Appendix A

Appendix A.1: Photos of the completed system

Appendix B: Data CD

Appendix B.1: System requirements specification

Systems requirements specification for the Active Magnetic Bearing and Drive Electronic System.

Appendix B.2: Communication drivers data sheets

ISensorboard driver chip datasheet, main controller driver chip data sheet and power amplifier driver chip data sheet.

Appendix B.3: CRC article

CRC tables

Appendix B.4: VHDL code

VHDL code

Appendix B.5: MATLAB® code

MATLAB® code

Appendix B.6: Example of Modelsim® test benches

Modelsim® simulation code

Appendix B.7: Hardware guides

Hardware guides

Appendix C

Appendix C.1: State machine of the UART receiver

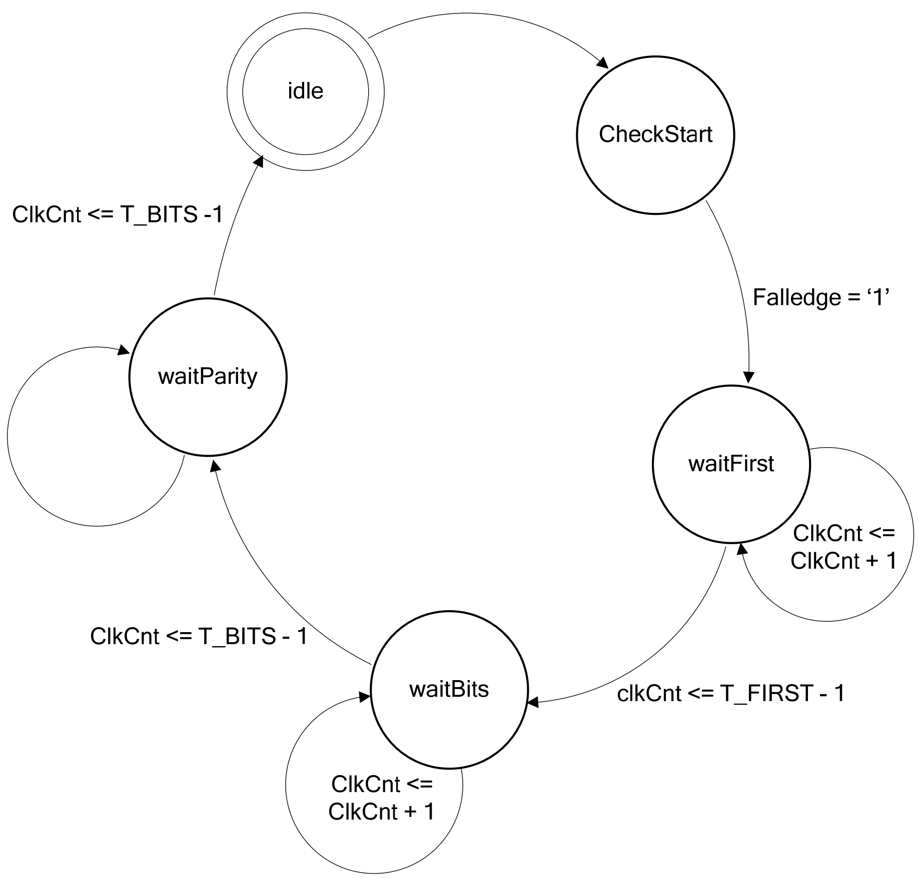


Figure C-1-1: UART receiver state machine

<table>
<tr><td>Idle</td><td>
During the idle state the UART receiver waits for the fallEdge signal to go high. This signal continuously monitors the serialIn line for a falling edge. Once a falling edge is detected as shown in Figure C-1-2 it is assumed that a start bit has been detected.

