Dokumentation Data-Logging-Server

Florian Pose, Ingenieurgemeinschaft IgH

30. Mai 2005

Zusammenfassung

Der „*DLS*“ ist ein Messdaten-Erfassungssystem, das in der Lage ist, hochfrequente Daten über lange Zeit zu sammeln und stark komprimiert abzulegen. Die Zielsetzung ist, dem Benutzer dabei jederzeit uneingeschränkten und performanten Zugriff auf die erfassten Daten zu gewähren: Sei es der gesamte Jahresüberblick oder eine winzige Schwankung im Bruchteil einer Sekunde.

Inhaltsverzeichnis

1 Allgemeines

1.1 Grundlagen der Datenerfassung

Der „Data-Logging-Server“ (im Folgenden „*DLS*“) ist ein Messdatenerfassungssystem, das in der Lage ist, beliebige Messdaten über lange Zeit zu erfassen, komprimieren, archivieren und bei Bedarf auch schnell wieder auszugeben.

Voraussetzung für eine Messdatenerfassung ist eine „*Datenquelle*“, die Messdaten liefert. In diesem Fall ist dies ein Server, der die Messdaten der von der *IgH* entwickelten *rt_lib* über das Netzwerk bereitstellt. Die Kommunikation mit der Datenquelle wird in Kapitel 2.2.3 beschrieben.

Alle zu liefernden Daten sind in „*Kanälen*“ organisiert. Ein Kanal ist die Abstraktion einer messbaren, physikalischen Größe, die von der Datenquelle angeboten wird. Eigenschaften eines Kanales sind die Einheit, die maximale Abtastfrequenz und der Datentyp.

Der *DLS* kann sich mit der Datenquelle verbinden und so Informationen über die angebotenen Kanäle abfragen. Auf dem selben Weg kann er dann auch die Messdaten zu bestimmten Kanälen anfordern und empfangen.

1.2 Messaufträge

Der *DLS* erfasst Daten über sog. „*Messaufträge*“. Diese beinhalten allgemeine Vorgaben zur Erfassung zusammen mit der Liste der zu erfassenden Kanäle und deren Vorgaben. Ein Messauftrag ist zwar immer an eine bestimmte Datenquelle gebunden. Es können beliebig viele Messaufträge gleichzeitig existieren.

Innerhalb eines Messauftrags können Daten von unterschiedlichen Kanälen gleichzeitig erfasst werden. Dazu muss für jeden, zu erfassenden Kanal eine sog. „*Kanalvorgabe*“ existieren. Diese fasst die Bedingungen zusammen, unter denen Daten von einem Kanal erfasst und gespeichert werden sollen. Dies sind die Abtastrate, die Blockgröße, die zu erfassenden Meta-Daten (siehe Kapitel 2.2.2), die Meta-Untersetzung und die Kopressionsmethode.

1.3 Datenablage

Erfasste Daten werden zusammen mit den Erfassungsvorgaben, Zeit- und Kanalinformationen nach Messaufträgen sortiert im „*DLS*-Datenverzeichnis“ gespeichert. Dieses ist standardmäßig `/vol/dls_data`. Wenn ein anderer Ablageort gewünscht ist, kann dies den Programmen des *DLS*-Packetes entweder über Kommandozeilenparameter (Option `-d`) mitgeteilt, oder in der Umgebungsvariable *$DLS_DIR* abgelegt werden. Die Suchreihenfolge ist immer: Parameter - Umgebungsvariable - Standardpfad.

Es können durchaus mehrere *DLS*-Datenverzeichnisse existieren, die aber von unterschiedlichen Instanzen des *DLS*-Daemons bedient werden müssen (siehe Kapitel 2.1).

Eine genaue Beschreibung der Struktur des *DLS*-Datenverzeichnisses und der darin enthaltene Daten befindet sich in Kapitel 3.

1.4 Werkzeuge

- `dls`
  Zur allgemeinen Kontrolle der Datenerfassung gibt es das Script `dls`. Es ist ein Kommandozeilentool, das die grundsätzliche Bereitschaft zur Messdatenerfassung für ein bestimmtes *DLS*-Datenverzeichnis steuert. Es bietet hierzu die Befehle `start`, `stop` und `restart`. Ausserdem kann mit diesem Script der Status der aktuellen Erfassungen textbasiert abgefragt werden.

- *DLS Manager*
  Messaufträge können über eine grafische Benutzeroberfläche, den *DLS Manager* (siehe Kapitel 4) editiert werden. Mit diesem ist der Benutzer in der Lage, neue Messaufträge anzulegen und bestehende anzupassen. Dies kann während einer Erfassung erfolgen. Es lässt sich auch einsehen, ob eine Erfassung gerade läuft.

- *DLS View*
  Zur Ansicht der Daten existiert ebenfalls ein grafisches Tool *DLS View* (siehe Kapitel 5). Der Benutzer kann damit beliebige, erfasste Kanäle eines Messauftrags über einer gemeinsamen Zeitskala anzeigen. Dabei ist auch die Navigation im Zeitfenster und das Bestimmen einzelner Datenwerte möglich.

2 Der *DLS*-Daemon (*dlsd*)

Für die gesamte Datenerfassung und -ablage ist der *DLS-Daemon* (kurz: *dlsd*) zuständig. Er ist ein Prozess, der im Hintergrund (ohne Verbindung zu einer Konsole) arbeitet und an ein bestimmtes *DLS*-Datenverzeichnis gebunden ist. Dieser muss immer laufen, wenn Daten erfasst werden sollen.

Eine Ansicht der Architektur des Gesamtsystems bietet Abbildung 1.

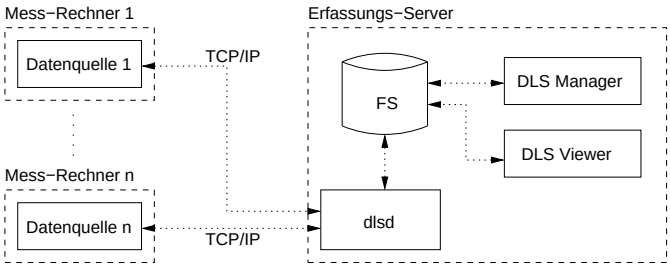

Abbildung 1: Architektur

2.1 Der *dlsd*-Mutterprozess

Ein mit dem Befehl `dls start` gestarteter *dlsd*-Prozess übernimmt die Überwachung der Messaufträge. Er heisst dann *dlsd*-Mutterprozess. Er hält im Speicher eine Liste von eigenen Kopien der im *DLS*-Datenverzeichnis vorhandenen Auftragsvorgaben vor, die zusätzlich Informationen über die dazugehörigen Erfassungsprozesse enthalten.

Jeder gestartete Mutterprozess muss in einem anderen *DLS*-Datenverzeichnis arbeiten. Es sind Schutzmechanismen implementiert, die dieses gewährleisten (*PID*-Dateien, siehe Anhang C).

2.1.1 Verhalten des Mutterprozesses

Anfangs liest der Mutterprozess alle Messauftrags-Vorgaben im *DLS*-Datenverzeichnis ein und spaltet für jeden aktiven Messauftrag eine Kopie seiner selbst (*dlsd*-Erfassungsprozess, siehe Kapitel 2.2) ab, die dann für die Erfassung des jeweiligen Messauftrages zuständig ist. Danach führt er in einem festgelegten Zeitintervall die folgenden Aufgaben aus:

- Prüfen, ob zwischenzeitlich Signale empfangen wurden. Das kann beispielsweise passieren, wenn der *dlsd* beendet werden soll, oder wenn einer der Erfassungsprozesse beendet wurde (siehe Kapitel 2.1.3).

- Prüfen des Spooling-Verzeichnisses auf neue Einträge. Wird im Spooling-Verzeichnis eine oder mehrere Dateien gefunden, werden diese als Spooling-Informationen bewertet und, wie im Kapitel 2.1.2 beschrieben, abgearbeitet.
- Prüfen der Erfassungsprozesse. Der Mutterprozess ist dafür verantwortlich, dass für jeden aktiven Messauftrag immer ein entsprechender Erfassungsprozess läuft.

2.1.2 Spooling

Um einen reibungsfreien Ablauf bei der Änderung von Messaufträgen während der Erfassung zu gewährleisten, wird ein Spooling-Verfahren angewendet. Da die Messaufträge in Dateien organisiert sind, wäre ohne ein derartiges Verfahren nicht sichergestellt, dass eine Vorgabendatei während des Schreibens durch den Benutzer nicht auch gleichzeitig vom *dlsd* gelesen würde. Dadurch könnten Fehler in der Verarbeitung entstehen.

Im *DLS*-Datenverzeichnis existiert daher ein Unterverzeichnis `spool`. Der *dlsd*-Mutterprozess leert dieses Verzeichnis beim Start und liest dann einmalig alle Auftragsvorgaben ein. Danach greift er nur noch lesend auf die Auftragsvorgaben zu, wenn er durch ein Spooling-Kommando explizit dazu aufgefordert wird.

Ein gültiges Spooling-Kommando ist für den *dlsd* eine Datei mit beliebigem Namen im Spooling-Verzeichnis, die lediglich eine ASCII-codierte Auftrags-ID (positive Ganzzahl) enthält. Anhand dieser ID kann er entscheiden, was zu tun ist:

- Kennt er noch keinen Auftrag mit dieser ID, so geht er davon aus, dass dieser neu angelegt wurde. Er importiert diesen und startet bei Bedarf den entsprechenden Erfassungsprozess.
- Kennt er hingegen einen Auftrag mit dieser ID **und** existiert die Datei mit den Auftragsvorgaben (`job.xml`, siehe Kapitel 3.2), werden die Vorgaben neu eingelesen und der Erfassungsprozess je nach Bedarf gestartet, gestoppt oder benachrichtigt.
- Kennt er einen Auftrag mit dieser ID, aber die Vorgabendatei existiert **nicht**, geht der *dlsd* davon aus, dass der Auftrag gelöscht wurde. Er beendet einen eventuell laufenden Erfassungsprozess und entfernt die Vorgabe aus seiner Liste.

Allgemein gilt, dass die Spooling-Datei zur Bestätigung der Übernahme der neuen Informationen gelöscht wird. Tritt während der Verarbeitung ein Fehler auf, wird die Datei im Spooling-Verzeichnis belassen.

2.1.3 Signalbehandlung

Folgende Signale werden vom *dlsd*-Mutterprozess verarbeitet:

- *SIGCHLD*
  Ein Kindprozess wurde beendet. Dies kann unterschiedliche Gründe haben:
  - Der Prozess wurde explizit beendet und hat den Rückgabewert 0 (kein Fehler). Dann wird er bei der nächsten Überprüfung neu gestartet, da die Beendigung nicht über die Vorgaben erfolgt ist und die Erfassung so schnell wie möglich fortgesetzt werden muss.
  - Der Prozess hat einen internen Fehler festgestellt und sich selbst mit dem Rückgabewert -1 beendet. Dies wird vom Mutterprozess so vermerkt, woraufhin der Prozess auch nicht neu gestartet wird. Allerdings erhält der Benutzer keine explizite Warnung, dass die Erfassung nicht mehr läuft. Es wird lediglich eine Nachricht an den *syslogd* verschickt.
  - Der Prozess hatte einen Timing-Fehler und hat sich selbst mit dem Rückgabewert -2 beendet. Der Prozess wird nach Ablauf einer festgelegten Zeitspanne vom Mutterprozess neu gestartet.

