Dynamic Headlight Leveling System

Software System Design

Glenn Denton

May 11, 2010
System Modules

Control Module
- MCU: 9S12DP512
  - 24 MHz bus clock
  - 2655 bytes RAM
  - 11329 bytes ROM
- Kernel: uC/OS-II
  - 1 ms tick period
  - 3 ISRs
  - 3 Semaphore
  - 1 Flags
  - 2 Queues
  - 1 Memory Partition

Headlight Module
- MCU: 9S12C128
  - 24 MHz bus clock
  - 1492 bytes RAM
  - 5296 bytes ROM
  - 100 bytes EEPROM
- Kernel: uC/OS-II
  - 1 ms tick period
  - 1 ISR
  - 3 Semaphores
  - 2 Queues
  - 1 Memory Partition
# Control Module Task Design

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Priority</th>
<th>Task Time (max)</th>
<th>Task Time (Ave)</th>
<th>Period</th>
<th>Ave. Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartTask</td>
<td>Initializes System</td>
<td>4</td>
<td>N/A</td>
<td>N/A</td>
<td>Once</td>
<td>N/A</td>
</tr>
<tr>
<td>CANTxTask</td>
<td>Transmits data over CAN.</td>
<td>5</td>
<td>200 us</td>
<td>100 us</td>
<td>Sporadic</td>
<td>0.01</td>
</tr>
<tr>
<td>CANRxTask</td>
<td>Waits for semaphore from CANRxISR, sends data to Control Task</td>
<td>6</td>
<td>450 us</td>
<td>150 us</td>
<td>Sporadic</td>
<td>0.08</td>
</tr>
<tr>
<td>ControlTask</td>
<td>Decides Headlight level, controls sending of data over CAN</td>
<td>7</td>
<td>400 us</td>
<td>200 us</td>
<td>Sporadic</td>
<td>0.08</td>
</tr>
<tr>
<td>SensorTask</td>
<td>Poles accelerometer and ADC, performs signal processing</td>
<td>8</td>
<td>1.5 ms</td>
<td>1 ms</td>
<td>5 ms</td>
<td>0.2</td>
</tr>
<tr>
<td>BusMonitorTask</td>
<td>Monitors state of CAN system</td>
<td>9</td>
<td>100 us</td>
<td>50 us</td>
<td>20 ms</td>
<td>0.00</td>
</tr>
<tr>
<td>ModCntISR</td>
<td>5 ms hardware interrupt</td>
<td>--</td>
<td>15 us</td>
<td>14.8 us</td>
<td>5 ms</td>
<td>0.00</td>
</tr>
<tr>
<td>CANTxISR</td>
<td>When Tx Interrupt set, will interrupt when TX buffers available</td>
<td>4</td>
<td>41 us</td>
<td>10 us</td>
<td>Sporadic</td>
<td>&lt; 0.1%</td>
</tr>
<tr>
<td>CANRxISR</td>
<td>ISR pending on CANRX interrupt</td>
<td>--</td>
<td>50 us</td>
<td>10 us</td>
<td>Sporadic</td>
<td>&lt; 0.1%</td>
</tr>
</tbody>
</table>

**Total Ave. Load**: 30.00%
Control Module Tasks: SensorTask

- Takes samples from the Accelerometer and ADC
  - Accelerometer sampled every 5 ms
  - ADC sampled every 10 ms
- Performs signal processing
  - Converts into accelerometer data into angle and magnitude
  - Filters data with 2 pole band pass filter
- Sets ControlFlag with ACCELRDY or UIRDY flag
- Pends on ModCntISR 5 ms timer
- Priority: 8
- About 1.5 ms execution time
Control Module Tasks: CANTxTask

- Pends on TxGrp Semaphore
- Data passed to task on CANMem Heap with space for up to 10 CAN Frames
- Sets an interrupt if MSCAN buffers are full
- Period: Sporadic
- Execution time: ~100 us
- Priority: 5
Control Module Tasks: CANRxTask

- Pends on Semaphore from CANRxISR
- Sends data to appropriate task through CANMem Heap
- Priority: 6
- Execution Time: ~150 us
- Period: Sporadic
Control Module Tasks: ControlTask

- Pends on CtrlGrp Flag
- Uses input from SensorTask to determine headlight level, transmits result to Headlight Module through CANTxTask
- Also takes signals from BusMonitorTask regarding state of CAN, and CANRxTask
- Period: Sporadic, < 5ms
- Execution time: ~300 us
- Priority: 7
Control Module ISRs: ModCntISR

- Utilizes Modulus Down Counter from 9S12DP512 Enhanced Capture Timer module
- Creates 5 ms interrupt for accurate sample time for AccelTask
- Execution Time: 14.8 us
- Period: 5 ms
Control Module ISRs: CANRxISR

- Hardware Interrupt from CAN0 Receive in MSCAN module
- Copies received transmission and sets RxISR Semaphore for CANRxTask
- Reads Rx buffer and transfers data to CANRxTask to prevent loss of Rx foreground buffer
- Period: Sporadic
- Execution Time: ~10 us
Control Module ISRs: CANTxISR

- Sets TxGrp Semaphore to signal a free Tx buffer for CANTxTask
- Only occurs if CANTxTask detects no available Tx buffers and sets TISR bit in MSCAN module
- Execution time: ~100 us
# Headlight Module Task Design