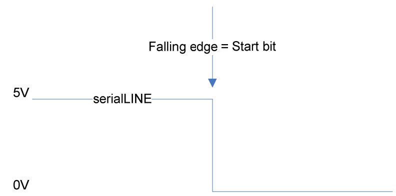


Figure C-1-2: Start bit detection

Noise immunity is an immense problem, due to the power amplifiers, therefore the probability exists that false falling edges can be detected. Thus an extra state is included to assist avoiding this critical problem. The added state is invoke just after the idle state and is called the checkStart state. What this added state does is instead of checking only for a falling edge and assuming that it is a start bit the extra state count to the mid point of the start bit and checks whether the serialIn signal is still low. Ensuring that a noise spike has not triggered the receiver, but a start bit has. After this has occurred fallEdge is set high and state transition will occur to start receiving the first data bit.
</td></tr>
<tr><td>WaitFirst</td><td>
During the WaitFirst state an internal counter, counts until it reaches the mid point of the first data bit available on the serialIn input signal, as shown in Figure C-1-3. The counter will only count from 0 to 7 to reach the mid point of the first data bit. The incoming data bit is than shifted into a register and the bit counter is incremented by one, indicating that the first data bit has been received. In this state the parity calculation starts. This is done by performing an XOR calculation between the parity signal which is set to 0 and the first incoming data bit.

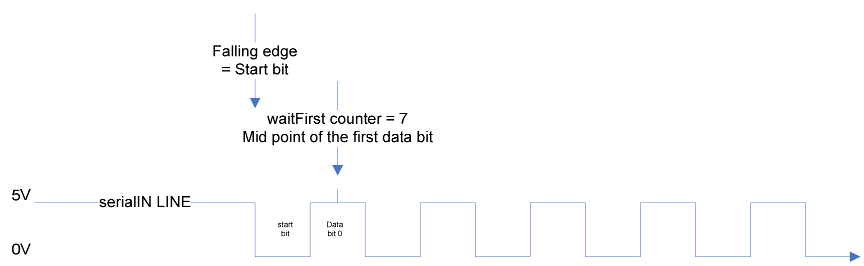
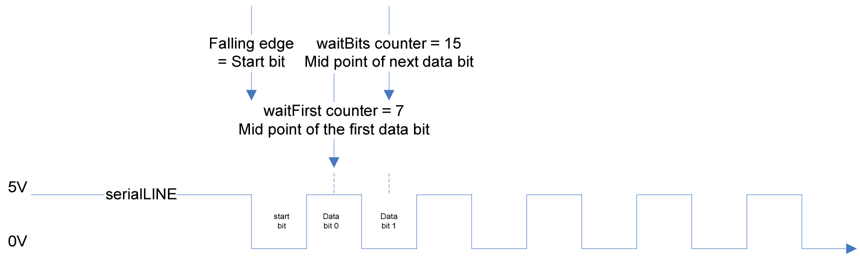
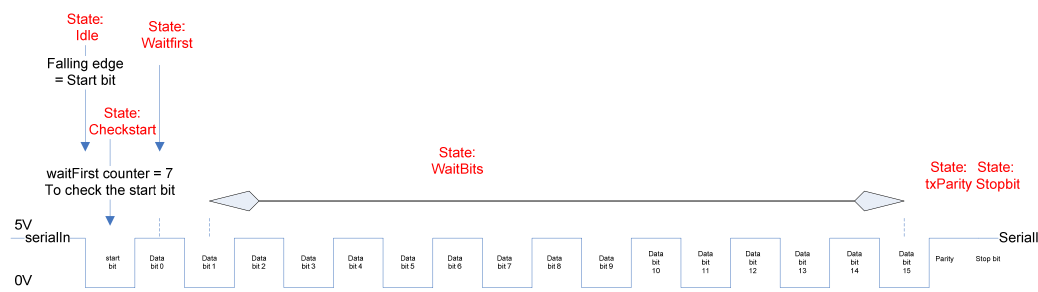


Figure C-1-3: WaitFirst counter
</td></tr>
<tr><td>WaitBits</td><td>
In the WaitBits state an internal counter counts until it reaches the mid point of the next incoming data bit, however this time it will count from 0 to 15 to reach the mid point of the next data bit as shown in Figure C-1-4.The data bit is shifted out and the parity signal is adjusted by performing another XOR operation between the new incoming data bit and the parity signal. The bit counter is incremented and the process is repeated until all the data bits are received. Now state transition can occur.
</td></tr>
</table>

	 **Figure C-1-4: WaitBits counter**
WaitParity	Once again the internal counter counts until it reaches the mid point of the last data bit which is the parity. If the data bit and the parity signal does not match the **Err** signal is flagged – which indicates an internal UART error. The **dataRxRdy** signal is set to '1' indicating that data is available in the **dataRxed** register to be used by other communication components. The state transition can now occur to the default state –Idle. The graphical representation of the procedure is shown in Figure C-1-5.  **Figure C-1-5: State addition**
Others state	An others state is incorporated in every component written for the internal communication. The reason being that in the unlikely event that the system goes into a possible undefined state the system always refers to the default idle state. This is considered to be crucial in safety critical software systems.