- *SIGINT*, *SIGTERM*
  Der *dlsd* soll beendet werden. Der Mutterprozess wird das Signal daraufhin an alle seine Kindprozesse weiterleiten, warten bis diese ihre Daten gespeichert haben und sich daraufhin selbst mit dem Rückgabewert 0 (kein Fehler) beenden.
- *SIGSEGV* (und weitere, kritische Signale)
  Diese Signale werden zur Sicherheit überwacht und vom *dlsd*-Mutterprozess und -Erfassungsprozess gleichartig behandelt. Der Prozess wird beim Auftreten eines solchen Zustandes eine Datei mit Namen `error_<PID>` im *DLS*-Datenverzeichnis hinterlassen, die Informationen über das empfangene Signal enthält, und sich kurz darauf mit dem Rückgabewert -3 beenden.

2.2 Der *dlsd*-Erfassungsprozess

Der vom *dlsd*-Mutterprozess abgespaltene Erfassungsprozess ist für die Kommunikation mit der Datenquelle, das Komprimieren der empfangenen Daten und das Speichern der komprimierten Daten auf der Festplatte zuständig. Er ist einem bestimmten Messauftrag zugeordnet, der ihm beim Start vom *dlsd*-Mutterprozess mitgegeben wurde. Dieser Messauftrag wird über das *DLS*-Datenverzeichnis und die entsprechende Messauftrags-ID eindeutig identifiziert. Jegliche Daten zu diesem Auftrag werden dort im Unterverzeichnis `job<ID>` abgelegt (siehe kapitel 3).

2.2.1 Verhalten des Erfassungsprozesses

Der *dlsd*-Erfassungsprozess importiert zunächst die Vorgaben seines Messauftrags aus der zentralen Vorgabendatei `job<ID>/job.xml` und verbindet sich dann über TCP/IP mit der dort angegebenen Datenquelle (Beschreibung des Kommunikationsprotokolles, siehe Kapitel 2.2.3).

Die erfassten Messdaten laufen für jeden Kanal in einen sog. *Block-Buffer*. Ist dieser voll, werden die Daten blockweise komprimiert, gespeichert und und mit dem Zeitstempel des ersten Datenwertes im Block indiziert. Dies hat den Vorteil, dass zur Komprimierung kein Streaming-Verfahren nötig ist und die einzelnen Datenwerte später gezielt auffindbar sind. Die Größe dieses Datenblockes (und somit des *Block-Buffers* kann vom Benutzer in den Kanalvorgaben angegeben werden.

Eine zeitlich kontinuierliche Folge von Datenblöcken wird in diesem Zusammenhang als „*Chunk*“ bezeichnet. Dieser enthält die Messdaten eines einzelnen Kanales, die in einem kontinuierlichen, abgeschlossenen Zeitbereich erfasst wurden. Er vereint die zu Grunde liegende Kanalvorgabe mit den realen Eigenschaften des entsprechenden Kanals der Datenquelle und den letztendlich erfassten Daten (siehe Kapitel 3.4).

2.2.2 Erzeugung von Meta-Daten

Um auch auf großen Datenmengen einen schnellen Überblick bieten zu können, speichert der *DLS*-Erfassungsprozess nicht nur die von der Datenquelle empfangenen („generischen“) Datenwerte, sondern legt zusätzlich auch noch über bestimmte Zeitspannen zusammengefasste Daten, sog. „Meta-Daten“ ab. Diese existieren in unterschiedlichen Reduktionsstärken, den sog. „Meta-Ebenen“. Ein lesend zugreifender Prozess kann dann anhand der gewünschten Auflösung entscheiden, welche Meta-Ebene er verwenden möchte und die Daten dadurch sehr schnell laden.

Mathematisch bedeutet das, dass die Komplexität des Algorithmus zum Laden von n Datenwerten einer Zeitspanne Δt und einer Abtastfrequenz f, wobei gilt: $n = \Delta t \cdot f$ nicht mehr $O(n)$, also linear Abhängig von der Anzahl zu Grunde liegender Datenwerte in der Zeitspanne ist, sondern nur noch Abhängig von der Anzahl der in der aktuellen Auflösung gewünschten Stützstellen ist, die aber wiederum unabhängig von der Anzahl der zu Grunde liegenden Datenwerte ist. Der Algorithmus hat also in Hinblick auf Zeit- und Speicherbedarf die Ordnung $O(1)$.

Zur Erzeugung der (redundanten) Meta-Daten existiert im *dlsd*-Erfassungsprozess parallel zum *Block-Buffer* ein sog. *Meta-Buffer*. Dieser Speicher fasst u Datenwerte, wobei u die sog. Meta-Untersetzung ist, die der Benutzer in den Kanalvorgaben festlegen kann. Die Meta-Untersetzung gilt für alle Ebenen. Ist

der *Meta-Buffer* voll, so wird aus den u generischen Datenwerten ein „Meta-Datenwert“ der Meta-Ebene 1 generiert und dort wiederum in einem *Block*- und einem *Meta-Buffer* abgelegt. Für diese gelten wieder die selben Regeln wie für die Ebene der generischen Daten, d. h. die Meta-Ebenen werden „kaskadiert“ erzeugt. So wird auch erst Speicher für eine neue Ebene reserviert, wenn der erste Meta-Wert dieser Ebene anfällt.

Es gibt verschiedene Typen von Meta-Daten, die auch gleichzeitig erzeugt werden können. Momentan werden Folgende unterstützt:

- Mittelwert („mean“, Maskenbit 0)
  Ein Meta-Wert der Ebene n ist der arithmetische Mittelwert von u Werten der Ebene $n-1$.
- Minima („min“, Maskenbit 1)
  Ein Meta-Wert der Ebene n ist der Kleinste von u Werten aus Ebene $n-1$.
- Maxima („max“, Maskenbit 2)
  Ein Meta-Wert der Ebene n ist der Größte von u Werten aus Ebene $n-1$.

Welche Typen von Meta-Werten letztendlich während der Erfassung erzeugt werden sollen, kann der Benutzer in den Kanalvorgaben mit der sog. Meta-Maske bestimmen. Diese entsteht durch die bitweise *ODER*-Verknüpfung der angegebenen Masken-Bits. (Beispiele: „Mittelwerte, Minima und Maxima“ entspricht Meta-Maske 7, „nur Mittelwerte“ entspricht Meta-Maske 1, und „minima und Maxima“ entspricht Meta-Maske 6).

Es kommt fast immer vor, dass beim Abschließen eines Chunks in einer Ebene weniger als u Werte übrig bleiben. Aus diesen Werten wird **kein** weiterer Meta-Wert mehr erzeugt, da man davon ausgehen kann, dass dieser Datenwert in der Anzeige immer schmaler als ein Pixel sein würde. Die überschüssigen Werte in den *Meta-Buffern* werden also verworfen.

2.2.3 Kommunikation mit der Datenquelle

Die Kommunikation mit der Datenquelle erfolgt über das an XML angelehnte Protokoll der *rt_lib* (Version ≥ 2.7) der *IgH*. Die Verbindung zur Datenquelle erfolgt über TCP/IP (TCP-Port 2345).

Identifikation der Datenquelle Nach dem Verbinden wird zunächst ein `<connected>`-Tag erwartet, das als Attribut `name` den Wert „`MSR`“ („Messen - Steuern - Regeln“, Kennung der *rt_lib*) enthalten muss. Die kodierte Version der Software der Datenquelle im Attribut `version` wird auf Kompatiblität mit der aktuellen *dlsd*-Version geprüft. Zusätzlich kann das `<connected>`-Tag ein Attribut `arch` enthalten, das die Architektur („Endianess“) der Datenquelle und somit der gesendeten Binärdaten enthält. Mögliche Werte sind `big` (für „big-endian“) oder `little` (für „little-endian“). Existiert kein `arch`-Attribut, wird als Quellenarchitektur „little-endian“ angenommen und eine Warnung ausgegeben.

Bestimmung der maximalen Abtastfrequenz Nachdem das `<connected>`-Tag empfangen und überprüft wurde, fragt der Erfassungsprozess zunächst den MSR-Parameter `/Taskinfo/Abtastfrequenz` ab und wartet auf die Antwort. Dieser Wert (die maximale Abtastfrequenz der Datenquelle) wird später benötigt, um die Plausibilität von Kanalvorgaben zu prüfen und die Untersetzung der Abtastfrequenzen zu berechnen.

Auslesen aller Kanäle Danach wird mit einem `<rk>`-Befehl die vollständige Liste der von der Datenquelle angebotenen Kanäle angefordert. Die Antwort der Datenquelle hat mit einem einleitenden `<channels>`-Tag zu beginnen, auf das dann die einzelnen Kanäle folgen, die jeweils in einem `<channel>`-Tag beschrieben sind. Das Ende der Kanalliste markiert wiederum ein `</channels>`-Tag.

Beispiel einer Antwort der Datenquelle auf einen `<rk>`-Befehl:

```
<channels>
<channel name="/Time" unit="s" alias="" index="0" ↩
 typ="TDBL" bufsize="50000" HZ="10000" value="1112814601.3209"/>
<channel name="/Taskinfo/Controller_Execution_Time" unit="us" ↩
 alias="" index="6" typ="TUINT" bufsize="50000" HZ="10000" value="22"/>
<channel name="/Taskinfo/Controller_Call_Time" unit="us" alias="" ↩
 index="7" typ="TUINT" bufsize="50000" HZ="10000" value="99"/>
<channel name="/Istwert/Kraft" unit="N" alias="" index="9" ↩
 typ="TDBL" bufsize="50000" HZ="10000" value="-0.6745"/>
<channel name="/Istwert/Druck" unit="bar" alias="" index="12" ↩
 typ="TDBL" bufsize="50000" HZ="10000" value="0.1372"/>
</channels>
```

Von den Attributen im `<channel>`-Tag werden Folgende zur späteren Verwendung gespeichert:

- `name` - Kanalname (eindeutig)
- `unit` - Einheit des Kanals (wird als String gespeichert)
- `index` - Die Position des Kanals innerhalb der Liste. Diese wird später auch als Identifier für ein Kanalverzeichnis innerhalb des *DLS*-Datenverzeichnis verwendet (siehe Kapitel 3).
- `typ` - Datentyp. Dieser muss einer der bekannten Datentypen aus der Tabelle in Anhang B sein, damit die später empfangenen Daten verarbeitet werden können.
- `bufsize` - Größe des Ringspeichers innerhalb der Datenquelle. Diese wird später zur Prüfung der Plausibilität einer vorgegebenen Abtastrate herangezogen:

$$BlockSize * Reduction \overset{!}{\leq} \frac{BufferSize}{2} \tag{1}$$

- `HZ` - Kanalspezifische, maximale Abtastrate.

All diese Kanalinformationen werden in einer Liste im Speicher jedes Erfassungsprozesses abgelegt und beim Hinzufügen oder bei der Änderung einer Kanalvorgabe für die Plausibilitätsprüfungen herangezogen.

Start der Erfassung Kennt der Erfassungsprozess die Liste der Kanäle, so wird die Erfassung gestartet (wenn nicht vorher auf den Trigger-Parameter gewartet werden soll). Dies geschieht über den Befehl `<xsad>`, der für jeden zu erfassenden Kanal einmal gesendet wird. Attribute des Befehls sind:

- `channels` - Enthält den Index des angefragten Kanals in der Liste aller Kanäle,
- `reduction` - der (ganzzahlige) Untersetzungsfaktor von der maximalen Abtastfrequenz des Kanals, um die absolute Abtastfrequenz zu beschreiben,
- `blocksize` - die Anzahl der in einem Block zu sendenden Werte (dieser Wert ist vollkommen unabhängig von der Blockgröße in der Kanalvorgabe), und
- `coding` - die Kodierung der Daten, welche im Moment auf „`Base64`“ festgelegt ist.

