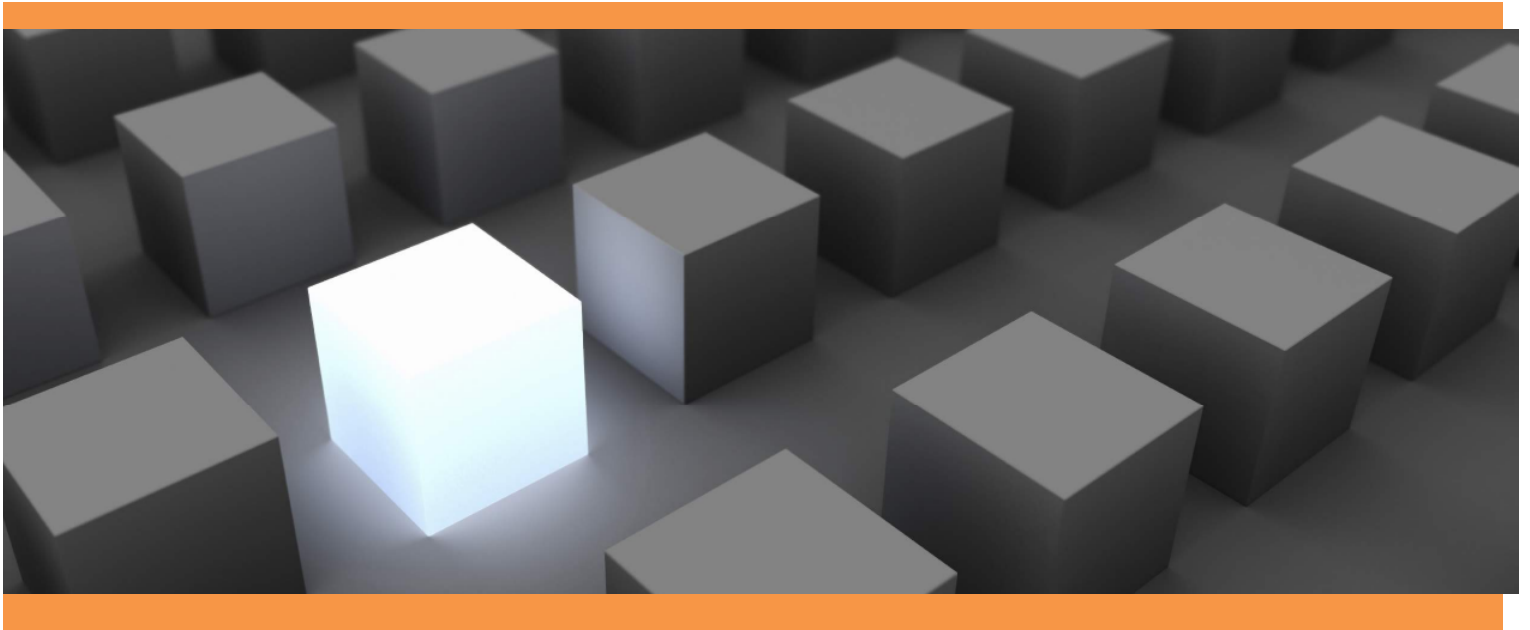


SYMEO

ABSOLUTE POSITIONING

SYMEO LPR 2D Protokoll

Protokollbeschreibung



LPR 2D Protokoll

Inhalt

1	GRUNDLAGEN	6
2	SCHNITTSTELLEN	7
3	BINÄRFORMAT DES PROTOKOLLS	8
3.1	DATENTYPEN	8
3.2	BYTESTUFFING	8
3.3	ALLGEMEINER AUFBAU	9
3.4	DATENFELDER.....	11
3.4.1	START	11
3.4.2	END.....	11
3.4.3	TIMESTAMP.....	12
3.4.4	POSITION.....	12
3.4.5	VELOCITY	13
3.4.6	ORIENTATION.....	13
3.4.7	POSITION-ERROR	14
3.4.8	VELOCITY-ERROR.....	14
3.4.9	ORIENTATION-ERROR	15
3.4.10	USER-DATA.....	15

3.4.11	SYSTEM-ERROR.....	16
3.4.12	SATELLITE-STATE.....	16
3.5	CRC.....	17
4	ASCII-FORMAT DES PROTOKOLLS	18
4.1	DATENTYPEN	18
4.2	ALLGEMEINER AUFBAU	18
4.3	DATENFELDER.....	20
4.3.1	START.....	20
4.3.2	END.....	20
4.3.3	TIMESTAMP.....	21
4.3.4	POSITION.....	21
4.3.5	VELOCITY	22
4.3.6	ORIENTATION.....	23
4.3.7	POSITION-ERROR	24
4.3.8	VELOCITY-ERROR.....	24
4.3.9	ORIENTATION-ERROR	25
4.3.10	USER-DATA.....	25
4.3.11	SYSTEM-ERROR.....	26
4.3.12	SATELLITE-STATE.....	27
4.3.13	CRC.....	28

5	BITMASKE SELECTED-FIELDS.....	28
6	CRC BERECHNUNG	29
7	FEHLERCODES.....	30

History

Version	Datum	Beschreibung
2.2	09.12.2008	Layout geändert
2.3	16.04.2009	Satellitenfelder hinzugefügt

1 Grundlagen

Das 2D Protokoll bildet die Schnittstelle eines LPR 2D Systems zum Anwender. Das Protokoll wurde so entwickelt, dass es ohne großen Aufwand zu parsen ist. Dabei wurden folgende Gesichtspunkte berücksichtigt:

- einheitliches Datenpaket
Bei diesem Protokoll gibt es nur ein einzelnes Datenpaket fester Länge.
- Konfigurierbarkeit
Der Aufbau des Datenpakets kann durch den Anwender konfiguriert werden. Nur die gewünschten Datenfelder werden ausgegeben. Durch Ausgabe nur der benötigten Datenfelder reicht auch eine langsame Schnittstelle aus.

