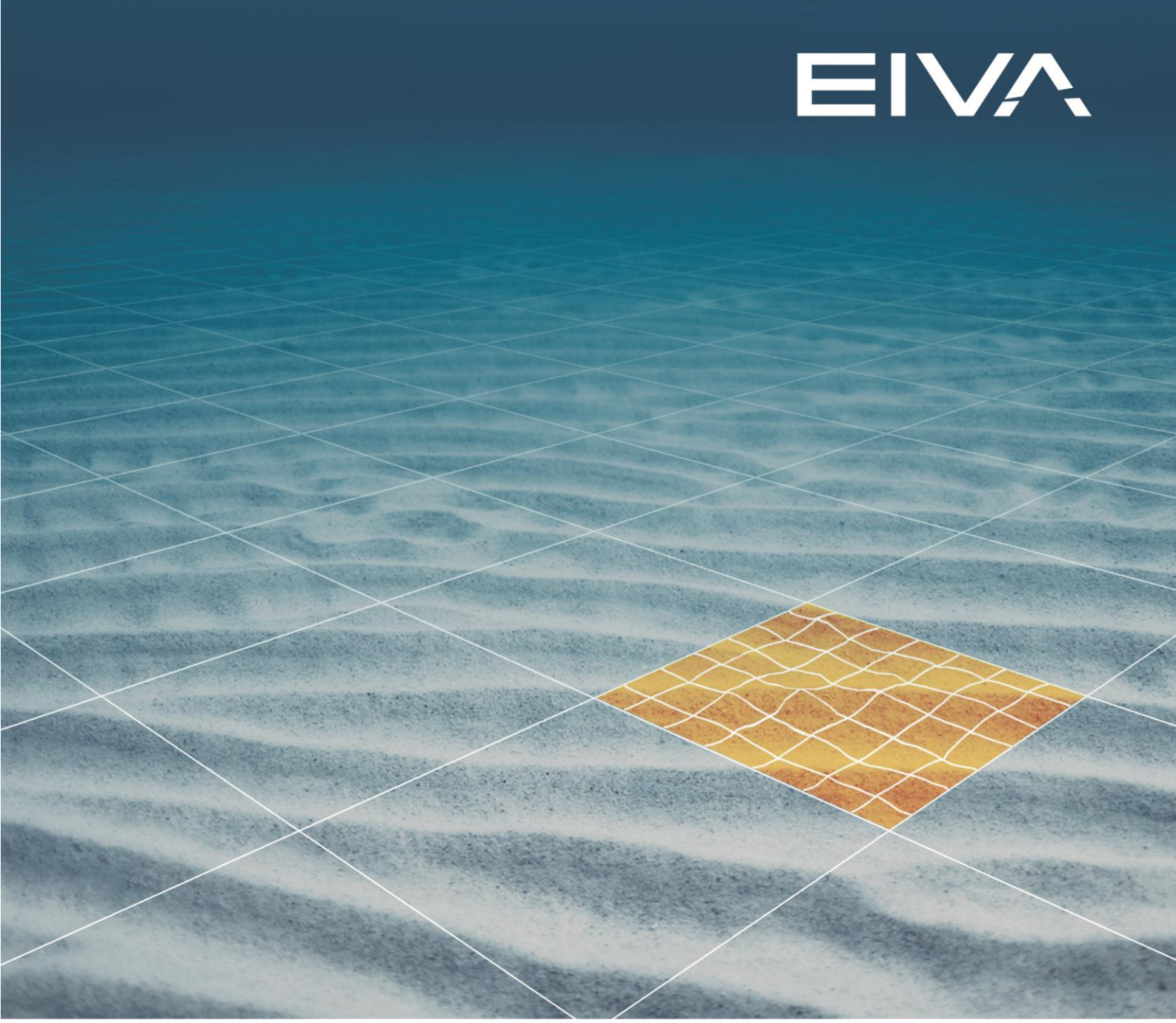

NAVIPAC

RAW RECORDING AND REPLAY

Last update: 02/04/2020
Version: 4.2

Contents

1 General overview

The NaviPac raw recording and replay function is a built-in part of NaviPac and coexists with the ordinary data logging.