Ein typisches Kommando zum Start der Erfassung von daten eines Kanals könnte also wie folgt aussehen:

```
<xsad channels="7" reduction="100" blocksize="1000" coding="Base64"/>
```

Empfang von Daten Der Erfassungsprozess wartet nun auf ein `<data>`-Tag, das den Beginn eines Blockes von Kanaldatentags bedeutet. Dieses Tag muss ein Attribut `time` enthalten, das dem Zeitstempel aller jeweils letzten Datenwerte in den folgenden Kanaldatentags entspricht. Diese werden als `<F>`-Tags erwartet, die jeweils die letzten Messdaten eines einzelnen Kanales enthalten und die Attribute `c` (Kanalindex) und `d` (kodierte Messdaten) besitzen. Nach dem letzten `<F>`-Tag hat ein `</data>`-Tag zu folgen, dass den Erfassungsprozess wieder in den Wartezustand versetzt.

Änderung von Kanalvorgaben Bei einer Änderung einer Kanalvorgabe während der Erfassung wird vom Erfassungsprozess ein erneutes `<xsad>`-Tag gesendet, dass neben den neuen Kanalvorgaben auch ein `id`-Attribut besitzt, welches eine (verbindungsweit) eindeutige Kommandokennung darstellt. Hat die Datenquelle die Kanalvorgaben übernommen und entsprechen die nächsten Daten des betroffenen Kanals definitiv den **neuen** Vorgaben, so wird von der Datenquelle vorher ein `<ack>`-Tag erwartet, das als Attribut die Kommando-ID des entsprechenden `<xsad>`-Tags besitzt. Dann wird auch der Erfassungsprozess endgültig auf die neuen Kanalvorgaben umgestellt.

2.2.4 Begrenzung des Datenvolumens (Quota)

Der *dlsd* kennt Mechanismen zum Begrenzen des erforderlichen Speicherplatzes für die erfassten Daten eines Messauftrages. Es werden verschiedene Kriterien für die Überschreitung dieser Grenzen unterstützt:

- Daten-Quota
  Die gesamte Größe des Job-Verzeichnisses im Dateisystem darf eine bestimmte Grenze nicht überschreiten.
- Zeit-Quota
  Die Zeitspanne der gesamten, erfassten Daten eines Messauftrages, soll eine bestimmte Breite nicht überschreiten.

Hat der Benutzer eine oder mehrere Quotas aktiviert und überschreitet die Gesamtheit der erfassten Daten eines oder mehrere der Kriterien, so wird jeweils der älteste Chunk entfernt, bis die Kriterien nicht mehr erfüllt werden. Der neueste Chunk jedes Kanales wird allerdings nie entfernt, da hier gerade eine Erfassung stattfinden könnte.

Die Aufgabe des Löschens übernimmt der *DLS*-Quota-Daemon. Dieser muss immer parallel zum *dlsd* laufen, sobald in mindestens einem Auftrag Quotas konfiguriert sind. Der Start erfolgt manuell mit `dls_quota`. Kommandozeilenparameter sind in Anhang D.5 einsehbar.

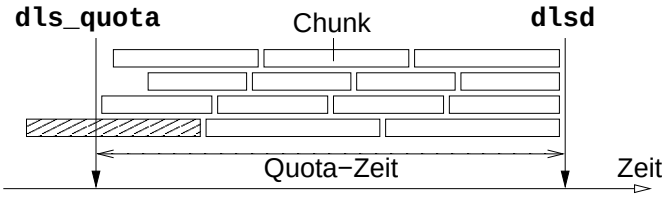

Abbildung 2: Einhaltung der Zeit-Quota

Da der *DLS*-Quota-Daemon immer nur komplette Chunks auf einmal entfernen kann, wäre ein sukzessives Löschen nicht möglich, wenn die erfassten Daten eines Kanales aus lediglich einem einzelnen Chunk bestehen würden. Es muss also sichergestellt sein, dass immer viele, kleine Chunks existieren, auch wenn der *dlsd*-Erfassungsprozess nicht unterbrochen wurde.

Daher überwacht der *dlsd*-Erfassungsprozess die Quota-Kriterien auch selber. Er sorgt bei aktivierter Quota dafür, dass innerhalb der kritischen Spanne immer genug einzelne Chunks entstehen (Abbildung 2). Dafür teilt er jedes gesetzte Quota-Kriterium in gleiche Anteile auf und beginnt bei Überschreitung eines dieser feineren Kriterien eigenmächtig einen neuen Chunk.

Da das Abschliessen eines Chunks sehr zeitaufwändig sein kann und dies nicht mit den Echtzeit-Anforderungen des *dlsd*-Erfassungsprozesses vereinbar ist, wird zum Speichern der restlichen, erfassten Daten ein eigener Prozess erzeugt. Dieser „Aufräumprozess“ speichert nun alle Daten des aktuellen Chunks ab, während der Erfassungsprozess diese einfach verwirft und sich der Erfassung der Daten des neu begonnenen Chunks widmet. Nach dem Abschliessen des „alten“ Chunks beendet sich der Aufräumprozess selbstständig.

2.2.5 Nachrichten der Datenquelle

Die Datenquelle kann -neben den Messdaten- auch jederzeit Nachrichten versenden. Diese Nachrichten beinhalten Notizen des Benutzers, die mit in den Datenstrom aufgenommen werden sollen, Warnungen oder Fehlerzustände. Insgesamt gibt es folgende Nachrichtstypen:

- `info` - Eine Information, die lediglich für den aktuellen Erfassungsprozess bestimmt ist.
- `warn` - Eine Warnung der Datenquelle.
- `error` - In der Datenquelle ist ein Fehler aufgetreten.
- `crit_error` - In der Datenquelle ist ein Fehler aufgetreten, der den weiteren Betrieb schwierig bis unmöglich macht.
- `broadcast` - Eine Nachricht für alle Prozesse, die gerade mit der Datenquelle verbunden sind.

Eine Nachricht wird von der Datenquelle immer als einzelnes XML-Tag versendet, welches als Titel den Typ der Nachricht enthält. Weiterhin sind ein Attribut `time` enthalten, dass den Zeitstempel der Nachricht in Sekunden enthält und -je nach Nachricht- ein Attribut `text`, das aber vom *dlsd* nicht weiter ausgewertet wird.

Ein Typisches Nachrichten-Tag sieht also so aus:

```
<broadcast time="1093072549.866241" text="Test-Nachricht"/>
```

Der *dlsd*-Erfassungsprozess speichert die Nachrichten im Unterverzeichnis `messages` des Auftragsverzeichnisses (siehe Kapitel 3.6). Nachrichten sind -wie Messdaten- in Chunks organisiert. Allerdings wurde dieses Konzept hier etwas abgewandelt: Nachrichten-Chunks sind nicht zeitlich kontinuierlich und wurden nur deshalb in Chunks angeordnet, um später bestimmte Zeitspannen mit Nachrichten einfacher löschen zu können.

2.2.6 Signalbehandlung

Folgende Signale werden vom *dlsd*-Erfassungsprozess verarbeitet:

- *SIGINT*, *SIGTERM*
  Der Erfassungsprozess soll beendet werden. Er wird daraufhin sofort die Verbindung zur Datenquelle schliessen und die noch im Speicher befindlichen Daten auf die Festplatte sichern. Dies kann einige Sekunden dauern, da u. U. viele Dateien geschrieben werden müssen. Passiert dabei kein Fehler, beendet sich der Prozess mit einem Rückgabewert von 0.
- *SIGHUP*
  Der Empfang dieses Signales bedeutet für den Erfassungsprozess, dass er seine Vorgabedaten neu einlesen muss. Nachdem er dies getan hat, prüft er sofort, ob er nach den neuen Vorgaben überhaupt noch erfassen muss. Wenn nicht, leitet er die Beendigung wie bei *SIGINT* oder *SIGTERM* ein. Ansonsten sendet er evtl. geänderte Vorgaben an die Datenquelle und erfasst nach Bestätigung (siehe Kapitel 2.2.3) unter den neuen Bedingungen weiter.

- *SIGCHLD*
  Ein „Aufräumprozess“ (siehe Kapitel 2.2.4) hat sich beendet. Dies wird lediglich über den *syslogd* vermerkt.
- *SIGSEGV* (und weitere, kritische Signale)
  Behandlung wie im Mutterprozess (siehe Kapitel 2.1.3).

3 Das *DLS*-Datenverzeichnis

Alle persistenten Daten des *DLS*-Systems sind in *DLS-Datenverzeichnissen* organisiert. Die grundsätzliche Struktur zeigt Abbildung 3.

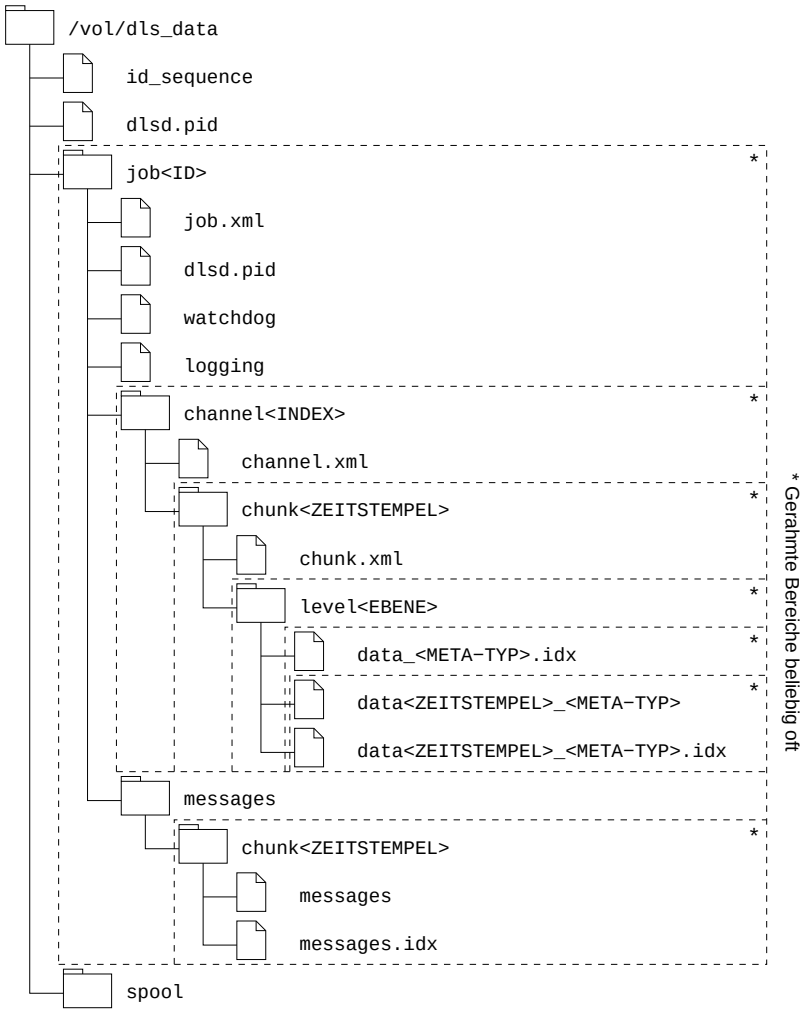

Abbildung 3: Struktur des *DLS*-Datenverzeichnisses

3.1 Wurzelverzeichnis

Das Wurzelverzeichnis ist die oberste Verzeichnisebene innerhalb des *DLS*-Datenverzeichnisses. Hier liegen hauptsächlich Dateien für den *dlsd*-Mutterprozess. Darüberhinaus liegen im Wurzelverzeichnis auch alle Auftragsverzeichnisse (siehe Kapitel 3.2).