Das Protokoll arbeitet unidirektional, es erfolgt nur eine Ausgabe vom LPR 2D System. Eine Kommunikation vom Anwender zum LPR 2D System ist nicht vorgesehen.

Das Protokoll kann wahlweise im Binär- oder ASCII-Format gesendet werden. Der allgemeine Aufbau eines Datenpakets unterscheidet sich allerdings nicht zwischen diesen beiden Formaten.

Zur Konfiguration des Protokolls existiert eine Konfigurationsdatei, in der die notwendigen Einstellungen des Protokolls vorgenommen werden.

Folgender Eintrag muss in der Konfigurationsdatei des Protokolls vorhanden sein, um das **Format des Protokolls** auszuwählen:

```
[Format]  binary
```

bzw.

```
[Format]  ascii
```

2 Schnittstellen

Das 2D Protokoll kann auf folgenden Schnittstellen angewandt werden:

- RS-232 (serielle Schnittstelle)
- TCP/IP
- UDP

Dabei ist nicht jede Kombination Schnittstelle – Protokollformat sinnvoll. Wird zum Beispiel das Binärprotokoll mit deaktiviertem Bytestuffing auf einer seriellen Schnittstelle angewendet, kann der Empfänger nicht sicher den Beginn eines neuen Datenpakets erkennen.

Bei Wahl der seriellen Schnittstelle muss außerdem eine ausreichend hohe Baudrate eingestellt werden.

Folgender Eintrag muss in der Konfigurationsdatei des Protokolls vorhanden sein, um die **Schnittstelle des Protokolls** auszuwählen:

```
[Interface]    serial
bzw.
[Interface]    tcp
bzw.
[Interface]    udp
```

Bei Wahl der seriellen Schnittstelle muss zusätzlich noch der jeweilige COM-Port angegeben werden, z.B.:

```
[Port]        1
oder
[Port]        /dev/tty0
```

Bei Wahl der TCP/IP oder UDP-Schnittstelle muss ebenfalls ein Port angegeben werden, z.B.:

```
[Port]        1234
```

Im TCP/IP Betrieb arbeitet das Programm dieses Protokolls als Server, d.h. der Empfänger muss sich auf den Server-Port verbinden. Im UDP-Betrieb arbeitet das Programm dagegen als Sender, als Port muss daher der Zielport des Empfängers angegeben werden. Außerdem muss bei UDP-Betrieb auch die Zieladresse (IP-Adresse) angegeben werden, z.B.:

```
[Destination] 192.168.255.255
```

3 Binärformat des Protokolls

Beim Binärformat des 2D Protokolls wird das gesamte Datenpaket binär übertragen. Damit ist das Binärformat kompakter als das ASCII-Format, jedoch ist das Protokoll nicht mehr vom Menschen lesbar.

3.1 Datentypen

Alle Datenfelder haben als **Länge** ein **Vielfaches von einem Byte**.

Als Datentypen werden ausschließlich Bitfelder und Ganzzahlen (Integer) verwendet. Integer können vorzeichenbehaftet (*signed*) oder nicht vorzeichenbehaftet (*unsigned*) sein, dies ist in der Beschreibung jedes Integer-Datenfeldes spezifiziert. Alle Integerdatenfelder mit mehr als einem Byte Länge werden in Network-Byte-Order (Big Endian) kodiert.

3.2 Bytestuffing

Die Übertragung erfolgt als fortlaufende Übermittlung von Datenpaketen mit konstanter Länge. Hierbei ergibt sich das Problem, den Beginn eines Datenpakets im Datenstrom zu erkennen. Auch wenn eine Anfangskennung zu Beginn des Datenpakets geschickt wird, so kann es dennoch passieren, dass genau diese Kennung auch im restlichen binären Datenstrom vorkommt. Somit kann ohne zusätzliche Maßnahmen der Start eines Datenpakets nicht sicher detektiert werden.

Wird das Protokoll auf einer TCP/IP-Schnittstelle eingesetzt, wird das erste Datenpaket gesendet, sobald der Socket verbunden ist. Das erste einzulesende Byte des Sockets entspricht daher in jedem Fall dem Beginn eines Datenpakets. Da alle Datenpakete die gleiche Länge besitzen und die Übertragung per TCP/IP fehlergesichert ist, kann man die Datenpakete einfach fortlaufend einlesen, indem immer dieselbe Anzahl Bytes (Länge des gesamten Datenpakets) auf der Empfängerseite eingelesen werden.

Bei der Verwendung des Protokolls auf einer RS-232 Schnittstelle gibt es dagegen keine garantierte fehlerfreie Übertragung. Der Empfänger einer Nachricht auf der seriellen Schnittstelle kann zu jedem beliebigen Zeitpunkt mit dem Einlesen der Schnittstelle beginnen. Er hat daher keine Möglichkeit, den Beginn eines Datenpaketes sicher zu erkennen.

