System Architecture for Software Peripherals

Siddharth Choudhuri, Tony Givargis

sid@cecs.uci.edu givargis@uci.edu

Center for Embedded Computer Systems
University of California, Irvine, USA
Outline

- Software Peripherals - Introduction
- Software Peripherals - Advantages
- Related Work
- Proposed System Architecture
- Experimental Setup
- Results
Peripherals - Traditional View

Diagram showing the traditional view of peripherals in a System-on-Chip (SoC) with CPU and peripheral components, detailing Pin I/O and output waveform rate and nature.
Peripherals - Traditional View

- Peripheral is a hardware entity
  - High NRE on redesign
  - Lower flexibility - Hardcoded
  - Cannot be upgraded in field
**Peripherals - Traditional View**

- Peripheral is a hardware entity
  - High NRE on redesign
  - Lower flexibility - Hardcoded
  - Cannot be upgraded in field

- Interface view - outside world
  - Output waveform - rate and nature
Software Peripherals

Traditional View
Software Peripherals

Traditional View

Software Peripheral

General Purpose I/O Hardware
Software Peripherals

**Traditional View**

**Software Peripheral**

- CPU
- Peripheral
- Pin I/O
- Output

**General Purpose I/O Hardware**

- CPU
- Peripheral
- Pin I/O
- Output

System Architecture for Software Peripherals – p.4/24
Software Peripherals

Traditional View

Software Peripheral

Outside World

System Architecture for Software Peripherals – p.4/24
Advantages of Software Peripherals

- Design reusability
Advantages of Software Peripherals

- Design reusability
- Easier upgrades
Advantages of Software Peripherals

- Design reusability
- Easier upgrades
- Reduced manufacturing cost
Advantages of Software Peripherals

- Design reusability
- Easier upgrades
- Reduced manufacturing cost
- Faster design turnaround time
Advantages of Software Peripherals

- Design reusability
- Easier upgrades
- Reduced manufacturing cost
- Faster design turnaround time
Goals

Computation

I/O Communication
Goals

- Applications
- Software Peripherals

Mismatch

- Computation
- I/O Communication
Goals

- Computation
- Applications
- Software Peripherals
- I/O Communication

Mismatch
Goals

Computation

I/O Communication

Software

Peripherals

Applications

Mismatch

System Architecture for Software Peripherals – p.6/24
Goals

Mismatch

Applications

Software Peripherals

Computation

I/O Communication
System architecture that minimizes the mismatch
What favors Software Peripherals?

- Increasing speeds in embedded processors

Example: ARM, PowerPC

Processors have support for register banking, on-board A/D convertors

Example: ARC processors, microcontrollers

Assumptions

- Fast context switch - processors with register banking
- High resolution timer
What favors Software Peripherals?

- Increasing speeds in embedded processors
  - Example: ARM, PowerPC
What favors Software Peripherals?

- Increasing speeds in embedded processors
  - Example: ARM, PowerPC

- Processors have support for register banking, on-board A/D convertors
What favors Software Peripherals?

- Increasing speeds in embedded processors
  - Example: ARM, PowerPC

- Processors have support for register banking, on-board A/D convertors
  - Example: ARC processors, microcontrollers
What favors Software Peripherals?

- Increasing speeds in embedded processors
  - Example: ARM, PowerPC

- Processors have support for register banking, on-board A/D convertors
  - Example: ARC processors, microcontrollers

- Assumptions
What favors Software Peripherals?

- Increasing speeds in embedded processors
  - Example: ARM, PowerPC
- Processors have support for register banking, on-board A/D convertors
  - Example: ARC processors, microcontrollers
- Assumptions
  - Fast context switch - processors with register banking
Related Work

Related Work


- Ubicom [www.ubicom.com](http://www.ubicom.com)
Related Work


- Ubicom [www.ubicom.com](http://www.ubicom.com)
  - IP 3000 family processors
Related Work


- Ubicom [www.ubicom.com](http://www.ubicom.com)
  - IP 3000 family processors

- Triscend [www.triscend.com](http://www.triscend.com)
Related Work


- Ubicom [www.ubicom.com](http://www.ubicom.com)
  - IP 3000 family processors

- Triscend [www.triscend.com](http://www.triscend.com)
  - Fastchip configuration tool
System Architecture