Dateien und Unterverzeichnisse im Wurzelverzeichnis:

- `id_sequence`
  Diese Datei enthält die nächste, freie Auftrags-ID als *ASCII*-kodierte Ziffernfolge. Sie wird vom *DLS Manager* benötigt, wenn ein neuer Auftrag angelegt werden soll. Dieser liest die ID, verwendet diese für den neuen Auftrag, erhöht sie um 1 und schreibt die neue ID in die Datei zurück.

- `dlsd.pid`
  Dies ist die *PID*-Datei des *dlsd*-Mutterprozesses (siehe Anhang C). Sie wird zur Laufzeit automatisch angelegt und zeigt an, dass gerade ein *dlsd*-Mutterprozess läuft.

- `job<ID>`
  Jedes Auftragsverzeichnis liegt im *DLS*-Wurzelverzeichnis. Der Name ist immer `job`, gefolgt von der Auftrags-ID. Siehe Kapitel 3.2.

- `spool`
  Dies ist das Spooling-Verzeichnis des *dlsd*-Mutterprozesses. Eine Beschreibung steht in Kapitel 2.1.2.

3.2 Auftrags-Verzeichnisse

Jedes Auftragsverzeichnis („`job<ID>`“) wird im laufenden Betrieb von einem eigenen *dlsd*-Erfassungsprozess bearbeitet. Dieser liest von dort seine Vorgaben und schreibt auch die erfassten Daten dorthin.

Dateien und Unterverzeichnisse in einem Auftragsverzeichnis:

- `job.xml`
  Die zentrale Vorgabendatei für einen Auftrag. Sie enthält Auftrags- und Kanalvorgaben. Wird diese editiert, während der dazu gehörige Erfassungsprozess läuft, so muss ein Spooling-Kommando (siehe Kapitel 2.1.2) erzeugt werden, damit dieser die neuen Vorgaben übernimmt. Der *DLS Manager* macht dies automatisch.

  Die Vorgabendatei ist im XML-Format und enthält folgende Informationen (Reihenfolge obligatorisch!):

```
<dlsjob>
 <description text="Beschreibung"/>
 <state name="(running|paused)"/>
 <source address="IP-Adresse oder Hostname"/>
 <quota size="Daten-Quota" time="Zeit-Quota"/>
 <trigger parameter="Trigger-Parameter"/>

 <channels>
  <channel name="Kanalname" ↩
           frequency="Abtastrate" ↩
           block_size="Blockgröße" ↩
           meta_mask="Meta-Maske" ↩
           meta_reduction="Meta-Untersetzung" ↩
           format="Kompressionsformat" ↩
           mdct_block_size="MDCT-Blockgröße" ↩
           mdct_accuracy="MDCT-Genauigkeit" ↩
           type="Datentyp"/>
 </channels>
</dlsjob>
```

  Die Attribute `mdct_block_size` bzw. `mdct_accuracy` werden nur benötigt, wenn das Kompressionsformat auf der MDCT (siehe Kapitel 6.2) basiert.

  Die Vorgaben können mit dem *DLS Manager* editiert werden. Die Beschreibungen der einzelnen Parameter finden sich entsprechend in den Kapiteln 4.2 bzw. 4.5.

- `watchdog` und `logging`
  Dies sind zwei leere Dateien, die für die Überwachung der *dlsd*-Erfassungprozesse seitens des *DLS Managers* verwendet werden. Läuft ein Erfassungsprozess für das Auftragsverzeichnis, so ändert er jede Sekunde den Zeitstempel der Datei `watchdog`. Erfasst der Prozess gleichzeitig auch Daten, so verfährt er ebenso mit der Datei `logging`. Der *DLS Manager* prüft die Zeitstempel dieser Dateien regelmäßig, erhält so Aufschluß über den Zustand des Erfassungsprozesses und kann dies so dem Benutzer anzeigen.

- `dlsd.pid`
  Dies ist die *PID*-Datei des *dlsd*-Erfassungsprozesses (siehe Anhang C). Sie wird zur Laufzeit automatisch angelegt und zeigt an, dass der Erfassungsprozess gerade läuft.

- `channel<INDEX>`
  Die erfassten Daten sind weiterhin in Kanälen organisiert, die jeweils ihr eigenes Kanal-Verzeichnis besitzen (siehe Kapitel 3.3). Der Index im Namen des Kanalverzeichnisses entspricht dem Kanalindex, den der Befehl `<rk>` beim Auslesen aller Kanäle der Datenquelle während des Startvorganges des *dlsd*-Mutterprozesses zurückgeliefert hat (siehe Kapitel 2.2.3).

- `messages`
  Jeder Erfassungsprozess legt die Nachrichten, die er währen der Erfassung von der Datenquelle empfangen hat in diesem Verzeichnis ab. Wenn es noch nicht existiert, erstellt er es bei Bedarf. Nachrichten sind -wie Messdaten- in Chunks organisiert. Siehe Kapitel 3.6.

3.3 Kanalverzeichnisse

In den Kanal-Verzeichnissen („`channel<INDEX>`“) werden alle Daten abgelegt, die zu diesem Kanal erfasst wurden. Ein Kanalverzeichnis ist fest einem bestimmten Kanal der Datenquelle zugeordnet. Zur Beschreibung der Eigenschaften des Kanals existiert die Datei `channels.xml`, die folgenden Inhalt hat:

```
<dlschannel>
 <channel name="Kanalname" index="Index" unit="Einheit" type="Datentyp"/>
</dlschannel>
```

Diese Datei dient nicht nur zur Beschreibung der Daten in den Chunk-Verzeichnissen, sondern wird auch jedesmal vom *dlsd*-Erfassungsprozess überprüft, wenn eine neue Erfassung in ein Kanalverzeichnis erfolgen soll. Diese darf nämlich nur stattfinden, wenn die Kanaldaten (Name, Index, Einheit und Typ) sich nicht geändert haben.

3.4 Chunk-Verzeichnisse

Die erfassten Daten eines Kanals sind in „*Chunks*“ organisiert („`chunk<ZEITSTEMPEL>`“). Ein Chunk ist eine lückenlos erfasste Serie von Daten, die ab einem bestimmten Zeitpunkt mit der selben Kanalvorgabe erfasst wurden. Der Zeitstempel im Verzeichnisnamen ist der Zeitstempel des ersten Datenwertes im Chunk. Zur Beschreibung der Chunk-Eigenschaften existiert die Datei `chunk.xml` mit folgendem Inhalt:

```
<dlschunk>
 <chunk sample_frequency="Abtastrate" ↩
        block_size="Datenblockgröße" ↩
        meta_mask="Meta-Maske" ↩
        meta_reduction="Meta-Untersetzung" ↩
        format="Kompressionsformat" ↩
        mdct_block_size="MDCT-Blockgröße" ↩
        mdct_accuracy="MDCT-Genauigkeit" ↩
```

```
        architecture="Architektur (Endianess)"/>
</dlschunk>
```

Die `mdct_*`-Attribute existieren nur, wenn das Kompressionsformat auf der MDCT beruht (siehe Kapitel 6.2).

In jedem Chunk-Verzeichnis sind die Daten wiederum in Verzeichnissen untergebracht, die deren Meta-Ebene entsprechen (generische Daten im Verzeichnis `level0`, Daten der ersten Meta-Ebene in `level1` usw.).

3.5 Daten-Verzeichnisse

Die Datenverzeichnisse („`level<Meta-Ebene>`“), die gleichzeitig auch die Sortierung der Daten nach der jeweiligen Meta-Ebene darstellen, sind der unterste Teil der Verzeichnishierarchie. Hier befinden sich die Datendateien und die dazugehörigen Index-Dateien.

Datendateien Datendateien beinhalten die erfassten Messdaten. Diese werden für jeden Meta-Typ getrennt angelegt. Daher haben sie folgendes Namensschema:

`data<ZEITSTEMPEL>_<META-TYP>`

Der Zeitstempel im Dateinamen ist der Zeitstempel des ersten Datenwertes im ersten Block in dieser Datei.

Im Verzeichnis `level0` ist der Meta-Typ immer „`gen`“(„generic“).

Datendateien besitzen eine einfache XML-Struktur. Jeder Datenblock erscheint hier als `<b>`-Tag. Dieses erhält den Zeitstempel des ersten Wertes im Block als Attribut `t` („time“), die Anzahl der komprimierten Datenwerte als Attribut `s` („size“) und die kodierten Daten als Attribut `d` („data“).

Datendateien haben eine festgelegte Maximalgröße. Sobals der *dlsd*-Erfassungprozess diese mit dem Anfügen des nächsten Blockes überschreiten würde, legt er zuerst eine neue Datendatei an.

Mit welchen Parametern letztendlich die Erfassung stattgefunden hat und in welcher Weise komprimiert wurde, ist nur zusammen mit den höhergelegenen Beschreibungsdateien `chunk.xml` und `channel.xml` ersichtlich.

Index-Dateien Index-Dateien sind Binärdateien mit fester Eintragslänge, die immer einer Datendatei zugeordnet sind. Sie stellen sehr schnell auslesbare Informationen über die Datenblöcke in der entsprechenden Datendatei bereit. Das Namensschema ist das der Datendatei mit einer entsprechenden Erweiterung:

`data<ZEITSTEMPEL>_<META-TYP>.idx`

Die Einträge in der Index-Datei entsprechen immer einem Block in der Datendatei. Den Aufbau eines Eintrags zeigt Abbildung 4.

Abbildung 4: Eintrag einer Index-Datei

Jeder Eintrag ist 20 Bytes lang. Die ersten 8 Bytes entsprechen dem in einen `long long int` konvertierte Zeitstempel des ersten Datenwertes im entsprechenden Block in Mikrosekunden. Die nächsten 8 Bytes entsprechen in gleicher Weise dem Zeitstempel des letzten Datenwertes im Block. Die letzten 4 Byte sind die als `unsigned int` kodierte Offset-Adresse des Block-Tags in der Datendatei, d. h. die Position des einleitenden „`<`“-Zeichens.

Globale Index-Dateien „Globale“ Index-Dateien erleichtern das Ermitteln der Zeitspannen der Daten der einzelnen Datendateien eines bestimmten Meta-Typs. Ihr Namensschema lautet:

`data_<META-TYP>.idx`

Ein Eintrag in einer globalen Index-Datei entspricht immer einer Datendatei des selben Meta-Typs. Die Struktur eines Eintrags zeigt Abbildung 5.

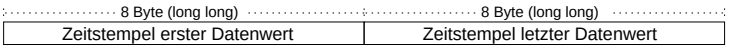

Abbildung 5: Eintrag einer globalen Index-Datei

Ein Eintrag in einer globalen Index-Datei ist immer 16 Bytes lang. Die ersten 8 Bytes entsprechen dem Zeitstempel des ersten Datenwertes in der Datendatei in Mikrosekunden als `long long int`, die letzten 8 Bytes dem des letzten Datenwertes.