Um das Protokoll auf einer RS232-Schnittstelle einsetzen zu können, kann das Binärformat so konfiguriert werden, das die Binärdaten mit einem **Bytestuffing** versehen werden. Bei aktiviertem Bytestuffing beginnt und endet jedes Datenpaket mit je einem reservierten Symbol, welches nicht im regulären Datenstrom vorkommen darf. Abb. 1 zeigt den prinzipiellen Aufbau eines binären Datenpakets.

Datenfeld	Inhalt
BINARY-START	0x7e
Inhalt des Datenpakets	...
BINARY-STOP	0x7f

Abb. 1: Anfangs- und Endekennung eines Binärdatenpakets

Sollten die reservierten Symbole innerhalb des Datenpakets vorkommen, so müssen sie nach den folgenden Regeln durch andere Symbole ersetzt werden:

Originalsymbol	wird im Protokoll ersetzt durch
0x7d	0x7d 0x5d
0x7e	0x7d 0x5e
0x7f	0x7d 0x5f

Abb. 2: Ersetzungsregeln beim Bytestuffing

Für jedes Auftreten eines der drei reservierten Symbole wird das Originalsymbol demnach durch zwei andere Symbole ersetzt. Im ungünstigsten Fall, wenn das gesamte Datenpaket aus reservierten Symbolen bestünde, würde sich die Datenmenge durch das Bytestuffing verdoppeln.

Dafür stellt das Bytestuffing sicher, dass der Empfänger des Protokolls das BINARY-START Feld eines Datenpakets in einem Datenstrom zweifelsfrei erkennen kann, selbst wenn das reservierte Symbol des BINARY-START Feldes im Inhalt des Datenpakets auftritt.

Das Dekodieren des Bytestuffings auf Empfängerseite kann wie folgt implementiert werden: Wenn Symbol 0x7d gelesen wurde, verwerfe dieses Symbol und XOR-verknüpfe das nächste Zeichen mit 0x20, um das Originalzeichen wiederherzustellen.

Das **Bytestuffing** ist **nur beim Binärformat** des 2D Protokolls verfügbar. Hierbei ist es standardmäßig aktiviert. Um das **Bytestuffing** zu **deaktivieren**, muss in der Konfigurationsdatei folgender Eintrag vorhanden sein:

```
[DisableBytestuffing] 1
```

3.3 Allgemeiner Aufbau

Der Aufbau eines Datenpakets des 2D Protokolls wird durch eine Konfigurationsdatei festgelegt. Einmal konfiguriert besitzt jedes Datenpaket dieselbe Länge und hat den gleichen Aufbau. Durch die Konfigurationsdatei werden unter anderem die

verwendeten Datenfelder festgelegt. Abb. 3 zeigt den Aufbau eines Datenpakets mit Standardkonfiguration.

Datenfeld
START
POSITION
END

Abb. 3: Allgemeiner Aufbau Datenpaket mit Standardkonfiguration

Bei Bedarf können auch zusätzliche Datenfelder aktiviert werden. Die Reihenfolge der aktivierten Datenfelder ist dabei fest vorgegeben und kann nicht verändert werden. Abb. 4 zeigt den Aufbau eines Datenpakets mit allen aktivierten Datenfeldern.

Datenfeld
START
TIMESTAMP
POSITION
VELOCITY
ORIENTATION
POSITION-ERROR
VELOCITY-ERROR
ORIENTATION-ERROR
USER-DATA
SYSTEM-ERROR
SATELLITE-STATE
CRC
END

Abb. 4: Allgemeiner Aufbau Datenpaket mit allen möglichen Datenfeldern

Die grau hinterlegten Datenfelder (**START** und **END**) sind in **jedem Datenpaket** vorhanden und können nicht deaktiviert werden. Alle anderen Datenfelder können in der Konfigurationsdatei entsprechend aus- bzw. ausgewählt werden.

3.4 Datenfelder

In diesem Kapitel werden die einzelnen Datenfelder beschrieben. Außer den beiden Datenfeldern START und END (in jedem Datenpaket enthalten) können alle anderen Datenfelder vom Anwender aktiviert bzw. deaktiviert werden.

Alle Längenangaben sind in Bytes angegeben.

3.4.1 START

Das Datenfeld START markiert den Beginn eines jeden Datenpakets. Es enthält zudem die Länge des gesamten Pakets und ein Bitfeld, in dem angegeben ist, welche Datenfelder aktiviert sind.

Name	Länge	Typ	Beschreibung
BINARY-START	1	unsigned int	Reserviertes Symbol 0x7e, welches den Beginn des Datenpakets eindeutig identifiziert.
LENGTH	2	unsigned int	Länge des gesamten Datenpakets in Byte (inkl. der Felder START und END).
SELECTED-FIELDS	4	bitmask	Bitmaske, die angibt, welche Datenfelder im Datenpaket aktiviert sind (siehe Kapitel 5).

Gesamtlänge des Datenfeldes: 7 Byte

3.4.2 END

Das Datenfeld END markiert das Ende eines jeden Datenpakets.

Name	Länge	Typ	Beschreibung
BINARY-STOP	1	unsigned int	Reserviertes Symbol 0x7f, welches das Ende des Datenpakets eindeutig identifiziert.