Recording

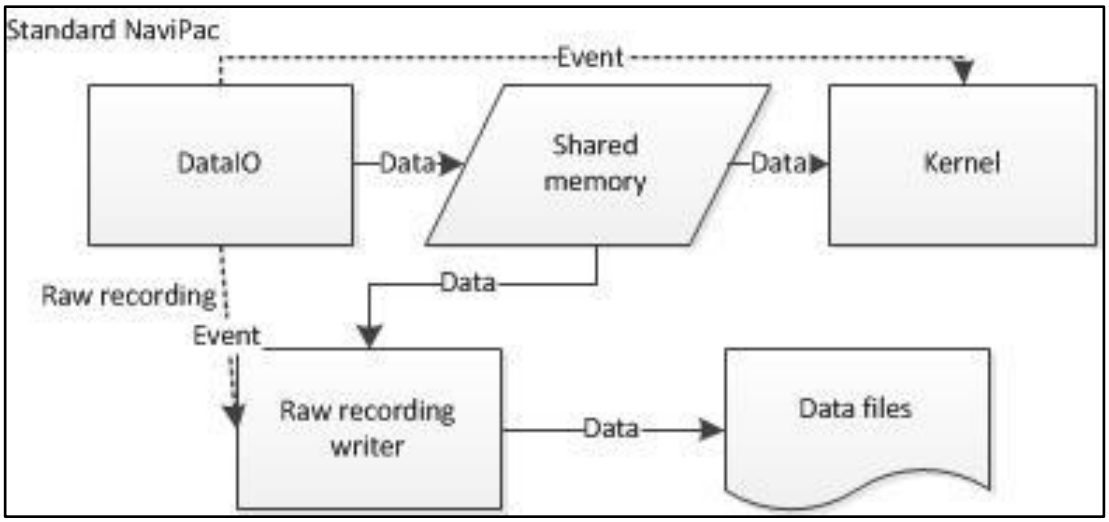

Figure 1 Recording data flow

Replay

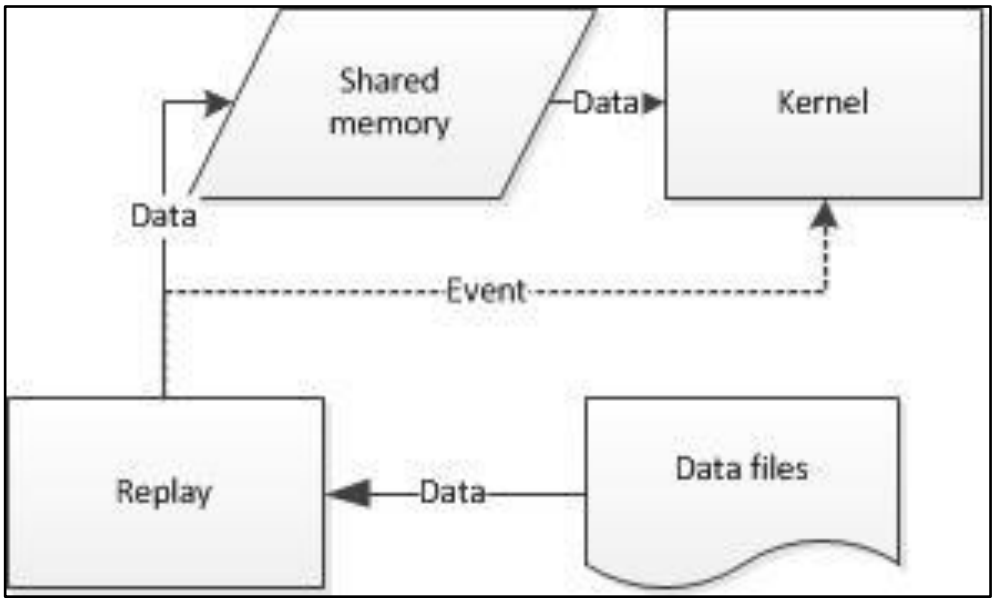

Figure 2 Replay data flow

As illustrated above, the remaining parts of NaviPac will not see the presence of recording and replay – to them, it just looks like an ordinary operation.