Example: RS 232 - UART

- 8 bits, parity, 1 start-stop bit
System Architecture

Example: RS 232 - UART
- 8 bits, parity, 1 start-stop bit

Computation
- Serialize bits
- Calculate Parity
- Add start, stop bits
Example: RS 232 - UART

- 8 bits, parity, 1 start-stop bit

Computation

- Serialize bits
- Calculate Parity
- Add start, stop bits

Communication

- Transfer bits to Pin I/O
**System Architecture**

- **Peripheral** → **Computation** → **Communication**

### Low Level Peripheral (LLP) Task
- Hard real time
- Communication intensive
- Statically scheduled
- As frequent as the highest operating peripheral
System Architecture

- Low Level Peripheral (LLP) Task
  - Hard real time
  - Communication intensive
  - Statically scheduled
  - As frequent as the highest operating peripheral

- Peripheral Memory Buffer
  - Shared Memory
  - Pool of buffers for each peripheral
System Architecture

- **Low Level Peripheral (LLP) Task**
  - Hard real time
  - Communication intensive
  - Statically scheduled
  - As frequent as the highest operating peripheral

- **Peripheral Memory Buffer**
  - Shared Memory
  - Pool of buffers for each peripheral

- **High Level Peripheral (HLP) Task**
  - Implementation of peripheral
  - Compute intensive
System Architecture

- **Low Level Peripheral (LLP) Task**
  - Hard real time
  - Communication intensive
  - Statically scheduled
  - As frequent as the highest operating peripheral

- **Peripheral Memory Buffer**
  - Shared Memory
  - Pool of buffers for each peripheral

- **High Level Peripheral (HLP) Task**
  - Implementation of peripheral
  - Compute intensive
Scheduling LLP Task

**Input** Task set $T = T_1, T_2, \ldots, T_N$, Context Switch Overhead $CS$

**Output** Schedule layout if successful

- Test taskset $T$ for schedulability $U > N(2^{1/N} - 1)$
Scheduling LLP Task

Input  Task set $T = T_1, T_2, ..., T_N$, Context Switch Overhead $CS$

Output  Schedule layout if successful

- Test taskset $T$ for schedulability $U > N(2^{1/N} - 1)$
- Calculate Hyperperiod $H = LCM(T_1, T_2, ..., T_N)$
Scheduling LLP Task

**Input**  Task set $T = T_1, T_2, ..., T_N$, Context Switch Overhead $CS$

**Output**  Schedule layout if successful

- Test taskset $T$ for schedulability $U > N(2^{1/N} - 1)$
- Calculate Hyperperiod $H = LCM(T_1, T_2, ..., T_N)$
- For each permutation of task set $T_c = permute(T)$ try schedule
Scheduling LLP Task

**Input**  Task set $T = T_1, T_2, \ldots, T_N$, Context Switch Overhead $CS$

**Output**  Schedule layout if successful

- Test taskset $T$ for schedulability $U > N(2^{1/N} - 1)$
- Calculate Hyperperiod $H = \text{LCM}(T_1, T_2, \ldots, T_N)$
- For each permutation of task set $T_c = \text{permute}(T)$ try schedule
- Generate code
Scheduling LLP Task

*Input*  Task set $T = T_1, T_2, ..., T_N$, Context Switch Overhead $CS$

*Output*  Schedule layout if successful

- Test taskset $T$ for schedulability $U > N(2^{1/N} - 1)$
- Calculate Hyperperiod $H = LCM(T_1, T_2, ..., T_N)$
- For each permutation of task set $T_c = permute(T)$ try schedule
- Generate code
Scheduling LLP Task

*Input*  Task set \( T = T_1, T_2, ..., T_N \), Context Switch Overhead \( CS \)

*Output*  Schedule layout if successful

- Test taskset \( T \) for schedulability \( U > N(2^{1/N} - 1) \)
- Calculate Hyperperiod \( H = LCM(T_1, T_2, ..., T_N) \)
- For each permutation of task set \( T_c = \text{permute}(T) \) try schedule
- Generate code