Gesamtlänge des Datenfeldes: 1 Byte

3.4.3 TIMESTAMP

Dieses Datenfeld gibt den Zeitpunkt an, zu dem eine Position gemessen wurde. Aufgrund der Rechenzeit für die Positionsbestimmung liegt dieser Zeitpunkt immer in der Vergangenheit.

Hinweis: Bei Betrieb der LPR-B 2D Software auf einer Hardwareplattform ohne batteriegepufferte RTC (*real time clock*), wie es beim SYMEO ARM9 Prozessor der Fall ist, ist die Uhrzeit dieses Systems nach Neustart immer auf den Zeitpunkt 1.1.1970 00:00:00 eingestellt.

Name	Länge	Typ	Beschreibung
TS-SEC	4	unsigned int	Anzahl Sekunden seit dem 01.01.1970 00:00:00
TS-MSEC	2	unsigned int	zusätzliche Anzahl Millisekunden (0..999)

Gesamtlänge des Datenfeldes: 6 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendTimestamp] 1
```

3.4.4 POSITION

Dieses Datenfeld gibt die 2D-Position (x,y) aus. Außerdem gibt dieses Datenfeld darüber Auskunft, ob diese Position zuverlässig ist.

Name	Länge	Typ	Beschreibung
POS-X	4	signed int	vorzeichenbehaftete x-Position in mm
POS-Y	4	signed int	vorzeichenbehaftete y-Position in mm
TRACKSTATE	1	unsigned int	Zuverlässigkeit der Position: 0, 1: Position ist nicht zuverlässig 2: Position ist zuverlässig

Gesamtlänge des Datenfeldes: 9 Byte

Dieses **Datenfeld** ist standardmäßig in jedem Datenpaket vorhanden, kann jedoch durch folgenden Eintrag in der Konfigurationsdatei **deaktiviert** werden:

[SendPosition] 0

Das Protokoll ist standardmäßig so konfiguriert, das nur zuverlässige Positionen ausgegeben werden (Trackzustand 2). Sollen auch unzuverlässige Positionen (z.B. während der Initialisierungsphase) ausgegeben werden, muss folgende Anweisung in der Konfigurationsdatei angegeben werden:

[OutputOnlyLockedTracks] 0

3.4.5 VELOCITY

Dieses Datenfeld gibt die Geschwindigkeit in x- und y-Richtung aus.

Hinweis: Ohne die Ausrichtung des Fahrzeugs zu kennen (Datenfeld ORIENTATION), kann aus der Geschwindigkeitsangabe nicht bestimmt werden, ob sich das zu ortende Objekt vorwärts oder rückwärts bewegt!

Name	Länge	Typ	Beschreibung
VEL-X	4	signed int	vorzeichenbehaftete Geschwindigkeit in x-Richtung in mm/s
VEL-Y	4	signed int	vorzeichenbehaftete Geschwindigkeit in y-Richtung in mm/s

Gesamtlänge des Datenfeldes: 8 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

[SendVelocity] 1

3.4.6 ORIENTATION

Dieses Datenfeld gibt die Ausrichtung (Winkellage) des zu ortenden Objekts an. Der Winkel wird gegen den Uhrzeigersinn gemessen, beginnend an der x-Achse.

Name	Länge	Typ	Beschreibung
ANGLE	2	unsigned int	Winkellage des Objekts in Grad (0..359°)

Gesamtlänge des Datenfeldes: 2 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendOrientation] 1
```

3.4.7 POSITION-ERROR

Dieses Datenfeld gibt den geschätzten Positionsfehler (EPE, *estimated position error*) an. Der EPE ist immer ein positiver Wert.

Name	Länge	Typ	Beschreibung
POS-ERR-X	4	unsigned int	geschätzter Fehler der x-Position in mm
POS-ERR-Y	4	unsigned int	geschätzter Fehler der y-Position in mm

Gesamtlänge des Datenfeldes: 8 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendPosError] 1
```

3.4.8 VELOCITY-ERROR

Dieses Datenfeld gibt den geschätzten Geschwindigkeitsfehler an. Dieser Wert ist immer positiv.

Name	Länge	Typ	Beschreibung
VEL-ERR-VX	4	unsigned int	geschätzter Fehler der Geschwindigkeit in x-Richtung in mm/s
VEL-ERR-VY	4	unsigned int	geschätzter Fehler der Geschwindigkeit in y-Richtung in mm/s

Gesamtlänge des Datenfeldes: 8 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendVelError] 1
```

3.4.9 ORIENTATION-ERROR

Dieses Datenfeld gibt den geschätzten Fehler der Ausrichtung an. Dieser Wert ist immer positiv.

Name	Länge	Typ	Beschreibung
ANGLE-ERR	2	unsigned int	geschätzter Fehler der Ausrichtung in Grad

Gesamtlänge des Datenfeldes: 2 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendOrientationError] 1
```

3.4.10 USER-DATA

Dieses Datenfeld kann verwendet werden, um einen Anwenderdatensatz auszugeben. Die Bedeutung des Anwenderdatensatzes ist applikationsabhängig.

Name	Länge	Typ	Beschreibung
USER-DATA-SET	8	unsigned int	Anwenderdaten

Gesamtlänge des Datenfeldes: 8 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendUserData] 1
```

3.4.11 SYSTEM-ERROR

Dieses Datenfeld gibt Auskunft über möglicherweise aufgetretene Fehler des Systems. Es können gleichzeitig bis zu fünf verschiedene Fehler pro Datenpaket ausgegeben werden. Ein Fehlercode wird solange gesendet, solange der Fehler besteht.