The raw recording is a power full tool for trouble shooting, so it could a case where the EIVA help desk ask you to perform a recording

Please note that the raw recording to generate rather huge data files – so consider carefully when using

2 Raw recording

2.1 Setting up raw recording

The use of NaviPac raw recording must be defined as a part of the general parameter configuration in **NaviPac Project Settings...**. Enable this function by checking the checkbox under **Project Settings > Advanced > Special raw logging**.

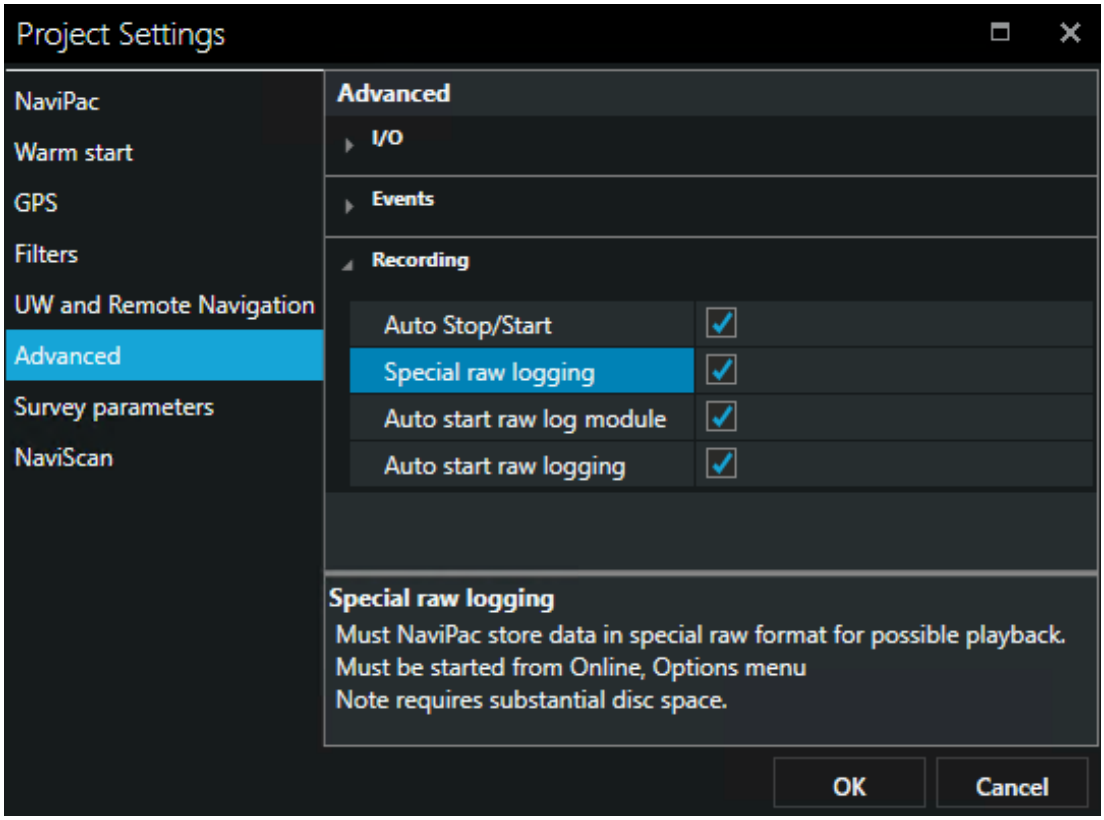

Figure 3 Enabling and setting up raw recording

- **Auto Stop/Start**
  Switches automatically to a new logfile if changes have been made in online setup (shift priority, weighting, C-O etc.) Recommended: Enabled.
- **Special raw logging**
  Enable the special raw recording function in NaviPac and store data in a special raw format for possible playback. The function must be started from **Online** > **Options > NaviPac Raw log** menu. Note that the logging requires substantial disc space.
- **Auto start raw log module**
  Include the raw recording module in the **Warm start** list of which modules to open automatically upon navigation start, e.g. C:\EIVA\NaviPac\bin\NPRawLog.exe.
  If not enabled here, you may start the module manually from **Online > Options > NaviPac Raw Log..**.
- **Auto start raw logging**
  Start raw recording (as a kind of black box recording) just after start of navigation.