Complexity \( O((N!) \times N \times H/p_{min}) \)

- \( N \) - Number of peripherals
- \( H \) - Hyperperiod
- \( p_{min} \) - Task with minimum periodicity
**Example Schedule**

![Diagram with LLP Tasks and Peripheral1, Peripheral2]

**LLP**: Hard real time
Example Schedule

**LLP**: Hard real time

**HLP**: Soft real time
Example Schedule

**LLP**: Hard real time

**HLP**: Soft real time

**Response Time**: Application task pre-empted due to LLP task

**Slack**: Time between two invocations of LLP task
# Simulated Peripherals

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>HLP Cycles</th>
<th>LLP Cycles</th>
<th>Frequency</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Port</td>
<td>364</td>
<td>64</td>
<td>19200 baud</td>
<td>2</td>
</tr>
<tr>
<td>Keypad</td>
<td>16</td>
<td>29</td>
<td>1 KHz</td>
<td>8</td>
</tr>
<tr>
<td>Timer</td>
<td>10</td>
<td>31</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>PWM</td>
<td>90</td>
<td>34</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>V.34 Modem</td>
<td>7660</td>
<td>32</td>
<td>33600 bps</td>
<td>2</td>
</tr>
</tbody>
</table>

*UART 19200, 8E1 configuration*
# Simulated Peripherals

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>HLP</th>
<th>LLP</th>
<th>Frequency</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cycles</td>
<td>Cycles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial Port</td>
<td>364</td>
<td>64</td>
<td>19200 baud</td>
<td>2</td>
</tr>
<tr>
<td>Keypad</td>
<td>16</td>
<td>29</td>
<td>1 KHz</td>
<td>8</td>
</tr>
<tr>
<td>Timer</td>
<td>10</td>
<td>31</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>PWM</td>
<td>90</td>
<td>34</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>V.34 Modem</td>
<td>7660</td>
<td>32</td>
<td>33600 bps</td>
<td>2</td>
</tr>
</tbody>
</table>

- **UART** 19200, 8E1 configuration
- **Keypad** 4 × 4 keypad
Simulated Peripherals

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>HLP</th>
<th>LLP</th>
<th>Frequency</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cycles</td>
<td>Cycles</td>
<td>baud/Hz</td>
<td></td>
</tr>
<tr>
<td>Serial Port</td>
<td>364</td>
<td>64</td>
<td>19200</td>
<td>2</td>
</tr>
<tr>
<td>Keypad</td>
<td>16</td>
<td>29</td>
<td>1 KHz</td>
<td>8</td>
</tr>
<tr>
<td>Timer</td>
<td>10</td>
<td>31</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>PWM</td>
<td>90</td>
<td>34</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>V.34 Modem</td>
<td>7660</td>
<td>32</td>
<td>33600</td>
<td>2</td>
</tr>
</tbody>
</table>

- **UART** 19200, 8E1 configuration
- **Keypad** 4 × 4 keypad
- **Timer** Configurable 10KHz timer
Simulated Peripherals

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>HLP Cycles</th>
<th>LLP Cycles</th>
<th>Frequency</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Port</td>
<td>364</td>
<td>64</td>
<td>19200 baud</td>
<td>2</td>
</tr>
<tr>
<td>Keypad</td>
<td>16</td>
<td>29</td>
<td>1 KHz</td>
<td>8</td>
</tr>
<tr>
<td>Timer</td>
<td>10</td>
<td>31</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>PWM</td>
<td>90</td>
<td>34</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>V.34 Modem</td>
<td>7660</td>
<td>32</td>
<td>33600 bps</td>
<td>2</td>
</tr>
</tbody>
</table>

- **UART** 19200, 8E1 configuration
- **Keypad** 4 × 4 keypad
- **Timer** Configurable 10KHz timer
- **PWM** Configurable pulse width modulator
## Simulated Peripherals