Name	Länge	Typ	Beschreibung
ERROR-CODE-1	1	unsigned int	Fehlercode des 1. Fehlers
ERROR-VALUE-1	2	unsigned int	Fehlerwert des 1. Fehlers
ERROR-CODE-2	1	unsigned int	Fehlercode des 2. Fehlers
ERROR-VALUE-2	2	unsigned int	Fehlerwert des 2. Fehlers
ERROR-CODE-3	1	unsigned int	Fehlercode des 3. Fehlers
ERROR-VALUE-3	2	unsigned int	Fehlerwert des 3. Fehlers
ERROR-CODE-4	1	unsigned int	Fehlercode des 4. Fehlers
ERROR-VALUE-4	2	unsigned int	Fehlerwert des 4. Fehlers
ERROR-CODE-5	1	unsigned int	Fehlercode des 5. Fehlers
ERROR-VALUE-5	2	unsigned int	Fehlerwert des 5. Fehlers

Gesamtlänge des Datenfeldes: 15 Byte

Eine Beschreibung der einzelnen Fehler findet sich in Kapitel 7. Sind mehr als fünf Fehler vorhanden, wird als ERROR-CODE-5 der spezielle Fehlercode 0xff gesendet.

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendSystemError] 1
```

3.4.12 SATELLITE-STATE

Dieses Datenfeld wird bei satellitengestützter Lokalisierung benutzt und enthält Informationen über die Qualität der Positionsermittlung.

Name	Länge	Typ	Beschreibung
SAT-COUNT	1	signed int	Anzahl der verfolgten Satelliten Der einzig erlaubte <i>negative</i> Wert ist „-1“, er steht für „unbekannt“, z.B. bei Ausfall der Hardware.
SAT-HDOP	2	signed int	10 * horizontal dilution of precision Der Integer-Wert 123 steht also für den HDOP-Wert 12,3. Ein HDOP-Wert von „-1“ (also „-10“ in diesem Format) steht für „unbekannt“, z.B. bei einem Hardwaredefekt.

Gesamtlänge des Datenfeldes: 3 Byte

3.5 CRC

Dieses Datenfeld gibt den CRC (*cyclic redundancy check*) des Datenpakets an.

Name	Länge	Typ	Beschreibung
CRC-16	2	unsigned int	CRC-Wert der Nachricht

Gesamtlänge des Datenfeldes: 2 Byte

Eine genaue Beschreibung des CRCs und ein passender Quelltext in C befinden sich in Kapitel 6.

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendCRC] 1
```

4 ASCII-Format des Protokolls

Beim ASCII-Format des 2D Protokolls wird das gesamte Datenpaket im ASCII-Code, nur unter Verwendung von Buchstaben, Zahlen und wenigen Sonderzeichen übertragen. Damit ist das ASCII-Format auch vom Menschen lesbar. Im Gegensatz zum Binärformat werden aufgrund des geringeren Zeichenvorrates die Daten weniger kompakt übertragen – die Datenmenge steigt an.

4.1 Datentypen

In jedem Datenpaket werden nur Zeichen(ketten), Zahlen (optional mit Dezimalpunkt und Vorzeichen) und Unterstriche verwendet. Jedes Datenpaket besteht aus einer Zeile, abgeschlossen durch das Sonderzeichen LF (*line feed* \n, ASCII-Code 0x0a)

Alle Zahlen haben vorgegebene, feste Anzahl an Ziffern. Werden weniger Ziffern für die Darstellung eines Wertes benötigt, müssen entsprechend viele führende Nullen vorangestellt bzw. angehängt werden.

Es sind sowohl Ganzzahlen (integer) als auch Gleitkommazahlen vorhanden. In der Beschreibung der einzelnen Datenfelder werden Zahlen folgendermaßen beschrieben:

- + Vorzeichen, immer (+ oder -)
- Vorzeichen, nur falls Wert negativ (dann -)
- # einzelne Dezimalstelle/Ziffer
- . Dezimalpunkt (nur bei Gleitkommazahlen)

Beispiel:

zu kodierende Gleitkommazahl: 12.34

Beschreibung: +###.####

→ kodierte Zahl: +012.3400

4.2 Allgemeiner Aufbau

Der Aufbau eines Datenpakets des 2D Protokolls wird durch eine Konfigurationsdatei festgelegt. Jedes Datenpaket besteht aus einer Textzeile - das Ende eines jeden Pakets ist eindeutig durch das ASCII-STOP Zeichen \n definiert.

Einmal konfiguriert besitzt jedes Datenpaket dieselbe Länge und hat den gleichen Aufbau. Durch die Konfigurationsdatei werden unter anderem die verwendeten

Datenfelder festgelegt. Abb. 5Abb. 5Abb. 5Abb. 5Abb. 5Abb. 5 zeigt den Aufbau eines Datenpakets mit Standardkonfiguration.

Datenfeld
START
POSITION
END

Abb. 5: Allgemeiner Aufbau Datenpaket mit Standardkonfiguration

Bei Bedarf können auch zusätzliche Datenfelder aktiviert werden. **Die Reihenfolge der aktivierten Datenfelder ist dabei fest vorgegeben und kann nicht verändert werden.** Abb. 6 zeigt den Aufbau eines Datenpakets mit allen aktivierten Datenfeldern.

