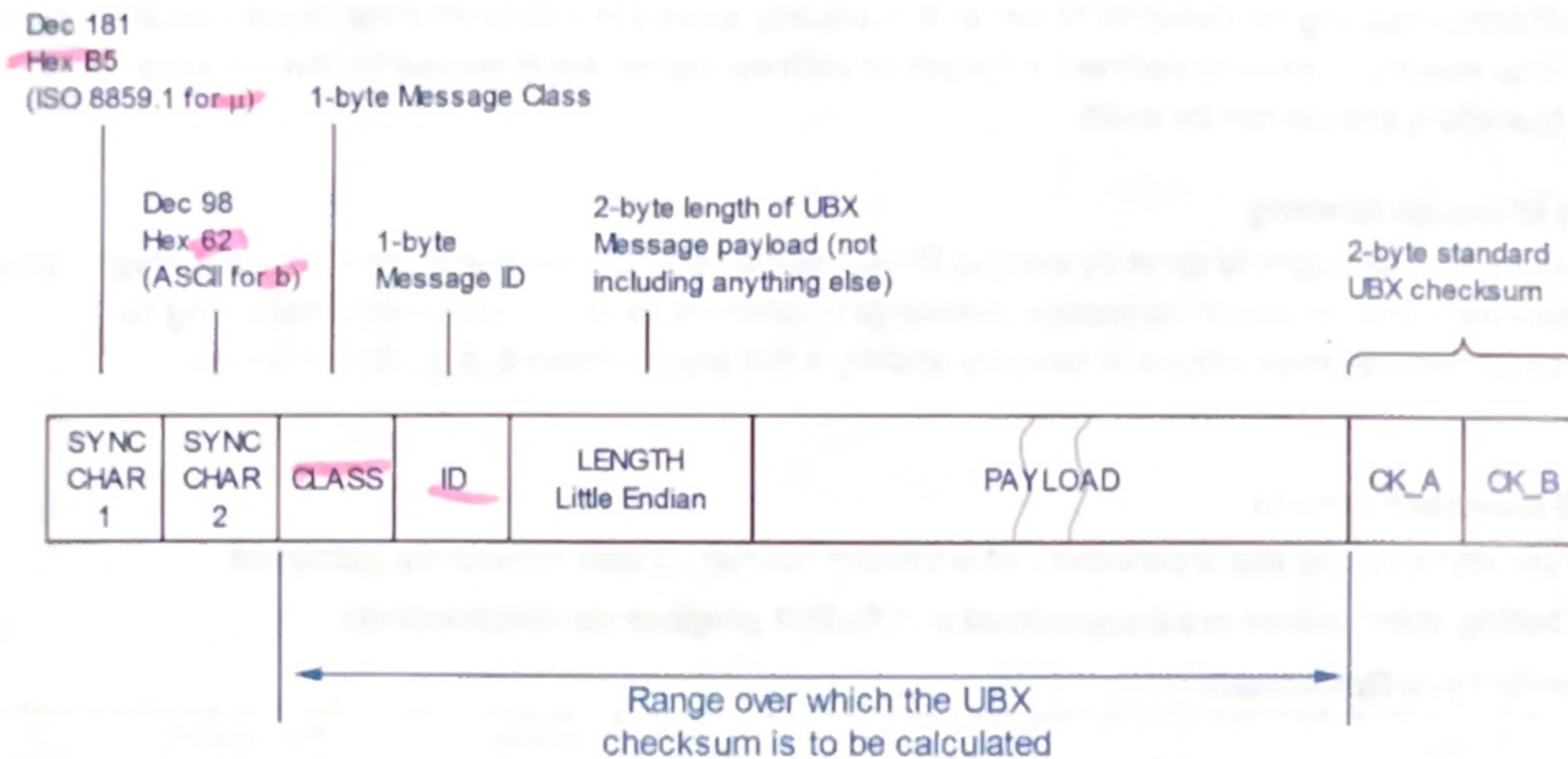


also see page 41

## 5.2 UBX Frame Structure

The structure of a basic UBX Frame is shown in the following diagram.