Wird in eine Datendatei gerade erfasst, so ist der zweite Zeitstempel des entsprechenden (letzten) Eintrags in der globalen Index-Datei 0. In diesem Fall muss der gesuchte Zeitstempel durch ein Auslesen des letzten Eintrags in der entsprechenden Daten-Indexdatei ermittelt werden. Sobald die Erfassung in die Datendatei beendet wurde, wird der zweite Zeitstempel mit dem „richtigen“ Wert belegt.

3.6 Nachrichtenverzeichnis

Nachrichten der Datenquelle sind im Nachrichtenverzeichnis („`messages`“) in getrennten Chunks gespeichert. Die Chunk-Verzeichnisse haben alle den Namen

`chunk<ZEITSTEMPEL>`,

wobei der Zeitstempel dem der ersten Nachricht entspricht, die in diesem Verzeichnis vermerkt ist. Innerhalb der Verzeichnisse gibt es immer nur zwei Dateien: Die Nachrichtendatei und die dazu gehörige Nachrichten-Index-Datei.

Nachrichtendateien Eine Datei mit Nachrichten der Datenquelle heisst immer `messages`. Die Nachrichten der Datenquelle werden unverändert in diese Datei gespeichert, so dass diese nur einzelne XML-Tags enthält, die jeweils den Nachrichten entsprechen (siehe Kapitel 2.2.5).

Nachrichten-Index-Dateien Die Index-Datei `messages.idx` gehört zu der eigentlichen Nachrichtendatei und wird benötigt, um Nachrichten einer bestimmten Zeitspanne schnell laden zu können, ohne gleich die die gesamte Nachrichten-Datei einlesen zu müssen.

Ein Eintrag in einer Nachrinten-Index-Datei entspricht immer einer Nachricht in der Nachrichten-Datei. Die Struktur eines Eintrags zeigt Abbildung 6.

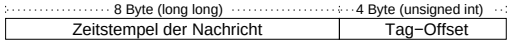

Abbildung 6: Eintrag einer Nachrichten-Index-Datei

4 Der *DLS Manager*

Der *DLS Manager* ist die grafische Benutzeroberfläche zur Konfiguration der Messaufträge in einem *DLS*-Datenverzeichnis. Er dient weiterhin zur Steuerung und Kontrolle der laufenden Erfassungsprozesse.

Wie auch der *dlsd* kann diesem Tool in der Kommandozeile mit dem Parameter `-d` das *DLS*-Datenverzeichnis mitgeteilt werden, auf dem es arbeiten soll (siehe Anhang D.3).

Beim Programmstart führt der *DLS Manager* selbstständig einige Prüfungen durch:

- Wenn das angegebene *DLS*-Datenverzeichnis ein noch leeres Verzeichnis ist, wird der Benutzer gefragt, ob darin eine gültige *DLS*-Datenverzeichnis-Struktur angelegt werden soll.
- Läuft zum angegebenen *DLS*-Datenverzeichnis noch keine Instanz des *dlsd*, wird der Benutzer gefragt, ob eine Instanz gestartet werden soll.

4.1 Hauptdialog

Abbildung 7 zeigt das Hauptfenster des *DLS Managers*, welches direkt nach dem Starten des Programmes sichtbar wird.

Abbildung 7: Hauptdialog des *DLS Managers*

4.1.1 Anzeigen

Im Hauptdialog werden die einzelnen Messaufträge als Zeilen in einer Tabelle angezeigt. Folgende Informationen stehen in den Spalten bereit:

- Auftrags-ID und Bezeichnung
  Die Beichnung des Auftrages ist willkürlich und kann jederzeit geändert werden. Die ID ist eine feste Nummer, die beim Anlegen des Auftrags automatisch gewählt wird. Alle Daten eines Auftrages werden im *DLS*-Datenverzeichnis im Unterverzeichnis `job<ID>` gespeichert.
- Quelle
  Der Name oder die IP-Adresse des Servers, der als Datenquelle dienen soll. Auf diesem muss ein *MSR*-Server über den TCP-Port 2345 zu erreichen sein. Die Quelle kann nur beim Erstellen des Auftrags gewählt werden und ist später nicht mehr veränderbar.
- Status
  Der vom Benutzer gewählte Status des Auftrags: „gestartet“, wenn die Erfassung laufen soll, sonst „angehalten“.
- Trigger
  Der Parameter der Datenquelle, der dem Erfassungsprozess als Trigger-Parameter dienen soll. Ist ein Trigger-Parameter gewählt, wird nur erfasst, wenn dieser den Wert 1 hat.

- Prozess
  Gibt an, ob derzeit ein Erfassungsprozess für diesen Auftrag läuft. Keine Anzeige, wenn der Auftrag angehalten ist.

- Erfassung
  Wenn ein Trigger konfiguriert ist, kann man hier erkennen, ob dieser gerade eingeschaltet ist. Ohne Trigger muss die Erfassung immer laufen, wenn auch der Erfassungsprozess läuft.

4.1.2 Interaktionen

- Die Zeilen mit den einzelnen Aufträgen können mit dem Mauscursor markiert werden.
- Ist ein Auftrag markiert, erscheint eine Schaltfläche zum Starten, bzw. Anhalten der Erfassung.
- Ein Doppelklick auf auf eine Auftrags-Zeile öffnet den Dialog zum Editieren des jeweiligen Auftrags (siehe Kapitel 4.3).
- Die Schaltfläche „Schliessen“ beendet das Programm.

4.2 Dialoge „Auftrag erstellen“ und „Auftrag ändern“

Das Aussehen des Dialoges zum Erstellen oder Ändern eines Messauftrages ist in Abbildung 8 zu sehen. Er erscheint nach Betätigen der Schaltflächen „Neuer Auftrag“ im Hauptdialog, oder „Ändern“ im Dialog „Auftrag bearbeiten“. Der Unterschied besteht darin, dass die Datenquelle nur beim Erstellen eines Auftrages geändert werden kann.

Abbildung 8: Dialog zum Erstellen oder Ändern eines Messauftrages

4.2.1 Anzeigen

Die Dialogmaske bietet dem Benutzer die Möglichkeit, mehrerere Angaben zum Messauftrag zu machen:

- Beschreibung
  Dies ist ein willkürlicher Name für den Messauftrag, der nur zum Zwecke der Wiedererkennung verwendet wird.

- Quelle
  Die Adresse der Datenquelle. Dies kann ein Hostname oder eine IP-Adresse sein. Wird ein Hostname verwendet, so muss sichergestellt sein, dass dieser vom *dlsd* zur Laufzeit in eine IP-Adresse aufgelöst werden kann. Auf dem angegebenen Host muss zur Erfassung und zum Hinzufügen von Kanälen über den entsprechenden Dialog (siehe Kapitel 4.4) eine Datenquelle vorhanden sein.

- Trigger
  Der Name des Parameters, der als Trigger-Parameter dienen soll. Ist hier ein Trigger-Parameter angegeben, so wird später nur erfasst, wenn dieser den Wert 1 hat. Wird dieses Eingabefeld leer gelassen, so wird kein Trigger verwendet.

- Zeit-Quota
  Die Zeit-Quota (Länge des maximal zu speichernden Zeitraumes erfasster Daten, siehe Kapitel 2.2.4) kann hier in einer Kombination aus einem (ganzzahligen) Wert und der dazugehörigen Zeiteinheit eingestellt werden. Kein Eintrag im Eingabefeld für den Wert bedeutet, dass keine Zeit-Quota verwendet werden soll.

- Daten-Quota
  Die Daten-Quota (Größe des maximal zu verwendenden Speicherplatzes, der im Dateisystem für die erfassten Daten reserviert werden soll) kann hier ebenfalls in einer Kombination aus einer Ganzzahl und der dazugehörigen Größeneinheit angegeben werden. Kein Eintrag bedeutet wieder, dass keine Daten-Quota verwendet werden soll.

4.2.2 Interaktionen

- Betätigen der Schaltfläche „OK“ (oder Drücken der *Enter*-Taste) prüft die angegebenen Daten. Sind diese fehlerhaft, wird ein Fenster mit den genauen Fehlermeldungen angezeigt. Andernfalls werden die Daten übernommen und der Dialog geschlossen. Läuft ein *dlsd* -Erfassungsprozess, wird dieser zur Übernahme der neuen Vorgaben aufgefordert.

- Betätigen der Schaltfläche „Abbrechen“ verwirft die angegebenen Daten und schliesst den Dialog.

4.3 Dialog „Auftrag bearbeiten“

Abbildung 9 zeigt den Dialog zum Bearbeiten eines Auftrages, der nach einem Doppelklick auf einen Auftrag im Hauptdialogfenster erscheint.

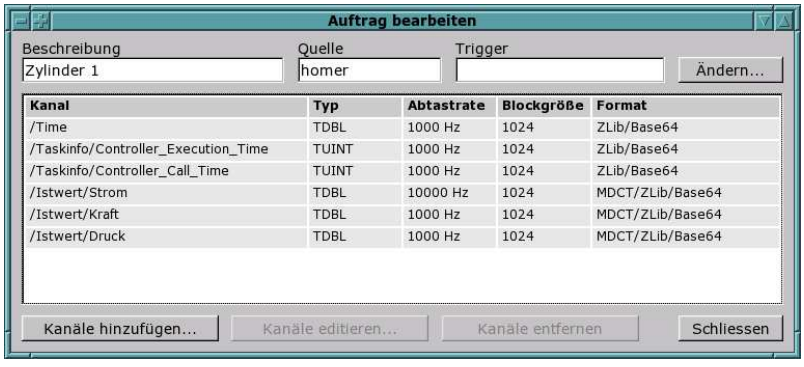

Abbildung 9: Dialog zum Bearbeiten eines Auftrages

4.3.1 Anzeigen

Im oberen Bereich werden ausgewählte Stammdaten zum Auftrag angezeigt. Darunter befindet sich die Liste der zu erfassenden Kanäle mit den wichtigsten Parametern. Diese sind:

- Kanal
  Der Name des zu erfassenden Kanals

- Typ
  Der Kanaltyp. Ein Kanal kann entweder einen ganzzahligen oder einen Gleitkommatyp haben (siehe Anhang B).

- Abtastrate
  Die Frequenz, mit der die einzelnen Messwerte gespeichert werden sollen.

- Blockgröße
  Die Anzahl der Messwerte, die zusammen in einer Einheit komprimiert und gespeichert werden sollen.

- Format
  Die Kompressionsmethode (siehe Kapitel 6).

4.3.2 Interaktionen

- Der Dialog zum Editieren der Auftrags-Stammdaten kann über die Schaltfläche „Ändern“ aufgerufen werden.

- Die Kanalzeilen der Tabelle können mit dem Mauscursor markiert werden. Dann sind auch die Schaltflächen „Kanäle editieren...“ und „Kanäle entfernen“ anwählbar.

- Durch Drücken der *Shift*-, bzw. *Strg*-Tasten können in der Tabelle auch mehrere Kanäle gleichzeitig angewählt werden, die dann auch gleichzeitig editiert oder entfernt werden können.

- Die Vorgaben zu einem oder mehrerer, markierter Kanäle können durch Betätigen der Schaltfläche „Kanäle editieren...“ im darauf folgenden Dialog bearbeitet werden. Beim Editieren mehrerer Kanäle gelten allerdings besondere Bedingungen (siehe Kapitel 4.5.3).