Datenfeld
START
TIMESTAMP
POSITION
VELOCITY
ORIENTATION
POSITION-ERROR
VELOCITY-ERROR
ORIENTATION-ERROR
USER-DATA
SYSTEM-ERROR
SATELLITE-STATE
CRC
END

Abb. 6: Allgemeiner Aufbau Datenpaket mit allen möglichen Datenfeldern

Die grau hinterlegten Datenfelder (START und END) sind in jedem Datenpaket vorhanden und können nicht deaktiviert werden. Alle anderen Datenfelder können in der Konfigurationsdatei entsprechend aus- bzw. ausgewählt werden.

4.3 Datenfelder

In diesem Kapitel werden die einzelnen Datenfelder beschrieben. Außer den beiden Datenfeldern START und END (in jedem Datenpaket enthalten) können alle anderen Datenfelder vom Anwender aktiviert bzw. deaktiviert werden.

Alle Längenangaben sind in Bytes angegeben.

4.3.1 START

Das Datenfeld START markiert den Beginn eines jeden Datenpakets. Es enthält zudem die Länge des gesamten Pakets und ein Bitfeld, in dem angegeben ist, welche Datenfelder aktiviert sind.

Name	Länge	Beschreibung
ASCII-START	1	ASCII Zeichen A (0x41)
LENGTH	3	Länge des gesamten Datenpakets in Byte (inkl. der Felder START und END). Zahlenkodierung: ###
SELECTED-FIELDS	8	Bitmaske, die angibt, welche Datenfelder im Datenpaket aktiviert sind (siehe Kapitel 5). Zahlenkodierung: ##### Bitfeld wird hexadezimal kodiert!
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 13 Byte

4.3.2 END

Das Datenfeld END markiert das Ende eines jeden Datenpakets. Es besteht nur aus dem einzelnen reservierten Symbol `\n`, welches das Ende des Datenpakets eindeutig identifiziert.

Name	Länge	Beschreibung
ASCII-STOP	1	ASCII Zeichen <code>\n</code> (0x0a)

Gesamtlänge des Datenfeldes: 1 Byte

4.3.3 TIMESTAMP

Dieses Datenfeld gibt den Zeitpunkt an, zu dem eine Position gemessen wurde. Aufgrund der Rechenzeit für die Positionsbestimmung liegt dieser Zeitpunkt immer in der Vergangenheit.

Hinweis: Bei Betrieb der LPR-B 2D Software auf einer Hardwareplattform ohne batteriegepufferte RTC (*real time clock*), wie es beim SYMEO ARM9 Prozessor der Fall ist, ist die Uhrzeit dieses Systems nach Neustart immer auf den Zeitpunkt 1.1.1970 00:00:00 eingestellt.

Name	Länge	Beschreibung
TIME	4	ASCII Zeichenkette t ime
TS-SEC	10	Anzahl Sekunden seit dem 01.01.1970 00:00:00 Zahlenkodierung: #####
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
TS-MSEC	3	zusätzliche Anzahl Millisekunden (0..999) Zahlenkodierung: ###
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 19 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendTimestamp] 1
```

4.3.4 POSITION

Dieses Datenfeld gibt die 2D-Position (x,y) aus. Außerdem gibt dieses Datenfeld darüber Auskunft, ob diese Position zuverlässig ist.

Name	Länge	Beschreibung
X	1	ASCII Zeichen x
POS-X	10	vorzeichenbehaftete x-Position in Metern

		Zahlenkodierung: +#####.###
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
Y	1	ASCII Zeichen y
POS-Y	10	vorzeichenbehaftete y-Position in Metern Zahlenkodierung: +#####.###
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
TRACKSTATE	1	Zahl, die die Zuverlässigkeit der Position angibt: 0,1: Position nicht zuverlässig 2: Position ist zuverlässig Zahlenkodierung: #
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 26 Byte

Dieses Datenfeld ist standardmäßig in jedem Datenpaket vorhanden, kann jedoch durch folgenden Eintrag in der Konfigurationsdatei deaktiviert werden:

```
[SendPosition] 0
```

Das Protokoll ist standardmäßig so konfiguriert, dass nur zuverlässige Positionen ausgegeben werden (Trackzustand 2). Sollen auch unzuverlässige Positionen (z.B. während der Initialisierungsphase) ausgegeben werden, muss folgende Anweisung in der Konfigurationsdatei angegeben werden:

```
[OutputOnlyLockedTracks] 0
```

4.3.5 VELOCITY

Dieses Datenfeld gibt die Geschwindigkeit in x- und y-Richtung aus.

Hinweis: Ohne die Ausrichtung des Fahrzeugs zu kennen (Datenfeld ORIENTATION), kann aus der Geschwindigkeitsangabe nicht bestimmt werden, ob sich das zu ortende Objekt vorwärts oder rückwärts bewegt!

Name	Länge	Beschreibung
VX	2	ASCII Zeichenkette vx

VEL-X	6	vorzeichenbehaftete Geschwindigkeit in x-Richtung in m/s Zahlenkodierung: +##.##
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
VY	2	ASCII Zeichenkette vy
VEL-Y	6	vorzeichenbehaftete Geschwindigkeit in y-Richtung in m/s Zahlenkodierung: +##.##
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 18 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendVelocity] 1
```

4.3.6 ORIENTATION

Dieses Datenfeld gibt die Ausrichtung (Winkellage) des zu ortenden Objekts an. Der Winkel wird gegen den Uhrzeigersinn gemessen, beginnend an der x-Achse.

