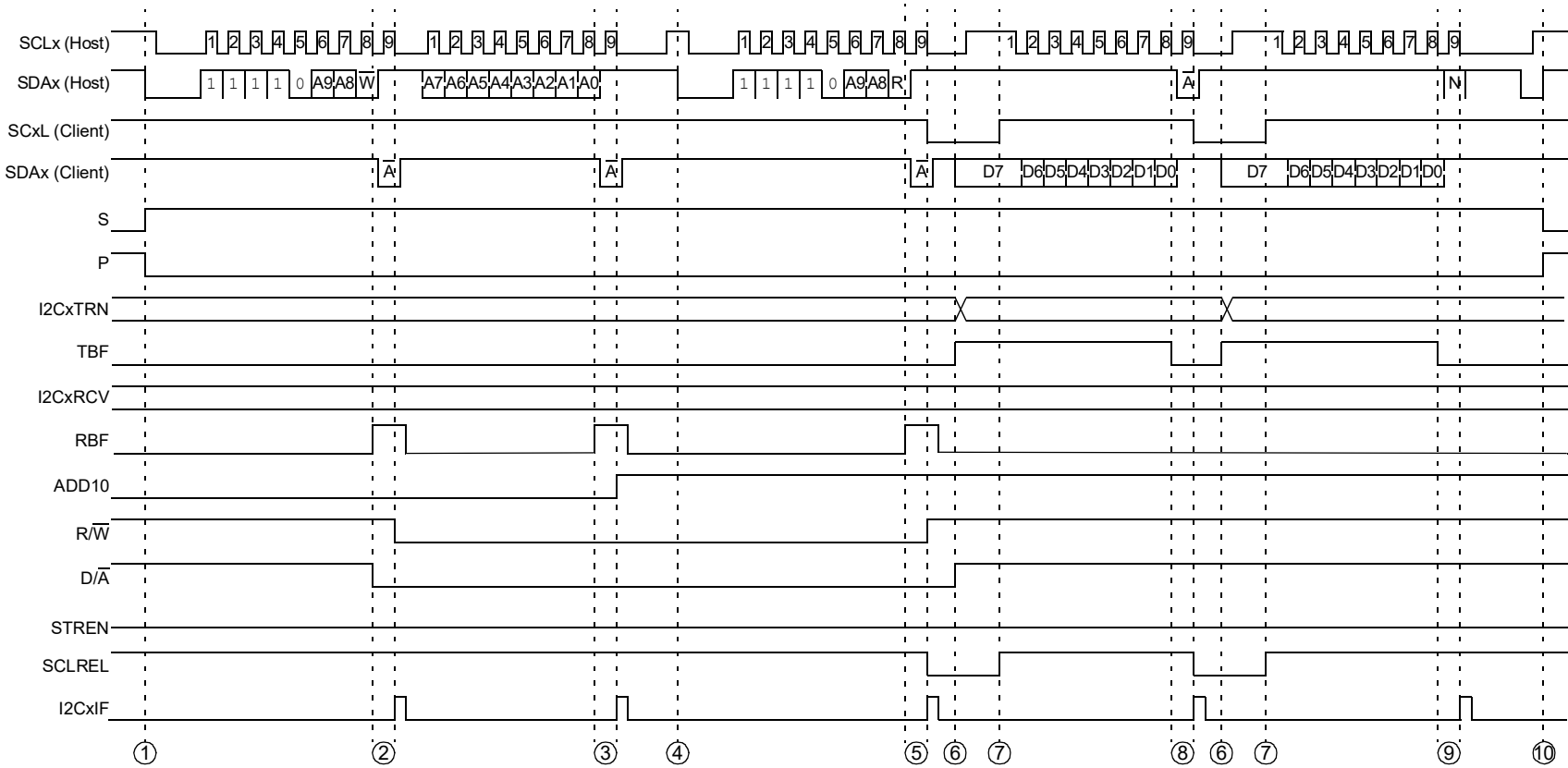

① Client recognizes Start event; S and P bits set/clear accordingly.

② Client receives first address byte. Write indicated. Client Acknowledges and generates interrupt. User software reads I2CxRCV register.

③ Client receives address byte. Address matches. Client Acknowledges and generates interrupt. User software reads I2CxRCV register.

④ Host sends a Repeated Start to redirect the message.

⑤ Client receives resend of first address byte. User software reads I2CxRCV register. Read indicated. Client suspends clock.

⑥ User software writes I2CxTRN with response data.

⑦ User software sets SCLREL to release clock hold. Host resumes clocking and client transmits data byte.

⑧ At the end of ninth clock, if host sent an $\overline{\text{ACK}}$, module clears SCLREL to suspend clock. Client generates interrupt.

⑨ At the end of ninth clock, if host sent a NACK, no more data expected. User software should stop writing to I2CxTRN. Module does not suspend clock and will generate an interrupt.

⑩ Client recognizes Stop event; S and P bits set/clear accordingly.