Appendix C.2: State machine of the UART transmitter

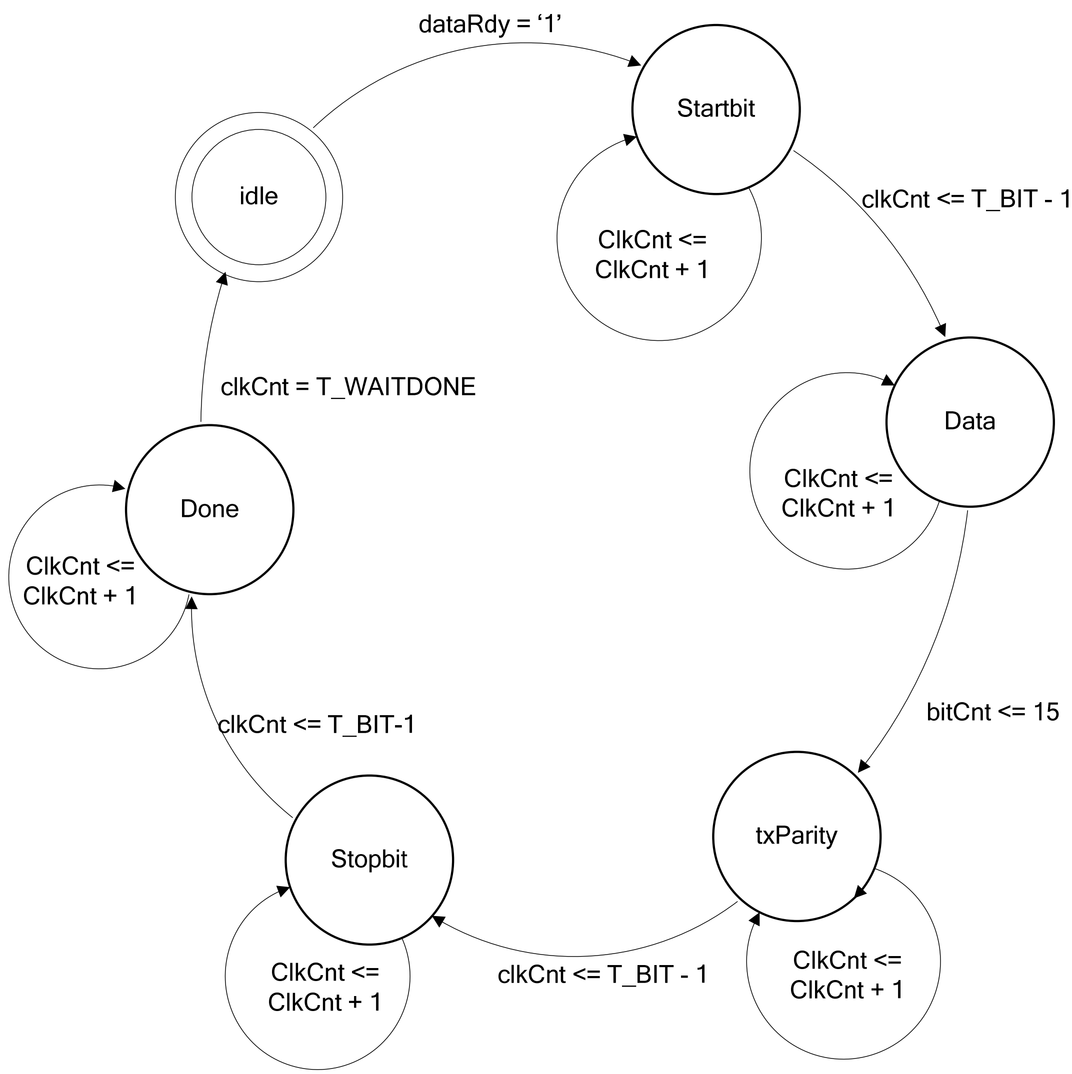


Figure C-2-1: FSM for UART transmitter

Idle	During the idle state the UART transmitter unit waits for the **dataRdy** line to go high. This transition indicates that data is ready to be transmitted commencing to the transmission of the start bit which is a '0'. The start bit is written out on the **serialOut** line. After this had occurred the system transitions to the next state.
Startbit	During this state the UART transmitter waits for 16 enabling tick before commencing to transmit the first data bit on the **serialOut** line. The parity calculation also starts in the state, by means of a XOR calculation. After this occurred the system transitions to the next state.
Data	During the data state the UART transmitter once again waits for 16 enabling ticks before transmitting the next data bit. After the first data bit is transmitted the bit counter is incremented and written out on the **serialOut**. This process is repeated until the bit counter reaches 15. Throughout the transmission of each of the data bits an XOR calculation occurs with systematically determines the parity of the data transmitted. In the event where the bit counter reaches 15 the parity is also transmitted. Immediately following state transmission occurs.
txParity	In the txParity state the UART transmitter once again waits for 16 enabling ticks where after the stop bit which is a '1' is written out on the **serialOut** line. State transition once again occurs.
Stopbit	In the stopbit state the UART transmitter waits for 16 enabling ticks before transitioning to the waitDone state.
waitDone	During the waitDone state an internal counter is set to count to 100 clock cycles. This is done to ensure that the communication controller does not start to receive immediately after transmitting. Keeping in mind this can lead to faulty communication over a half duplex transmission line.

Appendix C.3: Power amplifier communication controller

Figure C-3-1 illustrate the state machine used to design the communication controller situated on the power amplifiers which interfaces with the main controller. In

Table 3-1 the various states are described.