- Every **Frame** starts with a 2-byte Preamble consisting of two synchronization characters: 0xB5 0x62.
- A 1-byte Message **Class** field follows. A Class is a group of messages that are related to each other.
- A 1-byte Message **ID** field defines the message that is to follow.
- A 2-byte **Length** field follows. The length is defined as being that of the payload only. It does not include the Preamble, Message Class, Message ID, Length, or CRC fields. The number format of the length field is a Little-Endian unsigned 16-bit integer.
- The **Payload** field contains a variable number of bytes.
- The two 1-byte **CK\_A** and **CK\_B** fields hold a 16-bit checksum whose calculation is defined below. This concludes the Frame.

The checksum algorithm used is the 8-Bit Fletcher Algorithm, which is used in the TCP standard (RFC 1145). This algorithm works as follows:

- Buffer[N] contains the data over which the checksum is to be calculated.
- The two CK\_values are 8-Bit unsigned integers, only If implementing with larger-sized integer values, make sure to mask both CK\_A and CK\_B with 0xFF after both operations in the loop.

```
CK_A = 0, CK_B = 0
For (I=0; I<N; I++)
{
    CK_A = CK_A + Buffer[I]
    CK_B = CK_B + CK_A
}
```

- After the loop, the two U1 values contain the checksum, transmitted after the Message, which concludes the Frame.

### 5.5.2 Polling Mechanism

All messages that are output by the receiver in a periodic manner (i.e. messages in classes MON, NAV and RXM) and Get/Set type messages, such as the messages in the CFG class, can also be polled.

The UBX protocol is designed so that messages can be polled by sending the message required to the receiver but without a payload (or with just a single parameter that identifies the poll request). The receiver then responds with the same message with the payload populated.

The UBX protocol can be read out over I<sub>2</sub>C. We will continually read data out of the 0xFF register. To determine how many bytes are available, we can read the 0x FD and 0x FE register. See page 87 for more details.

There are lots of messages we can poll over UBX. See the datasheet for the full list.

We care about NAV-PVT because it gives position Velocity and time information.

Page	Mnemonic	Cls/ID	Length	Type	Description
154	NAV-POSLH	0x01 0x02	28	Periodic/Polled	Geodetic Position Solution
155	NAV-PVT	0x01 0x07	92	Periodic/Polled	Navigation Position Velocity Time...
158	NAV-RELPOSNE	0x01 0x3C	64	Periodic/Polled	Relative Positioning Information in...
160	NAV-RESETODO	0x01 0x10	0	Command	Reset odometer
161	NAV-SAT	0x01 0x35	8 + 12*numSvs	Periodic/Polled	Satellite Information
163	NAV-SIG	0x01 0x43	8 + 16*numSi...	Periodic/Polled	Signal Information
165	NAV-STATUS	0x01 0x03	16	Periodic/Polled	Receiver Navigation Status
167	NAV-SVIN	0x01 0x3B	40	Periodic/Polled	Survey-in data
168	NAV-TIMEBDS	0x01 0x24	20	Periodic/Polled	BDS Time Solution
169	NAV-TIMEGAL	0x01 0x25	20	Periodic/Polled	Galileo Time Solution
170	NAV-TIMEGLO	0x01 0x23	20	Periodic/Polled	GLO Time Solution
171	NAV-TIMEGPS	0x01 0x20	16	Periodic/Polled	GPS Time Solution
172	NAV-TIMELS	0x01 0x26	24	Periodic/Polled	Leap second event information
174	NAV-TIMEUTC	0x01 0x21	20	Periodic/Polled	UTC Time Solution
176	NAV-VELECEF	0x01 0x11	20	Periodic/Polled	Velocity Solution in ECEF
176	NAV-VELNED	0x01 0x12	36	Periodic/Polled	Velocity Solution in NED