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>HLP Cycles</th>
<th>LLP Cycles</th>
<th>Frequency</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Port</td>
<td>364</td>
<td>64</td>
<td>19200 baud</td>
<td>2</td>
</tr>
<tr>
<td>Keypad</td>
<td>16</td>
<td>29</td>
<td>1 KHz</td>
<td>8</td>
</tr>
<tr>
<td>Timer</td>
<td>10</td>
<td>31</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>PWM</td>
<td>90</td>
<td>34</td>
<td>10 KHz</td>
<td>1</td>
</tr>
<tr>
<td>V.34 Modem</td>
<td>7660</td>
<td>32</td>
<td>33600 bps</td>
<td>2</td>
</tr>
</tbody>
</table>

- **UART** 19200, 8E1 configuration
- **Keypad** 4 × 4 keypad
- **Timer** Configurable 10KHz timer
- **PWM** Configurable pulse width modulator
- **V.34 Modem** Implementation of V.34 protocol
Experimental Setup

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>HLP</th>
<th>LLP</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Port</td>
<td>364</td>
<td>64</td>
<td>19200</td>
</tr>
<tr>
<td>Keypad</td>
<td>...</td>
<td>..</td>
<td>...</td>
</tr>
<tr>
<td>V.34 Modem</td>
<td>...</td>
<td>..</td>
<td>...</td>
</tr>
</tbody>
</table>
Experimental Setup

- Peripherals Models
- Task Set Generator
- CPU Params
- Schedule Generator
- Application, HLP Task
- Compile and Link
- Binary
- MIPS Simulator
- Statistics

Task Sets:
- \((p_0, e_0, d_0)p_1, e_1, d_1\)...
- \((p_n, e_n, d_n)\)

(Period, Execution Time, Deadline)
- Task T1 \((P_1, E_1, D_1)\)
- Task T2 \((P_2, E_2, D_2)\)
- ...

System Architecture for Software Peripherals – p.13/24
Experimental Setup

Peripherals Models

Task Set Generator

Schedule Generator

CPU Params

Application, HLP Task

intr.c, llp.h

Compile and Link

Binary

MIPS Simulator

Statistics

p0, e0, d0
p1, e1, d1
...
pi, ei, di

Task Sets

System Architecture for Software Peripherals – p.13/24
Experimental Setup

```c
int T[] = {
    ...
    ...
};

int schedule[] = {
    ...
    ...
};

#define HYPERPERIOD
#define SPI  0
#define KEYPAD 1
...
```

System Architecture for Software Peripherals – p.13/24
Experimental Setup

- Schedule Generator
- CPU Params
- Task Set Generator
- Application, HLP Task
- Compile and Link
- MIPS Simulator
- Statistics

Perihral Models

p0,e0,d0
p1,e1,d1
...
py,ey,dy

Task Sets
Experimental Setup

Diagram:

- Experimental Setup
- Schedule Generator
- Task Set Generator
- Periphral Models
- CPU Params
- Application, HLP Task
- Intr.c, llp.h
- Compile and Link
- Binary
- MIPS Simulator
- Statistics
- p0, e0, d0
  p1, e1, d1
  ...
  pn, en, dn
- Task Sets

System Architecture for Software Peripherals – p.13/24
Results - Processor Overhead

- HLP
- LLP

Percentage Overhead vs. Frequency:

- Serial Port
- Modem
- Keypad
- Timer

Chart shows processor overhead for different peripherals and frequencies.
Results - Available Slack

Slack Distribution

Number of cycles

Slack (in cycles)

100 MHz
150 MHz
200 MHz
250 MHz

System Architecture for Software Peripherals – p.15/24
# Results - Processor Utilization

<table>
<thead>
<tr>
<th>CPU (MHz)</th>
<th>Hyper-period</th>
<th>Error %</th>
<th>HLP (%)</th>
<th>LLP (%)</th>
<th>Intr. cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3900000</td>
<td>-0.15</td>
<td>0.80</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>150</td>
<td>21450000</td>
<td>-0.10</td>
<td>-1.38</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>200</td>
<td>7800000</td>
<td>-0.15</td>
<td>0.80</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>250</td>
<td>9750000</td>
<td>-0.15</td>
<td>-0.80</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
## Results - Processor Utilization