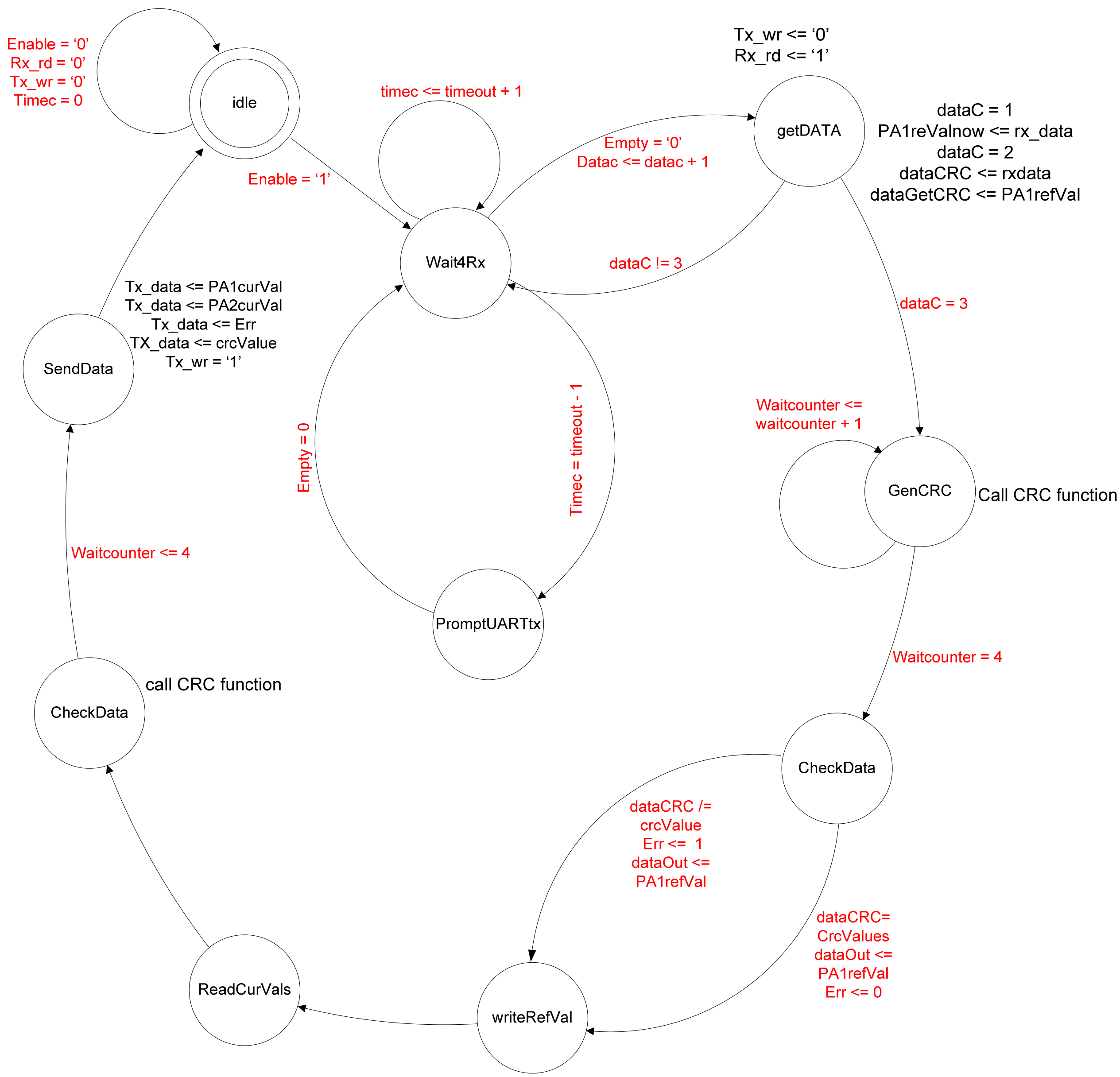


Figure C-3-1: State machine implemented on the power amplifier

Table 3-1: State description

State	State description
Idle	During the **idle** state the communication controller waits to be enabled by die sync signal. Once the communication controller has been enabled, state transition occurs.
Wait4rx	During the **wait4rx** state, a timeout is incremented. Once data becomes available in the FIFO by setting the empty signal to '0' state transition occurs to the **GetData** state. In the case where no data becomes available in the FIFO, the timeout will be reached and state transition will occur to the **PromptUARTtx** state.
GetData	In the **getdata** state, the data available in the FIFO is read out. Once all the data is read out state transition occurs to **GenCRC**.
PromptUart	During this state and error will be flagged, indicating that no data has been received. State transition will occur back to wait4rx.
GenCRC	In the GenCRC state the CRC function is called and the CRC is calculated. State transition occurs to the **CheckError** state.
CheckError	In the event of a CRC mismatch an error will be flagged. State transition will occur to **WriteRefVal**.
WriteRefVal	If no CRC error was flagged the received current reference value will be written into the DPR. If a CRC error was flagged an error will be written into the DPR notifying the power amplifier controller that a communication error has occurred. State transition will occur to **ReadCurVals**.
ReadCurVals	During the ReadCurVals the true current values will be read out of the memory space where after state transition will occur to **CheckData**.
CheckData	A CRC will be calculated in this state by calling the CRC function where after state transition will occur to **SendData**.
SendData	During this state the two true current values will be transmitted and the CRC value. State transition will occur to the default state **idle**

Appendix C.4: Main controller communication controller

Figure C-4-1 illustrate the state machine used to design the communication controller situated on the main controller which interface with a power amplifier. In this section each of the states will be discussed.