2.2 Recording raw data

Raw recording in NaviPac is performed by the NPRawLog.exe module, which is located in **\eiva\navipac\bin** folder. The module can be started from the **Online > Options > NaviPac Raw Log** menu item or as a warm start function as described above.

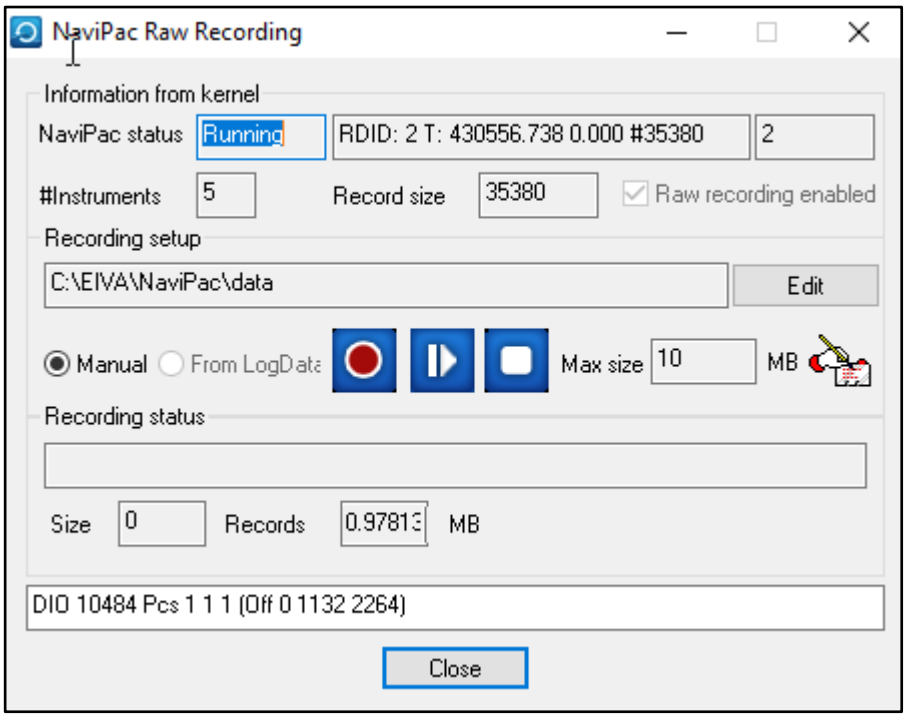

Figure 4 NaviPac Raw Recording module

The module contains all required information and parameters:

- **Information from kern**el:
  - **NaviPac status**: Read-only. Displays whether the NaviPac online navigation is running or not.
  - Packet information: RDID buffer id, NaviPac timer and packet size
  - Number of records (message counts).
  - **#Instruments**: Number of instruments included in the NaviPac setup.
  - **Record Size**: Read-only. Number of bytes recorded per cycle.
  - **Raw recording enabled**: Read-only. Displays that the raw recording function has been enabled.
- **Recording setu**p:
  - Path field: Read-only. The location where the data are to be recorded – change by pressing the **Edit** button to the right of this field**.**
  - **Manual** or **From LogData** control: If you select **From LogData**, recording starts/stops automatically upon signal from the **Log Data** module. If you select **Manual**, recording will be controlled by the **SOL**, **EOL** and **Pause** buttons in this module. The buttons are available/unavailable corresponding to actual recording status.
  - **Max size**: Read-only. Maximum file size in megabytes. If the value displayed is '0', then no maximum file size has been specified – for any other value, NaviPac changes the file automatically when it has reached the defined limit. Note that the total size of the recorded data file might be a little bigger than the defined size. To change the maximum size, please press the editing button to the right of the **Max size** field and enter the desired value in the pop-up box that opens.
- **Recording status:**
  - Current file name field: If LogData logging has been selected, it uses same name as LogData (extension *.NPR). If manual logging has been selected, it is named YYYYMMDDHHMMSS.NPR.
  - **Size**: Number of data records in the file – corresponds to the number of NaviPac cycles performed and size of the file in megabytes.
- Header size
  - Internal NaviPac information (just informative for the developer team)

Each raw data recording file is totally self-containing, as it besides the data also includes the internal configuration files required for a replay:

- GENSETUP.DB
- ONLSETUP.DB
- NAVIPAC.INI
- OBJECTS.TXT

For further details on raw data recording file format, please refer to Section 4 **Raw data** format.