- Ein Doppelklick auf eine Kanalzeile öffnet ebenfalls den Dialog zum Editieren der Vorgaben des entsprechenden Kanals.

- Durch Betätigen der Schaltfläche „Kanäle entfernen“ werden alle markierten Kanäle aus den Vorgaben gelöscht. Erfasste Daten bleiben aber weiterhin verfügbar.

- Beim Betätigen der Schaltfläche „Kanäle hinzufügen“ öffnet sich der Dialog zum Hinzufügen von Kanalvorgaben (siehe Kapitel 4.4).

- Die Schaltfläche „Schliessen“ beendet den Dialog und kehrt zum Hauptdialog zurück.

4.4 Dialog „Kanäle hinzufügen“

Nach dem Betätigen der Schaltfläche „Kanäle hinzufügen“ im Dialog zum Bearbeiten eines Auftrages öffnet sich ein Dialogfenster, wie in Abbildung 10 dargestellt. Gleichzeitig wird versucht, eine TCP-Verbindung zur Datenquelle aufzubauen, um deren Kanäle abzurufen.

4.4.1 Anzeigen

- Während versucht wird, die Verbindung mit der Datenquelle aufzubauen, erscheint eine Meldung „Empfange Kanäle...“. Kann die Verbindung auch nach einer festgelegten Wartezeit nicht aufgebaut werden, erscheint ein Fenster mit der entsprechenden Fehlermeldung und der Dialog schliesst sich.

- Wurde die Kanalliste erfolgreich abgerufen, erscheinen die einzelnen Kanäle in einer Tabelle. Dort sind Der Kanalname, die Einheit, die maximale Abtastfrequenz und der Kanal-Datentyp angegeben.

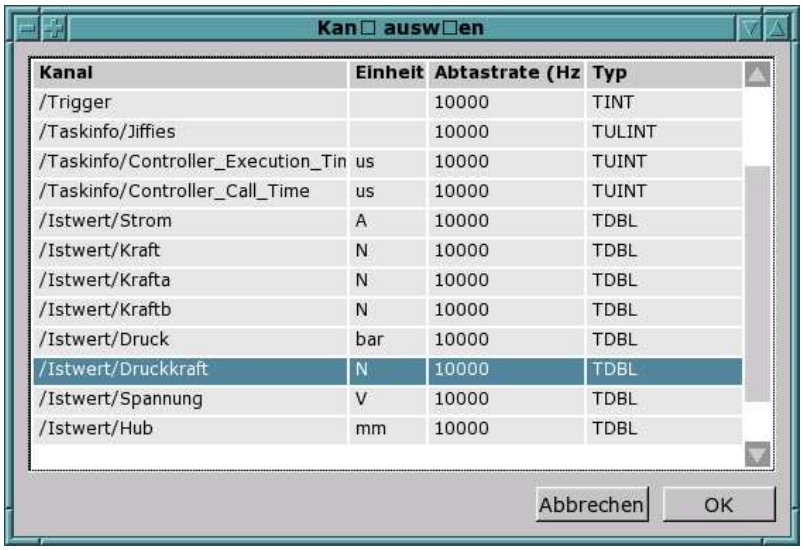

Abbildung 10: Dialog zum Hinzufügen von Kanalvorgaben

4.4.2 Interaktionen

- Der Benutzer kann einzelne Kanäle mit dem Cursor selektieren. Durch Drücken der *Strg*- bzw. *Shift*-Tasten lassen sich auch mehrere Kanäle gleichzeitig anwählen.
- Bei Betätigen der Schaltfläche „OK“ werden alle angewählten Kanäle in die Liste der Kanalvorgaben des Auftrags eingefügt. Ist ein Kanal schon vorhanden, so wird dies in einem Fenster mit der entsprechenden Warnmeldung angezeigt. Falls Änderungen durchgeführt wurden, wird ein evtl. laufender *dlsd*-Erfassungsprozess zur Übernahme der neuen Kanalvorgaben aufgefordert.
- Das Betätigen der Schaltfläche „Abbrechen“ schliesst den Dialog, ohne neue Kanalvorgaben zum Auftrag hinzuzufügen.

4.5 Dialog „Kanäle bearbeiten“

Der in Abbildung 11 dargestellte Dialog zum Bearbeiten von Kanalvorgaben erscheint bei einem Doppelklick auf eine Kanalvorgabe im Dialog zum Bearbeiten eines Auftrags, oder nach Betätigen des Schaltfläche „Kanäle bearbeiten“, nachdem ein oder mehrere Kanäle selektiert wurden.

Abbildung 11: Dialog zum Bearbeiten von Kanalvorgaben

4.5.1 Anzeigen

Folgende Eingabefelder stehen dem Benutzer zur Parametrisierung der Kanalvorgabe(n) bereit:

- Abtastrate
  Die Anzahl der Werte, die pro Sekunden gespeichert werden sollen. Diese Rate muss ein ganzzahliger Teiler der maximalen Abtastfrequenz des betroffenen Kanales sein.

- Blockgröße
  Die Anzahl der Werte, die in einem Block komprimiert und gespeichert werden. Je größer der Block, desto besser kann u. U. komprimiert werden, desto länger dauert allerdings auch der Kompressionsvorgang. Die Blockgröße muss bei einer MDCT-basierten Kompression ein ganzzahliges Vielfaches der MDCT-Blockgröße sein.

- Meta-Maske
  Dieser Wert kann momentan nicht editiert werden
  Die Meta-Maske ist eine Bitmaske, die die Arten der zu speichernden Meta-Daten angibt. Siehe dazu Kapitel 2.2.2.

- Untersetzung
  Die Meta-Untersetzung ist die Anzahl Datenwerte einer Meta-Ebene aus denen ein neuer Meta-Wert der nächsthöheren Ebene generiert wird. Siehe dazu ebenfalls Kapitel 2.2.2. Dieser Wert bedarf normalerweise keiner Anpassung.

- Format
  Das Kompressionsformat, mit dem die erfassten Daten vor der Speicherung komprimiert werden. Je nach Kompressionsverfahren sind noch weitere Parameter anzugeben.

- MDCT-Blockgröße
  Dieser Parameter ist bei MDCT-Basierten Kompressionsverfahren mit anzugeben und verhält sich ähnlich wie die Blockgröße. Siehe Kapitel 6.2.

- Genauigkeit
  Manche, verlustbehaftete Kompressionsverfahren erlauben es, den maximalen, absoluten Fehler anzugeben. Der Fehler muss immer in der Einheit des entsprechenden Kanales angegeben werden.

4.5.2 Interaktionen

- Betätigen der Schaltfläche „OK“ prüft die angegebenen Parameter zuerst auf Plausibilität. Schlägt dies fehl, wird ein Fenster mit den entsprechenden Fehlermeldungen angezeigt. Ansonsten werden alle angegebenen Parameter bei den zuvor selektierten Kanalvorgaben angewendet, der evtl. laufende *dlsd*-Erfassungsprozess zur Übernahme der neuen Parameter aufgefordert und der Dialog geschlossen.

- Betätigen der Schaltfläche „Abbrechen“ führt keine Änderungen durch und schliesst den Dialog.

4.5.3 Gleichzeitiges Editieren mehrerer Kanalvorgaben

Der Dialog zum Bearbeiten von Kanalvorgaben (Abbildung 11) kann auch dazu genutzt werden, mehrere Kanäle gleichzeitig zu editieren. In diesem Fall gilt Folgendes:

- Beim Öffnen des Dialoges werden alle Parameter, die zu diesem Zeitpunkt bei allen zu ändernden Kanalvorgaben gleich sind, in der Dialogmaske angezeigt. Bei Parametern, die **nicht** bei allen Kanalvorgaben gleich sind, bleiben die entsprechenden Eingabefelder leer.

- Das Verändern oder Eintragen eines Wertes in der Dialogmaske führt dazu, dass dieser Wert nach Betätigen der „OK“-Schaltfläche bei **allen** zuvor selektierten Kanalvorgaben eingetragen wird.

- Ist ein Wert in der Maske bei Betätigen der „OK“-Schaltfläche leer, so wird dieser bei **keiner** Kanalvorgabe verändert. Alle zuvor selektierten Kanäle behalten in diesem Fall den entsprechenden, vorherigen Wert. So ist es möglich bei einer Reihe von Kanalvorgaben z. B. nur die Abtastrate zu ändern.

- Die Parameter der Kompression (*Format*, *MDCT-Blockgröße* und *Genauigkeit*) sind diesbezüglich eine Einheit. Das bedeutet, dass die Kompressionsparameter anfangs nur gezeigt werden, wenn sie bei **allen** Kanalvorgaben exakt gleich sind. Respektive können die genannten Kompressionsparameter auch nur alle gemeinsam geändert werden.

5 Der Betrachter *DLS View*

Das Programm *DLS View* dient zur einfachen Ansicht der erfassten Daten eines Auftrages. Es besteht deshalb lediglich aus einem einzelnen Dialog (siehe Abbildung 12).

5.1 Hauptdialog

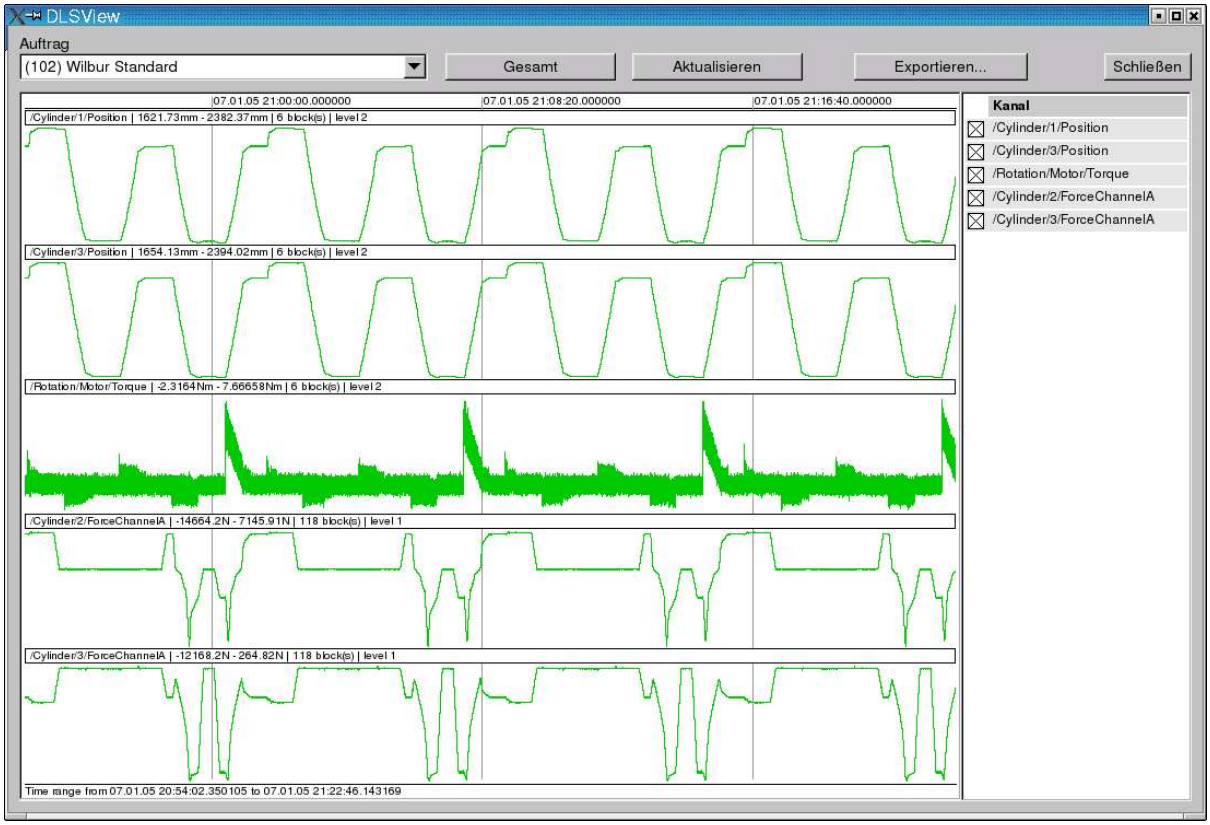

Abbildung 12: Hauptdialog des Betrachters *DLS View*

5.1.1 Anzeigen

- Am oberen Rand befindet sich links ein Auswahlfeld, in dem der Benutzer den Messauftrag anwählen kann, zu dem Daten angezeigt werden sollen.
- Rechts befindet die Liste der Kanäle, die zu dem gewählten Auftrag erfasst wurden.
- Den Großteil des Dialogfensters macht die Datenanzeige aus. Diese ist so aufgebaut, dass die Daten mehrerer Kanäle übereinander auf einer gemeinsamen Zeitskala angezeigt werden können, die am oberen Rand zu sehen ist. Die Skalenstriche erscheinen als senkrechte, graue Linien im Koordinatensystem.