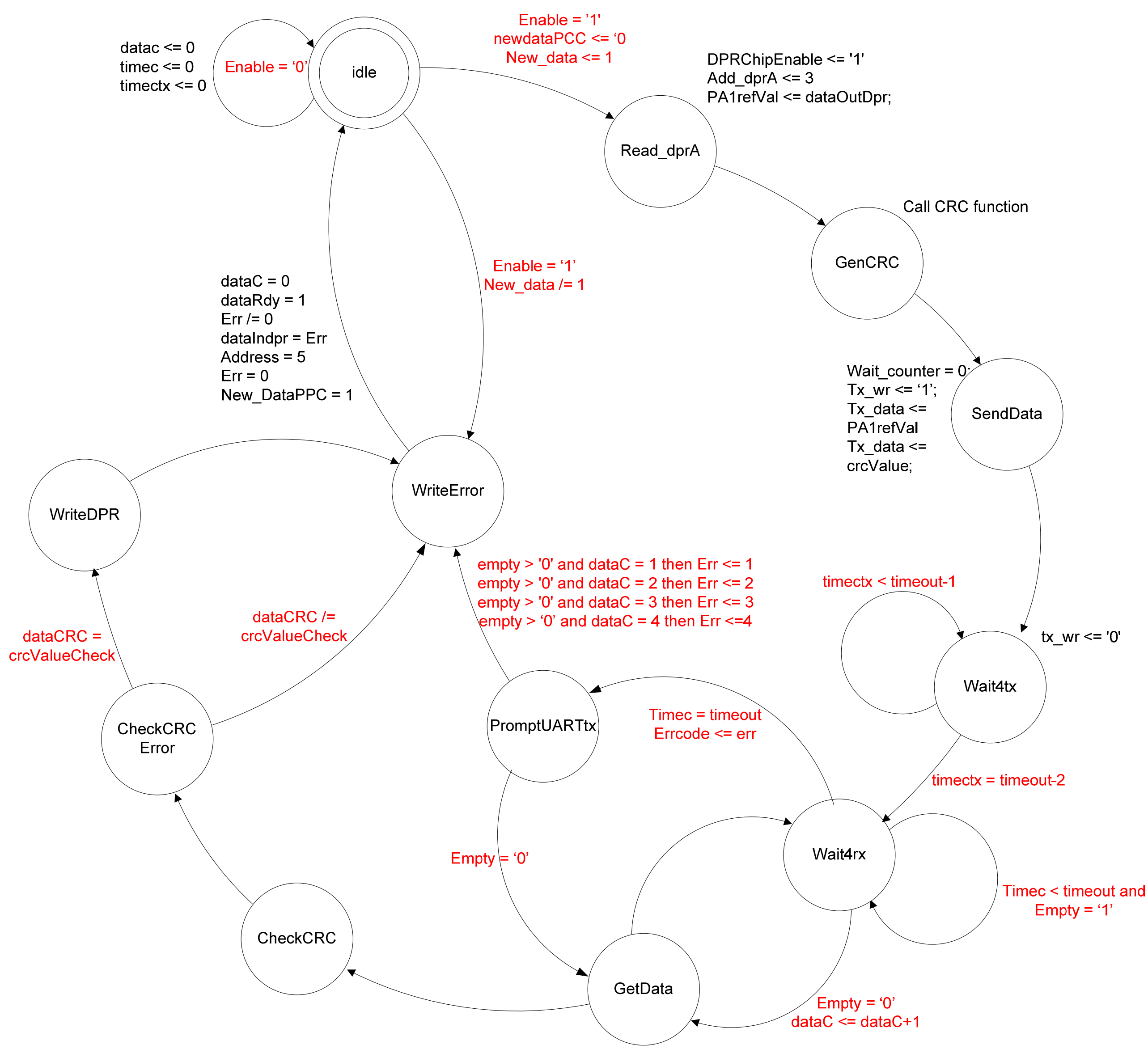


Figure C-4-1: State machine implemented on main controller

Table 4-1: State description

Idle	During the **idle** state the communication controller waits to be enabled by die sync signal. Once the communication controller has been enabled, the **new data** input is checked. This input will be high if new data is available in the DPR. If the new data input is not high an error will be flagged and state transition will occur to **writeError**. If the new data signal is high, state transition will occur to **Read_dprA.**
Read_dprA	In this state the current reference value is read out the DPR where after state transitions occur to GenCRC.
GenCRC	In the GenCRC state the CRC function is called and the CRC is calculated. State transition occurs to the **SendData** state.
SendData	During this state the current reference value and the CRC value is written into the FIFO where after state transition occurs to the **wait4tx** state.
Wait4tx	In the wait4tx state a timeout is incremented until the estimated time it will take for transmission to be complete is reached where after state transmission will occur to the **Wait4rx** state.
Wait4rx	During the **wait4rx** state, a timeout is incremented. Once data becomes available in the FIFO by setting the empty signal to '0' state transition occurs to the **GetData** state. In the case where no data becomes available in the FIFO, the timeout will be reached and state transition will occur to the **PromptUARTtx** state.
GetData	In the **getdata** state, the data available in the FIFO is read out. Once all the data is read out state transition occurs to **CheckCRC**.
PromptUart	In this state errors will be flagged according to the data not received. Where after state transition to the **WriteError** state will occur.
CheckCRC	During the CheckCRC state the CRC function is called after the CRC is calculated state transition occurs to **CheckCRCerror.**
CheckCRCerror	In the event of a CRC mismatch an error will be flagged. State transition will occur to **WriteError**. In the event that no error was flagged the system will transition to **WriteData**.
WriteDPR	During this state the true current values will be written into the DPR where after state transition will occur to **WriteError**.
WriteError	The errors flagged will be written into the DPR where after state transition will

	occur to the default state idle.