<table>
<thead>
<tr>
<th>CPU (MHz)</th>
<th>Hyper-period</th>
<th>Serial</th>
<th>Modem</th>
<th>Keypad</th>
<th>PWM</th>
<th>Timer</th>
<th>HLP (%)</th>
<th>LLP (%)</th>
<th>Intr. cycles</th>
<th>Total cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3900000</td>
<td>-0.15</td>
<td>0.80</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>7.13</td>
<td>2.98</td>
<td>2869</td>
<td>3999900</td>
</tr>
<tr>
<td>150</td>
<td>21450000</td>
<td>-0.10</td>
<td>-1.38</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>4.88</td>
<td>2.08</td>
<td>10628</td>
<td>21834904</td>
</tr>
<tr>
<td>200</td>
<td>7800000</td>
<td>-0.15</td>
<td>0.80</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.60</td>
<td>1.51</td>
<td>2869</td>
<td>7899198</td>
</tr>
<tr>
<td>250</td>
<td>9750000</td>
<td>-0.15</td>
<td>-0.80</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.89</td>
<td>1.21</td>
<td>2869</td>
<td>9847698</td>
</tr>
</tbody>
</table>
Results - Response Time

- Best case 32 cycles
Results - Response Time

- Best case 32 cycles
- Worst case 190 cycles
Results - Response Time

- Best case 32 cycles
- Worst case 190 cycles
- Response time bounded by \[\left[ \min_{i=1}^{N} e_i, \sum_{i=1}^{N} e_i \right]\]
  - \(e_i\) - LLP task execution time of peripheral \(P_i\).
  - \(N\) - Number of peripherals
Conclusion

- Proposed a system architecture for software peripherals
  - Design methodology
  - Scheduling
  - Code generation

- Software Peripherals can be realized on embedded processors with right architecture support
  - Fast context switch
  - High resolution timers
  - On chip A/D support

- Minimal impact on user level applications
Thank You
Backup slides
```c
intr_handle() {
    switch(schedule[i]) {
        case SPI: /* 0 */
            ...
            outp(spi_buf[start]);
            start = (start+1)%SPI_BUF;
            break;
        case KEYPAD: /* 1 */
            ...
            ...
            ...
            ...
        case PWM: /* 3 */
            ...
    }
    i = (i + 1) % HYPERPERIOD
    set_timer(T[i] - T[i-1]);
}
```

```
Peripheral Buffers

spi_buf[]

Memory

Timer Layout

Schedule Layout

Hyperperiod

T[]

Hyperperiod

schedule[]

SPI_BUF

start

end
```
Results - Response Time

Response Time Distribution

- 100, 200, 250 MHz
- 150 MHz

Number vs. Response Interval (in cycles)
Results - Response Time

- Best case 32 cycles
- Worst case 190 cycles
- Response time bounded by \[ \left[ \min_{i=1}^{N} e_i, \sum_{i=1}^{N} e_i \right] \]
  - \( e_i \) - LLP task execution time of peripheral \( P_i \).
  - \( N \) - Number of peripherals
# Results - Processor Overhead

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>100MHz LLP</th>
<th>HLP</th>
<th>Total</th>
<th>150MHz LLP</th>
<th>HLP</th>
<th>Total</th>
<th>200MHz LLP</th>
<th>HLP</th>
<th>Total</th>
<th>250MHz LLP</th>
<th>HLP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Port</td>
<td>1.22</td>
<td>0.70</td>
<td>1.92</td>
<td>0.82</td>
<td>0.46</td>
<td>1.28</td>
<td>0.60</td>
<td>0.34</td>
<td>0.94</td>
<td>0.48</td>
<td>0.35</td>
<td>0.83</td>
</tr>
<tr>
<td>Modem</td>
<td>1.07</td>
<td>6.43</td>
<td>7.50</td>
<td>0.71</td>
<td>4.29</td>
<td>5.00</td>
<td>0.53</td>
<td>3.22</td>
<td>3.75</td>
<td>0.43</td>
<td>2.57</td>
<td>3.00</td>
</tr>
<tr>
<td>Keypad</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>Timer</td>
<td>0.31</td>
<td>0.10</td>
<td>0.41</td>
<td>0.20</td>
<td>0.07</td>
<td>0.27</td>
<td>0.15</td>
<td>0.05</td>
<td>0.20</td>
<td>0.12</td>
<td>0.04</td>
<td>0.16</td>
</tr>
<tr>
<td>PWM</td>
<td>0.34</td>
<td>0.11</td>
<td>0.45</td>
<td>0.17</td>
<td>0.08</td>
<td>0.25</td>
<td>0.22</td>
<td>0.06</td>
<td>0.28</td>
<td>0.13</td>
<td>0.05</td>
<td>0.18</td>
</tr>
</tbody>
</table>
## Results - Processor Overhead