3 Raw replay

Raw replay is performed via the special module **NPRawPlayBack.exe**, which is located in **\eiva\navipac\bin**. The module must be started manually via the file browser.

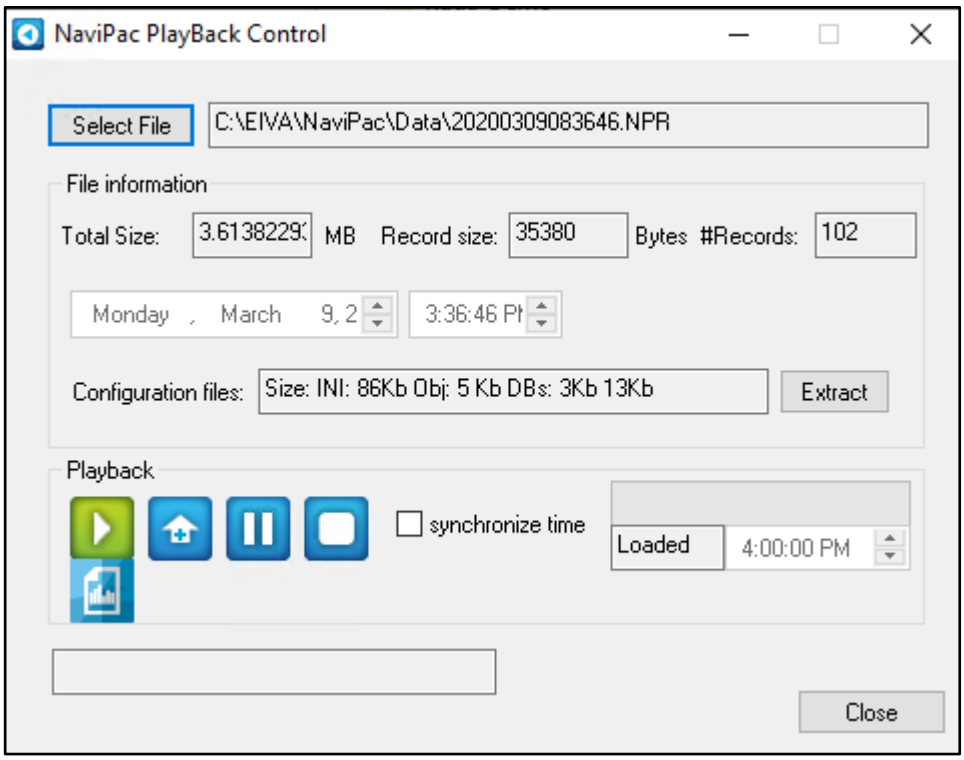

Figure 5 NaviPac PlayBack Control module

The first step in the use of the raw replay is to select a file via the **Select File** button. Selecting an acceptable file enables display of file information.

File information:

- **Total Size**: Read-only. Total size of the file in megabytes.
- **Record size**: Read-only. Size of one data cycle in bytes.
- **#Records**: Read-only. Number of records in the file – corresponds to the number of NaviPac cycles performed.
- Date field: Displays start date of the file.
- Time field: Displays start time of the file.
- **Configuration files**: Size of the four configuration files.
- **Extract**: Opens special dialogue box to extract the configuration files from the recorded file.