- Ganz unten in der Anzeige befindet sich eine schmale Textzeile, in der die jeweils angezeigte Zeitspanne zu sehen ist.
- Über jedem, angezeigten Kanal ist eine schmale Kopfzeile zu sehen, in der der Kanalname, der angezeigte Wertebereich, die Anzahl der geladenen Datenblöcke und die geladenen Meta-Ebenen (siehe Kapitel 2.2.2) angezeigt werden. Darunter folgen jeweils die Daten. Eine blaue Kurve bedeutet, dass es sich bei den angezeigten Daten um generische Daten (Meta-Ebene 0) handelt, wurde zur Anzeige eine höhere Meta-Ebene verwendet, ist der entsprechende Kurvenabschnitt grün.
- Wurden in einer Zeitspanne keine Daten erfasst, wird dieser in der betroffenen Kanalzeile hellgelb hinterlegt (siehe Abbildung 13).
- Da alle Kanalzeilen sich die Gesamthöhe der Anzeige teilen müssen, werden die Kanalzeilen mit zunehmender Anzahl schmaler. Unterschreitet die Höhe einer einzelnen Kanalzeile einen festgelegten Wert, wird rechts eine Scroll-Leiste eingeblendet.

5.1.2 Interaktionen

- In der Auswahlliste „Auftrag“ kann der Benutzer den Auftrag wählen, dessen erfasste Daten er anzeigen möchte. Daraufhin wird die Liste der erfassten Kanäle aktualisiert.
- Der Benutzer kann durch Markieren der Kästchen vor den Kanalnamen in der Kanalliste einzelne Kanäle in der Datenanzeige ein- oder ausblenden.
- Durch Betätigen der Schaltfläche „Gesamt“ wird die Gesamtzeitspanne, in der zu den gewählten Kanälen Daten erfasst wurden, ermittelt und angezeigt.

  Werden unmittelbar danach Kanäle hinzugefügt oder entfernt, wird immer wieder die neue Gesamtzeitspanne ermittelt. Erst wenn der Benutzer explizit eine andere Zeitspanne zur Anzeige wählt, bleibt diese auch beim Hinzufügen oder Entfernen von Kanälen konstant.
- Das betätigen der Schalfläche „Aktualisieren“ liest alle Kanaldaten der eingestellten Zeitspanne neu ein und zeigt diese an.
- *Die Schaltfläche „Exportieren“ ist momentan nicht benutzbar*
- Durch Drücken und Halten der linken Maustaste im Datenbereich der Anzeige kann der Benutzer eine neue Zeitspanne anwählen, die während des Haltens der Maustaste durch zwei senkrechte, rote Linien markiert wird. Diese sind am oberen Rand mit den genauen Zeitpunkten beschriftet. Durch Loslassen der linken Maustaste wird der neue Zeitbereich übernommen und es werden die entsprechenden Daten geladen.

  Übrigens kann die Position beim Loslassen der Maustaste durchaus auch ausserhalb des Anzeigebereiches liegen, womit eine leichte Vergrößerung der angezeigten Zeitspanne möglich ist.
- In ähnlicher Weise kann durch Drücken und Halten der *rechten* Maustaste auf dem Anzeigebereich die angezeigte Zeitspanne verschoben werden. Loslassen der Maustaste bewirkt die Übernahme der neuen Zeitspanne.
- Ein Doppelklick in den Datenbereich bewirkt, dass die angezeigte Zeitspanne um den Faktor 2 erweitert wird. Vorher wird sie um den angeklickten Zeitpunkt zentriert. Ist während des Doppelklicks die *Shift*-Taste gedrückt, wird für die Erweiterung der Faktor 10 verwendet.
- Hat der Daten-Bereich den Tastatur-Fokus, bewirkt das Drücken der *Strg*-Taste, dass an dem Zeitpunkt, auf den der Mauscursor zeigt, eine vertikale „Scanlinie“ gezeichnet wird (siehe Abbildung 13). Schneidet diese Linie eine angezeigte Kurve, so wird der Datenwert am Schnittpunkt angezeigt.

  Da die Scanlinie nicht unendlich schmal ist, fallen evtl. viele Datenwerte in den Bereich, den sie überdeckt. In diesem Fall wird der gesamte Wertebereich der „unter“ der Scanlinie liegenden Werte mit zwei horizontalen Linien markiert, die dann dem Minimum und dem Maximum entsprechen (siehe 3. Kanal in Abbildung 13).

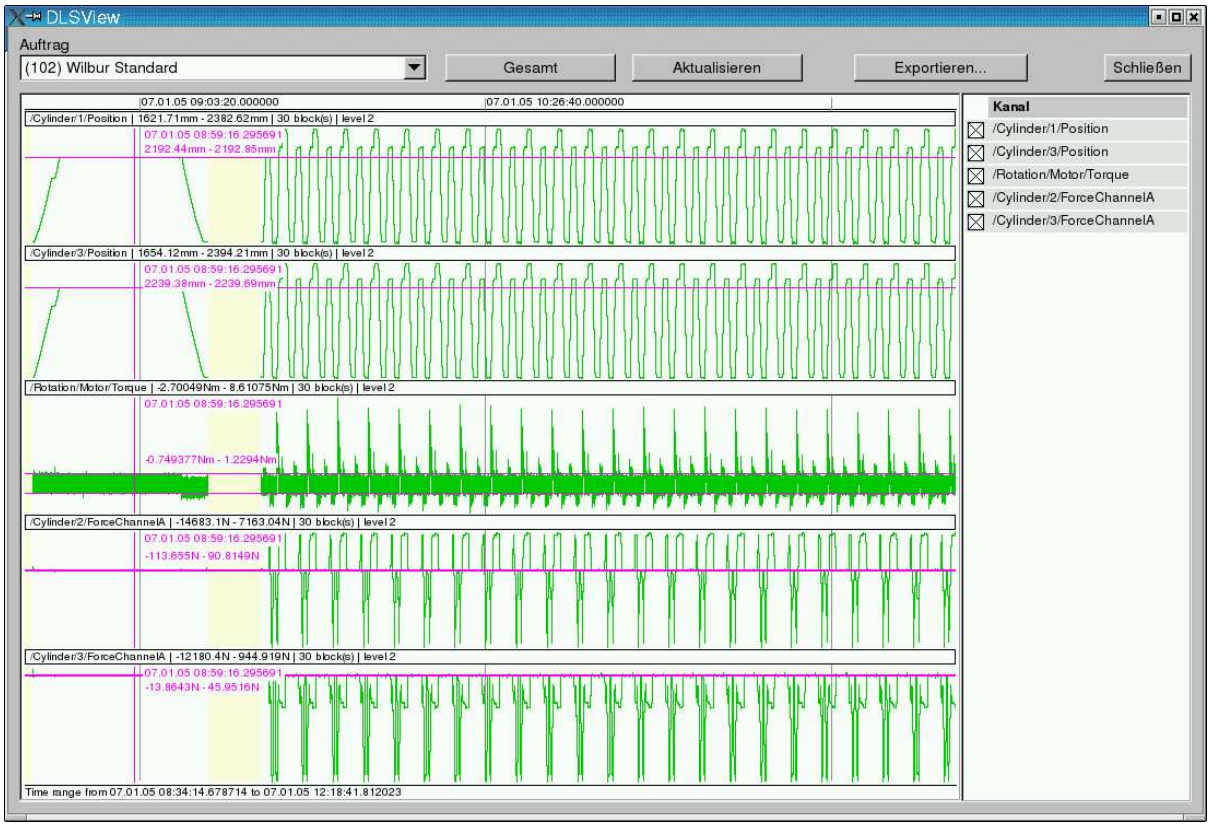

Abbildung 13: Scanlinien bei gedrückter *Strg*-Taste

6 Kompressionsmethoden

Die Kompression der von der Datenquelle empfangenen Messdaten dient zur Verkleinerung des benötigten Speicherplatzes im Dateisystem. Sie findet immer Blockweise statt (d. h. es werden immer eine bestimmte Anzahl Datenwerte gemeinsam komprimiert), wobei der Benutzer die Blockgröße in den Kanalvorgaben einstellen kann. Man unterscheidet grundsätzlich zwischen verlustfreier und verlustbehafteter Kompression.

Das *DLS*-System unterstützt verschiedene Kompressionsalgorithmen. Da die meisten Algorithmen binäre Daten als Ausgabe haben, werden diese vor der Speicherung in die Datendateien in *Base64* kodiert. Das hat zwar zur Folge, dass der benötigte Speicherplatz um ein Drittel vergrößert wird, hat aber zum Vorteil, dass die komprimierten Daten in „druckbaren“ Zeichen vorliegen, und somit in XML kodiert werden können. Deswegen haben alle Kompressionsmethoden des *DLS* das Suffix `/Base64`.

Folgende Kompressionsmethoden werden vom *DLS* unterstützt:

- `ZLib/Base64`
  Ein einfaches, aber effizientes Kompressionsverfahren, dass eine verlustfreie Kompression bietet. Siehe Kapitel 6.1.

- `MDCT/ZLib/Base64`
  Ein erweitertes Kompressionsverfahren, dass die Daten durch eine Transformation und eine Quantisierung aufbereitet und dann erst komprimiert. Siehe Kapitel 6.2.

6.1 Kompression mit der ZLib

Die Bibliothek „ZLib“ (*http://www.gzip.org/zlib*) stellt Funktionen zur verlustfreien Kompression von Daten bereit. Der *dlsd*-Erfassungprozess nutzt diese in den Kompressionsverfahren „`ZLib/Base64`“ und „`MDCT/ZLib/Base64`“ um die „rohen“ oder bereits aufbereiteten Daten blockweise zu komprimieren. Diese werden danach in *Base64* kodiert und abgespeichert.

6.2 Kompression mit der MDCT

Das Kompressionverfahren der „modifizierten, diskreten Cosinus-Transformation“ (MDCT) des *DLS* ist eigentlich ein hybrides Verfahren in dem die Daten zuerst mit der MDCT transformiert, dann quantisiert und schliesslich bitweise transponiert werden, um die darauf folgende Kompression mit der ZLib effektiver zu machen. Dabei wird das Ziel verfolgt, die Messdaten zwar verlustbehaftet, aber mit begrenztem Fehler zu komprimieren.