Name	Länge	Beschreibung
O	1	ASCII Zeichen o
ANGLE	3	Winkellage des Objekts in Grad (0..359°) Zahlenkodierung: ###
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 5 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendOrientation] 1
```

4.3.7 POSITION-ERROR

Dieses Datenfeld gibt den geschätzten Positionsfehler (EPE, *estimated position error*) an. Der EPE ist immer ein positiver Wert.

Name	Länge	Beschreibung
EX	2	ASCII Zeichenkette ex
POS-ERR-X	5	geschätzter Fehler der x-Position in m Zahlenkodierung: ##.##
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
EY	2	ASCII Zeichenkette ey
POS-ERR-Y	5	geschätzter Fehler der y-Position in m Zahlenkodierung: ##.##
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 16 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendPosError] 1
```

4.3.8 VELOCITY-ERROR

Dieses Datenfeld gibt den geschätzten Geschwindigkeitsfehler an. Dieser Wert ist immer positiv.

Name	Länge	Beschreibung
EVX	3	ASCII Zeichenkette evx
POS-ERR-X	5	geschätzter Fehler der Geschwindigkeit in x-Richtung in m/s Zahlenkodierung: ##.##
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
EVY	3	ASCII Zeichenkette evy

POS-ERR-Y	5	geschätzter Fehler der Geschwindigkeit in y-Richtung in mm/s Zahlenkodierung: ##.##
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 18 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendVelError] 1
```

4.3.9 ORIENTATION-ERROR

Dieses Datenfeld gibt den geschätzten Fehler der Ausrichtung an. Dieser Wert ist immer positiv.

Name	Länge	Beschreibung
EO	2	ASCII Zeichenkette eo
ANGLE-ERR	3	geschätzter Fehler der Ausrichtung in Grad Zahlenkodierung: ###
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 6 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendOrientationError] 1
```

4.3.10 USER-DATA

Dieses Datenfeld kann verwendet werden, um einen Anwenderdatensatz auszugeben. Die Bedeutung des Anwenderdatensatzes ist applikationsabhängig.

Name	Länge	Beschreibung
USER	4	ASCII Zeichenkette user
USER-DATA-SET	16	Anwenderdaten Zahlenkodierung: ##### Anwenderdaten werden hexadezimal kodiert!
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 21 Byte

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendUserData] 1
```

4.3.11 SYSTEM-ERROR

Dieses Datenfeld gibt Auskunft über möglicherweise aufgetretene Fehler des Systems. Es können gleichzeitig bis zu fünf verschiedene Fehler pro Datenpaket ausgegeben werden. Ein Fehlercode wird solange gesendet, solange der Fehler besteht.

Name	Länge	Beschreibung
ERR	3	ASCII Zeichenkette err
ERROR-CODE-1	2	Fehlercode des 1. Fehlers Zahlenkodierung: ## Wert wird hexadezimal kodiert!
ERROR-VALUE-1	4	Fehlerwert des 1. Fehlers Zahlenkodierung: #### Wert wird hexadezimal kodiert!
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
ERROR-CODE-2	2	Fehlercode des 2. Fehlers
ERROR-VALUE-2	4	Fehlerwert des 2. Fehlers
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

ERROR-CODE-3	2	Fehlercode des 3. Fehlers
ERROR-VALUE-3	4	Fehlerwert des 3. Fehlers
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
ERROR-CODE-4	2	Fehlercode des 4. Fehlers
ERROR-VALUE-4	4	Fehlerwert des 4. Fehlers
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
ERROR-CODE-5	2	Fehlercode des 5. Fehlers
ERROR-VALUE-5	4	Fehlerwert des 5. Fehlers
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 38 Byte

Eine Beschreibung der einzelnen Fehler findet sich in Kapitel 7. Sind mehr als fünf Fehler vorhanden, wird als ERROR-CODE-5 der spezielle Fehlercode 0xff gesendet.

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendSystemError] 1
```

4.3.12 SATELLITE-STATE

Dieses Datenfeld wird bei satellitengestützter Lokalisierung benutzt und enthält Informationen über die Qualität der Positionsermittlung.

Name	Länge	Beschreibung
SAT	3	ASCII Zeichenkette sat
SAT-COUNT	2	Anzahl der verfolgten Satelliten Zahlenkodierung: ## (oder „-1“ für „unbekannt“)
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)
SAT-HDOP	4	horizontal dilution of precision Zahlenkodierung: ##.# (oder „-1.0“ für „unbekannt“)
UNDERLINE	1	Unterstrich (ASCII Code 0x5F)

Gesamtlänge des Datenfeldes: 11 Byte

4.3.13 CRC

Dieses Datenfeld gibt den CRC (*cyclic redundancy check*) des Datenpakets an.

Name	Länge	Beschreibung
CRC	3	ASCII Zeichenkette crc
CRC-16	4	CRC-Wert der Nachricht Zahlenkodierung: #### Wert wird hexadezimal kodiert!

Gesamtlänge des Datenfeldes: 7 Byte

Eine genaue Beschreibung des CRCs und ein passender Quelltext in C befinden sich in Kapitel 6.