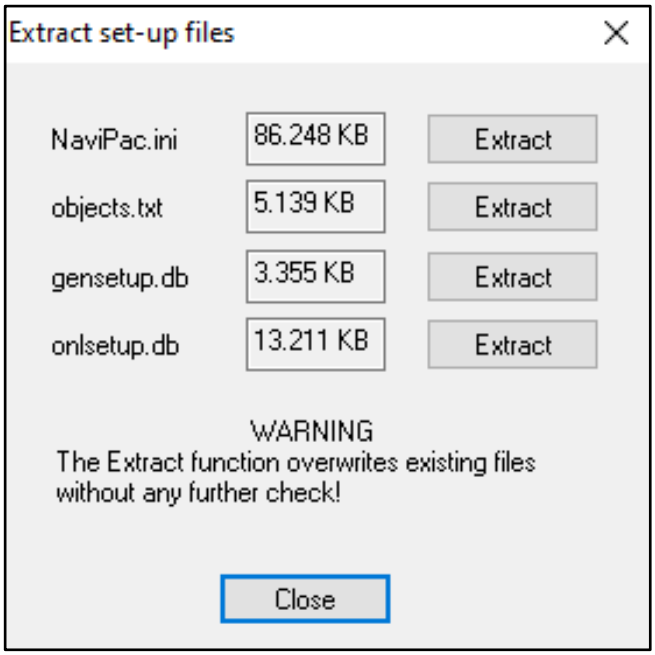

Figure 6 Extracting NaviPac configuration files

In this dialogue box, press **Extract** for each file you want to extract. Please note – if you are unsure of whether a file is already extracted or not – extract it. It is the user's responsibility to have compatible setup files prior to the playback.

Once the setup files are successfully extracted (or are already compatible), then you may access the replay functions.

Playback:

- **Synchronise time**: Select whether NaviPac should set the PC clock to match the time that the data was recorded. This must NOT be combined with any ordinary automatic time synchronisation functions. After replay, the PC clock is set back to the original time.

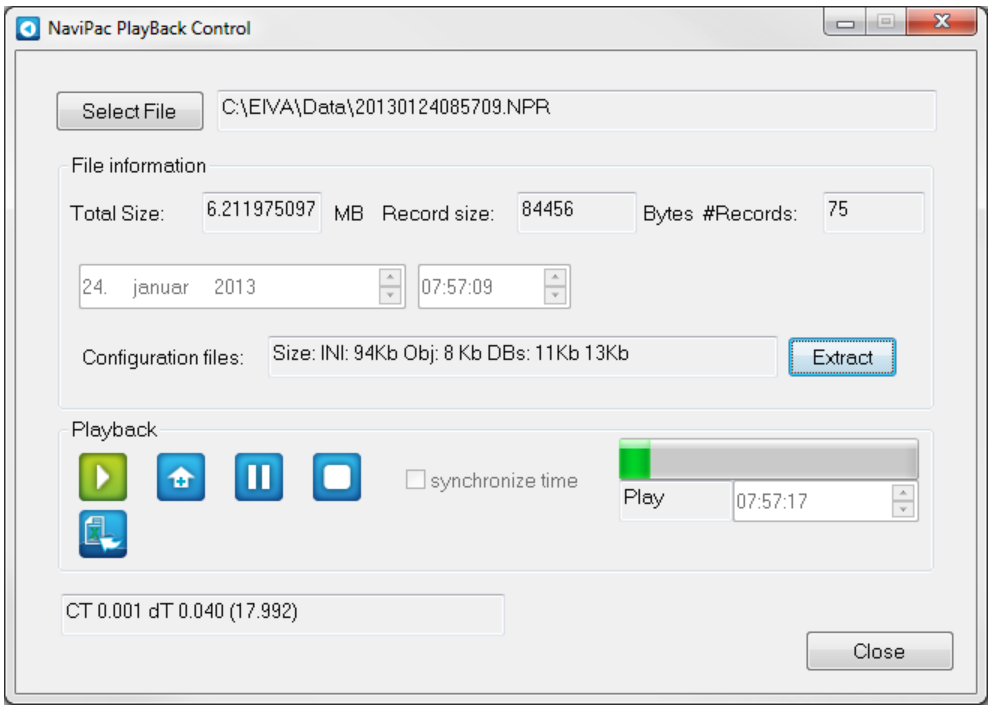

Figure 7 NaviPac PlayBack Control replay control buttons

- **Start** button: Start replay. Opens NaviPac online navigation and starts navigating using default settings (from restored DB and INI files). Opens output ports, enables recording, etc as if it was an ordinary online setup. If synchronisation is on, data will be timestamped as when collected.
- **Fast forward** button: Replay with four times the original speed. Data recording is less accurate – but computations are valid.
- **Pause** button: Pause the replay – keep the NaviPac windows open.
- **Stop** button: Stop replay and close all related NaviPac windows.
- **Extract raw data** button: Using this tool, you can extract the recorded raw data into a listing file including header and timestamped data – very similar to the raw data part of NaviPac LogData G format.
- **Play** field: Displays the current replay time position.

