Example of EXAM solution

\[ f_{\text{CLK-RX}} = \frac{f_{\text{osc-CLK-in}}}{2 \cdot N_1} \Rightarrow N_1 = 2500 \]
2. Inventing a **counter mod 4** using the plan C2 and a standard **counter mod 16**

→ **Using an up counter** → **Detector 3**

![Diagram of an up counter with annotations]

Load zero when 3 and enabled 0, 1, 2, 3, 0, 1
Load count up

→ **Using a down counter**

![Diagram of a down counter with annotations]

When counting down TC16 is '1' when 0 and CE = '1'
It detects the zero to load 3 in parallel

(The same for the counter mod 8)
The OPC table is like a XOR of many inputs.
4. 8-bit data register using the plan C2 and components counter-mod16

LD  Q^+  CLK
0  Q  Do nothing because the CE = 0 (Keep data)
1  Din  Parallel Load is write data
5. Datapath for the receiver

Chip 1... Chip 5 from P3
Chip 6 is a combinational circuit from P3 (standard arithmetic blocks)

Clock and CD for all the chips: 2 + 3 + 3 + 9 + 8 = 25 D-FF

Byte received: Q(7..0)

Data_out(7..0)
6. Example of state diagram for the control unit

Load the data register and sample the Stop bit

Odd Parity check

Error pulse

Delay 4 clk

Sample start bit

Delay 8 clk

Sample data

Stop bit check

Data available pulse

Idle

R XR = '1'

TC4D = '0'

Del4 = '0'

R XR = '0'

S = '10'

S = '11'

S = '10'

LP = '1'

OPC = '0'

TC8B = '1'

TC8D = '0'

S = '10'

BC = '1'

Del8 = '0'

S = '00'

R XR = '0'

Error = '1'

S = '10'

S = '11'

S = '10'

OPC = '0'

R XR = '1'

R XR = '0'

R XR = '1'

This is an initial state diagram that can be modified on testing design stage
7.

Control unit → 4 D-FF (coding in binary)
Datapath → 25 D-FF
CLK Generator → 13 D-FF

\[ \underline{42 \text{ D-FF}} \]

→ It is possible to use a 10-bit shift register so that the stop bit check and the data write can be solved in two states
→ using components Counter modulo the
Counter modulo can be 4 D-FF instead of 2 depending on the VHDL Synthesiser options
8. Solving the project using a μC

Hardware circuit

15.36 MHz

CLK_RX/μC PIC18F4520

(4.8 kHz)

RBO/INTO to generate baud rate

RX → If an external interrupt is used → RB1/INT1

If it is polled in the main loop → any port pin → RA0

( or the same RB1 used as simple digital input)

Convenient value to generate standard UART transmitter and receiver frequencies using timer peripherals

TRISA 0 0 0 0 0 0 0

Error

TRISB 0 0 0 0 0 1 1

Clk RX

TRISC 0 0 0 0 0 0 0

DA

TRISD 0 0 0 0 0 0 0

(R) not used pins → '0' outputs
9. Software-hardware diagrams and RAM variables of interest

- RX to RX
- RBO/MD/ISR
- Active edge select
- INTEGO = 1
- Clear hardware interrupt
  WHEN EXECUTING INTOIF = 0
  INTOE = 1
- Global interrupt enabled (GIE = 1)
- Interrupt vector

variables for counting bits and time delays depending on current state and var_CLK_flag

- var_RX
- var_DA
- var_Error
- var_Data
- var_CLKD
- var_TCDB
- var_TCB
- var_OPC
- Current state

state logic

write outputs

output logic

read inputs

variables for counting bits and time delays depending on current state and var_CLK_flag

Interrupt vector
Init_system

var_clk_flag = 0?

read_input()
state_logic()
output_log()
write_outputs

ISR

Interrupt when a 5 is detected (depending on the state, the variables for counting interrupts (delays) and bits will be updated.

var_clk_flag = 1

ISR()
10. State logic truth table and flowchart

(RAM variables) (when $f$ is deleted) (next value)

<table>
<thead>
<tr>
<th>RX</th>
<th>TC4D</th>
<th>TC3D</th>
<th>TC9B</th>
<th>OPC</th>
<th>current_state</th>
<th>current_state+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Idle</td>
<td>Idle</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Idle</td>
<td>Delay 4CLK</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Delay 4CLK</td>
<td>Sample start bit</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Sample start bit</td>
<td>Delay 8CLK</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Delay 8CLK</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>Delay 8CLK</td>
<td>Sample data</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>Delay 8CLK</td>
<td></td>
</tr>
</tbody>
</table>

etc.

Thus, the flowchart is using switch-case and then the if-else when required
M. Output logic truth table and flowchart

Depends on current state and RX (only in some states to be able to shift data left)

<table>
<thead>
<tr>
<th>RX</th>
<th>Current state</th>
<th>RAM variables</th>
<th>Data</th>
<th>Error</th>
<th>DA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Delay 4CLK</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sample start bit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Delay 8CLK</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sample Data</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Odd parity check</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stop bit check</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Error pulse</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Data available pulse</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Switch-case

Idle

DA = 0
Error = 0
Data = 0

Sample data

DA = 0
Error = 0
Data = (Data | RX) ≤ 1

⇒ Use the watch window and step by step

Load and shift left in C language
12. Using the Timer 0 to generate baud rate frequencies
13. Programming delays

Source active edge

\[ \text{Source active edge} \]

\[ \text{15.36 MHz} \]

\[ \downarrow \]

\[ 3.84 \text{ MHz} \]

\[ 3.84 \text{ MHz} \left( \frac{1}{N_1} \right) \left( \frac{1}{N_2} \right) = 4.8 \text{ kHz} \]

\[ N_1 \cdot N_2 = 800 \]

For instance: \( N_1 = 16, N_2 = 50 \)

This peripheral is acting as the data path and the clock generator in this application (timing delays)

\[ \text{TCck} = \frac{1}{3.84 \text{ MHz}} = 260,417 \text{ ns} \]

\[ \text{TCGD} = 833.3 \text{ ns} = N_1 \cdot N_2 \cdot \text{TCck} \]

\[ \left( \frac{1}{4.8 \text{ kHz}} \right) \]

\[ \frac{3200}{N_1 = 64; N_2 = 50} \]

\[ \text{TCBD} = 1.67 \text{ ms} = N_1 \cdot N_2 \cdot \text{TCck} \]

\[ N_1 = 64 \]

\[ N_2 = 100 \]