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>100MHz LLP</th>
<th>100MHz HLP</th>
<th>100MHz Total</th>
<th>150MHz LLP</th>
<th>150MHz HLP</th>
<th>150MHz Total</th>
<th>200MHz LLP</th>
<th>200MHz HLP</th>
<th>200MHz Total</th>
<th>250MHz LLP</th>
<th>250MHz HLP</th>
<th>250MHz Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Port</td>
<td>1.22</td>
<td>0.70</td>
<td>1.92</td>
<td>0.82</td>
<td>0.46</td>
<td>1.28</td>
<td>0.60</td>
<td>0.34</td>
<td>0.94</td>
<td>0.48</td>
<td>0.35</td>
<td>0.83</td>
</tr>
<tr>
<td>Modem</td>
<td>1.07</td>
<td>6.43</td>
<td>7.50</td>
<td>0.71</td>
<td>4.29</td>
<td>5.00</td>
<td>0.53</td>
<td>3.22</td>
<td>3.75</td>
<td>0.43</td>
<td>2.57</td>
<td>3.00</td>
</tr>
<tr>
<td>Keypad</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>Timer</td>
<td>0.31</td>
<td>0.10</td>
<td>0.41</td>
<td>0.20</td>
<td>0.07</td>
<td>0.27</td>
<td>0.15</td>
<td>0.05</td>
<td>0.20</td>
<td>0.12</td>
<td>0.04</td>
<td>0.16</td>
</tr>
<tr>
<td>PWM</td>
<td>0.34</td>
<td>0.11</td>
<td>0.45</td>
<td>0.17</td>
<td>0.08</td>
<td>0.25</td>
<td>0.22</td>
<td>0.06</td>
<td>0.28</td>
<td>0.13</td>
<td>0.05</td>
<td>0.18</td>
</tr>
</tbody>
</table>
### Results - Processor Utilization

<table>
<thead>
<tr>
<th>CPU (MHz)</th>
<th>Hyper-period</th>
<th>Serial Error %</th>
<th>Modem Error %</th>
<th>Keypad Error %</th>
<th>PWM Error %</th>
<th>Timer Error %</th>
<th>HLP (%)</th>
<th>LLP (%)</th>
<th>Intr. (%)</th>
<th>Total Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3900000</td>
<td>-0.15</td>
<td>0.80</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>7.13</td>
<td>2.98</td>
<td>2869</td>
<td>3999900</td>
</tr>
<tr>
<td>150</td>
<td>21450000</td>
<td>-0.10</td>
<td>-1.38</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>4.88</td>
<td>2.08</td>
<td>10628</td>
<td>21834904</td>
</tr>
<tr>
<td>200</td>
<td>7800000</td>
<td>-0.15</td>
<td>0.80</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.60</td>
<td>1.51</td>
<td>2869</td>
<td>7899198</td>
</tr>
<tr>
<td>250</td>
<td>9750000</td>
<td>-0.15</td>
<td>-0.80</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.89</td>
<td>1.21</td>
<td>2869</td>
<td>9847698</td>
</tr>
</tbody>
</table>
## Results - Processor Utilization

<table>
<thead>
<tr>
<th>CPU (MHz)</th>
<th>Hyper-period</th>
<th>Error %</th>
<th>HLP (%)</th>
<th>LLP (%)</th>
<th>Intr. cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3900000</td>
<td>-0.15</td>
<td>0.80</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>150</td>
<td>21450000</td>
<td>-0.10</td>
<td>-1.38</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>200</td>
<td>7800000</td>
<td>-0.15</td>
<td>0.80</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>250</td>
<td>9750000</td>
<td>-0.15</td>
<td>-0.80</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>