4 Raw data format

The recorded raw data file format is named NPR (NaviPac raw data) and consists of a file header and a series of raw data records.

The file header is defined as the following, using C language syntax:

```
class RawLogDataHeader
{
public:
       void       readHeader(void);
       char       title[255];            // File name
       double     timeGetTimeVal;        // Time at SOL  timeGetTime/100.0
       SYSTEMTIME GetSystemTimeVal;      // Time at SOL  GetSystemTime
       int        recordSize;            // Size of each logged entry
       RawLogDataHeader ()
             {timeGetTimeVal = 0;title[0] = 0;};
protected:
       char mess[100];                   // Small note - test only
       char objectText[12*1024];         // Copy of objects.txt
       char INIFileText[1024*100];       // Copy of NAVIPAC.INI
       char gensetupDB[40*1024];         // Copy of gensetup.DB
       char onlsetupDB[20*1024];         // Copy of onlsetup.DB
       int  objTextLen, INILen, onlLen, GenLen;
                                         // Actual size of supporting files
};
```

This information is always stored as the first record in the file – it may be read using standard file operations and the size of method (e.g. Read(lh,sizeof(RawLogDataHeader))).

Each time NaviPac performs a cycle, it generates a new record in the file. A record consists of two parts: a header and the data. The record header is defined as:

```
typedef struct _myHeader
{
       double        start_of_cycle,    // Local timestamp for start of cycle
                     cycle_time,        // How long time the processing took
                     nowT;              // Time of logging
       SYSTEMTIME    nowST;             // Start of cycle - via GetSystemTime
} MyHeader;
```

The record header is needed in order to establish the relationship between kernel time (starting computation) and data time (sensor timestamps) in both PC time (timeGetTime) and absolute time

The data itself follows the internal structure between the NaviPac dataIO and kernel modules (copy of shared memory block).

The data layout is described in dataio-kernel.h, and is in short terms:

```
/*=====================================================================
       Data from Data I/O to Kernel through shared memory
 =====================================================================*/

static char RawDataDBKeyPath[3][20] = {"RawDataDB1","RawDataDB2","RawDataDB3"};
#define MAX_INSTANCES      3        // 1 for reading, 1 for writing and one free
#define MAX_PORTS        128        // Maximum ports included
#define MAX_CYCLE_TIME     5        // RawDataDB contains data for maximum 5 seconds

typedef struct                      // Header information
 {
       short  Port;                 // Number of port or simulate id
       short  PacketLength;         // Length of current data type
       int    StartOffset;          // Reference to RawDataDB.Packet
       short  NoPackets;            // Number of packets in current cycle
 } RawDataDBHead;

typedef struct                      // One Packet in RawDataDB
 {
       double TStamp;               // Time stamp giving time a la time_t
       TIMEVAL TimeAbs;             // Time stamp in absolute time
       char   Data[1];              // RawDataDBHead.PacketLength bytes
 } SensorPacket;

 typedef struct                     // One RawDataDB instance
 {
       int    NoInstruments;        // Current number of instruments
       int    KernPID;              // Process ID
       int    DataioPID;
       RawDataDBHead Head[MAX_PORTS];     // Header for each port
       char   Data[1];              // All data for this cycle
 } RawDataDB;
```

Data from one cycle is kept in a **RawDataDB**, where Head defines the actual layout (RawDataDBHead defines port number, packet length and start offset) and Data includes all the sensor data records (SensorPacket gives timestamp and telegram).

The layout and size are fixed from cycle to cycle, so a lot of the space may be left empty.