Dieses Datenfeld kann mit folgender Anweisung in der Konfigurationsdatei aktiviert werden:

```
[SendCRC] 1
```

5 Bitmaske SELECTED-FIELDS

Die Bitmaske SELECTED-FIELDS ist Bestandteil des Datenfelds START. Sie gibt Auskunft, welche Bitfelder im Datenpaket aktiviert sind. Einmal konfiguriert besitzt jedes Datenpaket dieselbe Länge und hat den gleichen Aufbau. Daher ist auch der Wert der Bitmaske SELECTED-FIELDS konstant. Durch Auswerten der Bitmaske auf Empfängerseite des Protokolls kann effektiv kontrolliert werden, ob die Konfiguration des Protokolls korrekt ist.

Jedes Datenfeld wird in der Bitmaske als einzelnes Bit repräsentiert. Ist das Datenfeld aktiviert, ist das dazugehörige Bit gesetzt.

Abb. 7 zeigt die Zuordnung der einzelnen Bits zu den Datenfeldern. Bit 31 ist dabei das höchstwertige Bit (MSB, *most significant bit*), befindet sich in hexadezimaler Darstellung (ASCII-Format des Protokolls) ganz links.

Wichtig: Die Reihenfolge der Bits in dieser Bitmaske entspricht nicht notwendigerweise der Reihenfolge der aktivierten Datenfelder im Datenpaket. Die

Reihenfolge der Datenfelder ist in Kapitel 3.3 (Binärformat) bzw. Kapitel 4.2 (ASCII-Format) festgelegt.

Bit	Datenfeld	Bit	Datenfeld
31	-	15	-
30	-	14	-
29	-	13	-
28	-	12	-
27	-	11	-
26	-	10	SATELLITE-STATE
25	-	9	CRC
24	-	8	SYSTEM-ERROR
23	-	7	USER-DATA
22	-	6	ORIENTATION-ERROR
21	-	5	VELOCITY-ERROR
20	-	4	POSITION-ERROR
19	-	3	ORIENTATION
18	-	2	VELOCITY
17	-	1	POSITION
16	-	0	TIMESTAMP

Abb. 7: Zuordnung Bitmaske – Datenfelder

6 CRC Berechnung

Zur Erkennung von Fehlern während der Übertragung kann das Datenfeld CRC aktiviert werden. Als CRC wird der CRC-16-IBM verwendet mit dem Polynom $x^{16}+x^{15}+x^2+1$. Der CRC wird über alle vorherigen Datenfelder des Datenpaketes, jedoch nicht über das Datenfeld START gerechnet.

Beispiel-Quellcode für CRC-Berechnung in C:

```
// Holds a table to calculate crc16 values
static Uint16 crc_table[256];

// Initializes the CRC table
// MUST BE RUN before first crc calculation
void InitCRCTable( void )
{
    int i, j;
```

```
    Uint16 k;

    for (i = 0; i < 256; i++)
        crc_table[i] = i;

    for (i = 0; i < 256; i++)
    {
        k = 0xC0C0;
        for (j = 1; j < 256; j <= 1)

            {

                if (i & j)

                    crc_table[i] ^= k;

                k = ((k & 0x7FFF) << 1) ^ 0x4003;

            }

    }

}

// Adds calculation for an 8 bit value to crc.
// Initially crc should be zero.
Uint16 CalcCRC8(Uint16 crc, Uint8 value)
{
    crc = (crc >> 8) ^ crc_table[(crc & 0xFF) ^ value];
    return crc;
}
```

7 Fehlercodes

Ein Systemfehler wird gesendet, sobald das System selbstständig einen Fehler erkannt hat (Selbstdiagnose). Der Fehlercode wird solange gesendet, bis der Fehler behoben ist. Wird zum Beispiel ein Transponder als defekt erkannt, wird der entsprechende Fehlercode solange weitergesendet, bis wieder eine korrekte Messung mit diesem Transponder durchgeführt werden kann.

Ein Spezialfall ist der Fehlercode 0xff. Er wird gesendet, wenn mehr als fünf Fehler gleichzeitig aktiv sind und daher nicht alle Fehlercodes übertragen werden können. Dieser spezielle Fehlercode wird daher auch nur im letzten Fehlercode des Systemfehlers (ERROR-CODE-5) gesendet.

Ein Fehler besteht immer aus einem Fehlercode kann optional einen Fehlerwert besitzen, der zusätzliche Informationen zum entsprechenden Fehlercode enthält.

Übersicht der Fehlercodes:

Code	Bedeutung
0x01	Keine Verbindung zum LPR-B DSP
0x02	Transponder defekt
0xff	Weitere Fehler aktiv

Ausführliche Beschreibung der Fehlercodes und Fehlerwerte:

Code	Beschreibung
0x01	Es konnten in den letzten 3 Sekunden keine Messungen vom LPR-B DSP empfangen werden. Möglicherweis Ursachen: - physikalische Verbindung zum DSP ist unterbrochen - DSP fehlerhaft konfiguriert - DSP defekt
0x02	In einem LPR-B 2D Messsystem wurde über längere Zeit ein bestimmter Transponder nicht erkannt. Der Fehlerwert gibt die LPR-Adresse des vermutlich defekten Transponders an. Ein Transponder wird erst dann als defekt erkannt, wenn er an mehreren Positionen nicht empfangen wurde. Möglicherweise ist der TP auch verdeckt, so dass er daher nicht empfangen werden konnte oder er befindet sich außerhalb der Reichweite.
0xff	Es sind momentan mehr als 5 Fehler aktiv. Der Fehlerwert gibt die Gesamtzahl der aktiven Fehler an (> 5).