6.2.1 MDCT

Die MDCT eine Art diskretes Äquivalent zur Fourier-Transformation, dass ein Signal in die entsprechende Repräsentation als Koeffizienten von harmonischen Schwingungen transformiert, die überlagert wieder das Originalsignal ergeben.

Während die „normale“ DCT darauf basiert, dass immer n Werte in n Koeffizienten transformiert werden, aus denen dann das Originalsignal vollständig wiederhergestellt werden kann, werden bei der „modifizierten“ DCT immer n Werte in $\frac{n}{2}$ Koeffizienten transformiert, die an sich eine unvollständige Repräsentation des Originalsignales sind. Da die Transformation aber zu 50% überlappend durchgeführt wird, kann das Originalsignal durch erneutes Überlappen zweier aufeinanderfolgender, rücktransformierter Sequenzen von Koeffizienten wiederhergestellt werden. Dieses Verfahren ist in Abbildung 14 dargestellt.

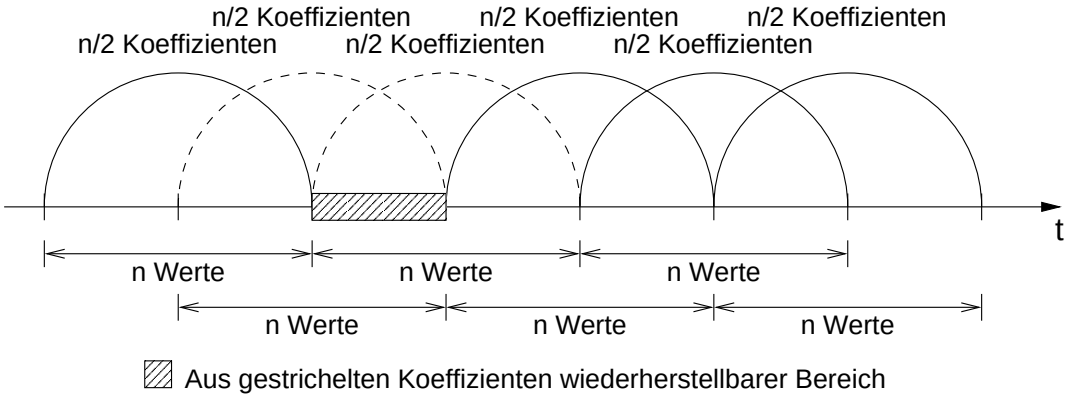

Abbildung 14: Modifizierte, diskrete Cosinus-Transformationen

Diese modifizierte DCT wird ergänzt durch das Überlagern der zu transformierenden Werte mit einer Fensterfunktion, die die am Rand liegenden Werte geringer gewichtet. So wird verhindert, dass die unterschiedlichen Fehler an den Randbereichen der Einzeltransformationen zu „Artefakten“ führen und das Originalsignal sich nahtlos wiederherstellen lässt.

6.2.2 Quantisierung

Die durch die MDCT ermittelten Koeffizienten werden einer Integer-Quantisierung unterzogen. Dabei wird mit einem Bisektionsverfahren entschieden, wie viele Bit minimal zur Quantisierung herangezogen

werden können, ohne das der maximale Fehler bei einer Rücktransformation zu groß würde. Das Quantisierungsverfahren liefert neben den auf n Bits quantisierten Koeffizienten auch noch den Gleitkomma-Skalierungsfaktor, der zur Wiederherstellung der Originalkoeffizienten benötigt wird.

Da der Maximale Fehler bei einer einzelnen, inversen MDCT nicht genau bestimmt werden kann. Wird dieser durch die Hälfte des vorgegebenen Fehlers abgeschätzt. Wenn zwei Sequenzen von Rücktransformierten Koeffizienten zur Wiederherstellung des Originalsignales überlagert werden, kann der vorgegebene Fehler nicht überschritten werden, wenn der absolute Fehler in beiden Teilsignalen die Hälfte des maximalen Gesamtfehlers nicht überschreitet.

Signale aus technischen Prozessen eignen sich oft sehr gut für die Transformation mit der MDCT, da sie in den meisten Fällen auf überlagerte, harmonische Schwingungen zurückzuführen sind. Das bedeutet, dass die hochfrequenten Anteile der entsprechenden Koeffizienten oft nicht stark ausgeprägt sind und durch die Quantisierung weitestgehend angeglichen werden können. Das führt später zu einer guten Kompressiblität.

6.2.3 Transponierung

Die Transponierung wird deshalb benötigt, weil das Kompressionsverfahren ZLib byteweise arbeitet und ähnliche Bitmuster in den meisten Fällen nicht erkennen kann. Also werden die einzelnen Bits der quantisierten Koeffizienten im Speicher umsortiert. Dazu werden die Koeffizienten zuerst von ihren Vorzeichnen getrennt. Die Vorzeichenbits werden separat vor den Koeffizientenbits abgelegt. Dann folgen zuerst alle *MSB*'s („most significant bits“) der Koeffizienten und zuletzt alle *LSB*'s („least significant bits“). So entstehen durch die meist sehr kleinen Koeffizienten der höherfrequenten Schwingungen viele Null-Bytes, die sich gut komprimieren lassen.

Ein Komprimierter MDCT-Block besteht also aus dem Skalierungsfaktor der quantisierten Koeffizienten (4 Bytes, bzw. 8 Bytes), der Anzahl q der benutzten Quantisierungsbits (1 Byte), n Vorzeichnenbits und schliesslich $q \cdot (n-1)$ Koeffizientenbits.

6.2.4 MDCT über schnelle FFT

Marios Athineos[1] hat ein Verfahren entwickelt, um eine MDCT über n Werte auf eine Fourier-Transformation über $\frac{n}{4}$ Werte zu reduzieren. Der *DLS* benutzt dieses Verfahren in Kombination mit der *FFTW*-Bibliothek (siehe Anhang A) um den Rechenaufwand erheblich zu mindern. Diese Bibliothek vereint effiziente Algorithmen zur Berechnung der Fourier-Transformation mit der Benutzung von Prozessor-Erweiterungen wie MMX oder DDE.

[1] marios@ee.columbia.edu, http://www.ee.columbia.edu/~marios, Columbia University

Anhang

A Systemvoraussetzungen

Der *DLS* ist größtenteils in der Programmiersprache C++ implementiert. Er benötigt zum Kompilieren und Laufen ein Linux-Betriebssystem.

Folgende Software muss ebenfalls zum Kompilieren und zur Laufzeit installiert sein:

- Für das Aufzeichnen der während der Laufzeit anfallenden Nachrichten wird der *syslogd* verwendet, der standardmäßig jeder Linux-Distribution beiliegt.
- Für das Kompilieren der grafischen Benutzeroberflächen *DLS Manager* und *DLS View* ist zusätzlich die GUI-Bibliothek *FLTK* in der Version 1.1.*X* nötig, die auf der *FLTK*-Homepage

  http://www.fltk.org

  heruntergeladen werden kann.
- Für die Kompression wird die *ZLib* benötigt. Diese ist in nahezu jeder Linux-Distribution enthalten. Ansonsten kann sie von der *ZLib*-Homepage

  http://www.gzip.org/zlib

  heruntergeladen werden.
- Ebenfalls für die Kompression wird die *FFTW*-Bibliothek benötigt. Mit dieser ist der *DLS* in der Lage die für die MDCT-Kompression nötigen Fourier-Transformationen noch schneller zu berechnen. Die Bibliothek kann von der offiziellen Homepage:

  http://www.fftw.org/download.html

  heruntergeladen werden.
- Für den *DLS Manager* und für *FLTK* wird die *pthreads*-Bibliothek benötigt.

B Datentypen

Tabelle 1 zeigt alle bisher unterstützten Kanal-Datentypen und die jeweils möglichen Kompressionsmethoden.

Tabelle 1: Unterstützte Kanal-Datentypen

Typ	Beschreibung	Kompression
`TCHAR`	1 Byte Ganzzahl (mit Vorzeichen)	ZLib/Base64
`TUCHAR`	1 Byte Ganzzahl (ohne Vorzeichen)	ZLib/Base64
`TINT`	4 Byte Ganzzahl (mit Vorzeichen)	ZLib/Base64
`TUINT`	4 Byte Ganzzahl (ohne Vorzeichen)	ZLib/Base64
`TLINT`	4 Byte Ganzzahl (mit Vorzeichen)	ZLib/Base64
`TULINT`	4 Byte Ganzzahl (ohne Vorzeichen)	ZLib/Base64
`TFLT`	4 Byte Gleitkomma	ZLib/Base64, MDCT/ZLib/Base64
`TDBL`	8 Byte Gleitkomma	ZLib/Base64, MDCT/ZLib/Base64

C PID-Dateien

Das *DLS*-System verwendet an mehreren Stellen sog. *PID*-Dateien, einen Mechanismus, der verhindern soll, dass für eine konkrete Aufgabe mehrere Prozesse laufen. Eine *PID*-Datei enthält die *ASCII*-kodierte Prozess-ID (*PID*) des aktuell laufenden Prozesses. Nach jedem Prozessstart wird deshalb ermittelt, ob die entsprechende Datei und evtl. schon ein Prozess mit der dort angegebenen PID existiert. Wenn beides zutrifft, darf kein neuer Prozess gestartet werden. Der Prozess muss sich sofort beenden. Existiert keine andere Instanz, kann der neu gestartete Prozess weiterlaufen un eine neue PID-Datei erstellen. Eine veraltete *PID*-Datei (d. h. der angegebene Prozess existiert nicht mehr) kann zuvor gelöscht werden.

D Kommandozeilenparameter

D.1 dlsd

```
dlsd build 56 (DLS version 0.9.2) - Mar 11 2005, 12:29:04 - built by fp
Aufruf: dlsd [OPTIONEN]
        -d [Verzeichnis]   DLS-Datenverzeichnis angeben
        -k                 Nicht von der Konsole trennen
        -h                 Diese Hilfe anzeigen
```

D.2 dls

```
Aufruf: dls [OPTIONEN] [KOMMANDO]
Optionen:
        -d [Verzeichnis]   DLS-Datenverzeichnis
        -h                 Diese Hilfe anzeigen
Kommandos:
        status             Status anzeigen (Default)
        start              dlsd starten
        stop               dlsd anhalten
        restart            dlsd neu starten
```

D.3 dls_ctl

```
dls_ctl build 51 (DLS version 0.9.2) - Mar 11 2005, 12:29:17 - built by fp
Aufruf: dls_ctl [OPTIONEN]
        -d [Verzeichnis]   DLS-Datenverzeichnis angeben
        -h                 Diese Hilfe anzeigen
```

D.4 dls_view

```
dls_view build 51 (DLS version 0.9.2) - Mar 11 2005, 12:29:31 - built by fp
Aufruf: dls_view [OPTIONEN]
        -d [Verzeichnis]   DLS-Datenverzeichnis angeben
        -h                 Diese Hilfe anzeigen
```

D.5 dls_quota

```
Aufruf: dls_quota [OPTIONEN]
        -d [Verzeichnis]   DLS-Datenverzeichnis
        -i [Sekunden]      Überprüfungs-Intervall
        -k                 Kein Daemon werden
        -h                 Diese Hilfe anzeigen
```

Index