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Priority</th>
<th>Task Time (max)</th>
<th>Task Time (Ave)</th>
<th>Period</th>
<th>Ave. Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartTask</td>
<td>Initializes System</td>
<td>4</td>
<td>N/A</td>
<td>N/A</td>
<td>Once</td>
<td>N/A</td>
</tr>
<tr>
<td>ControlTask</td>
<td>Uses data from Control Module to determine direction, and amount to move headlight</td>
<td>7</td>
<td>300 us</td>
<td>200 us</td>
<td>Sporadic</td>
<td>0.00</td>
</tr>
<tr>
<td>MotorTask</td>
<td>Receives direction, time and derivative true or false from ControlTask</td>
<td>9</td>
<td>4 ms</td>
<td>3 ms</td>
<td>Sporadic</td>
<td>0.5</td>
</tr>
<tr>
<td>CANTxTask</td>
<td>Used to control transmission of messages over CAN</td>
<td>6</td>
<td>200 us</td>
<td>100 us</td>
<td>Sporadic</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>CANRxTask</td>
<td>Controls received data from CAN</td>
<td>5</td>
<td>200 us</td>
<td>150 us</td>
<td>Sporadic</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>BusMonitorTask</td>
<td>Monitors state of MSCAN, attempts to restart MSCAN if errors occur</td>
<td>8</td>
<td>150 us</td>
<td>100 us</td>
<td>20 ms</td>
<td>0.01</td>
</tr>
<tr>
<td>CANRxISR</td>
<td>Interrupt from MSCAN Rx flag</td>
<td>--</td>
<td>14.8 us</td>
<td>14 us</td>
<td>Sporadic</td>
<td>&lt; 0.01</td>
</tr>
</tbody>
</table>

Total Load 51.00%
Headlight Module Tasks

ControlTask

- Receives data from Control Module over CAN
- Decides time on, direction, and derivative control TRUE or FALSE, sends to MotorTask
- Execution Time: 300 us
- Period: Sporadic
- Priority: 7
Headlight Module
CANRxISR

- Hardware Interrupt from CAN Receive in MSCAN module
- Copies received transmission and notifies ControlTask
- Data passed to ControlTask via CANHeap partition
- Period: Sporadic
- Execution Time: ~50 us
Headlight Module
CANTxTask, RxTask, BusTask

- Tasks are functionally identical to ControlModule tasks except:
  - CANTxTask pends on Queue, not Semaphore and does not set CANTxISR if hardware buffers are full
## Control Module Libraries

<table>
<thead>
<tr>
<th>Module</th>
<th>Tasks and ISRs</th>
<th>Public Functions and Data</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>StartTask, AccelTask, UITask, CANTask, ControlTask, CAN_RxISR</td>
<td>ControlFlag group</td>
<td>Glenn Denton</td>
</tr>
<tr>
<td>MMA7455_Spi.c</td>
<td></td>
<td>MMA7455Init(), MMA7455Read_8b(), MMA7455Read_10b(), MMA7455Write()</td>
<td>Glenn Denton</td>
</tr>
<tr>
<td>ATD.c</td>
<td></td>
<td>ATDInit(), UiAngle()</td>
<td>Glenn Denton</td>
</tr>
<tr>
<td>math.c</td>
<td></td>
<td>ISO standard C library</td>
<td>--</td>
</tr>
<tr>
<td>CAN.c</td>
<td></td>
<td>CANInit(), CANTx(), CANRx(), CANCheckStatus(), CANSetStatus(), CANFillId(), CANSetISR()</td>
<td>Glenn Denton, parts based on CAN.c by Tim Jackson</td>
</tr>
<tr>
<td>Interrupt.c</td>
<td></td>
<td>ModCntInit(), Wait5ms(), OC0Init(), MSPend()</td>
<td>Glenn Denton, MSPend derived from OC0Isr by Todd Morton</td>
</tr>
<tr>
<td>DSP.c</td>
<td></td>
<td>HPFilter_SP(), LPFilter_SP(), BPFilter_2P(), GetPolar(), AvgFilter(), GetDiff()</td>
<td>Glenn Denton</td>
</tr>
<tr>
<td>uC/OS</td>
<td></td>
<td>--</td>
<td>Jean Lebrosse</td>
</tr>
</tbody>
</table>
Control Module Data Flow

- SensorTask()
- BusMonitorTask()
- CANRxTask()

Control Flag Group:
- UIData
- AccelData
- (CAN_Frame *)

Control Task:

- OSFlagPost()
- OSFlagPend()
- OSSemPend()

CANTxTask:

- OSFlagPost()
- OSSemPend()

CANRxISR:

- OSFlagPost()
Control Module
CANRxISR Dataflow

CAN.c

main.c

CAN_Rx()

CANRxISR()

RxIsrGrp

OSSemPost()

CAN_Rx_Frame

OSSemPend()

CANRxTask()
# Headlight Module Libraries

<table>
<thead>
<tr>
<th>Module</th>
<th>Tasks and ISRs</th>
<th>Public Functions and Data</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>StartTask, ControlTask, CANRxISR, CANTxTask, CANRxTask, BusMonitorTask, MotorTask</td>
<td>CtrlGrp Semaphore, CAN_FRAME pointer</td>
<td>Glenn Denton</td>
</tr>
<tr>
<td>PWM.c</td>
<td></td>
<td>PWM0Init(), ChangeAngle()</td>
<td>Glenn Denton</td>
</tr>
<tr>
<td>CAN.c</td>
<td></td>
<td>CANInit(), CANTx(), CANRx(), CANSetISR(), CANSetStatus(), CANCheckStatus(), CANFillId()</td>
<td>Glenn Denton, parts based on CAN.c by Tim Jackson</td>
</tr>
<tr>
<td>EEPROM.c</td>
<td></td>
<td>EEPROM_Save(), EEPROM_Read, EEPROMInit()</td>
<td>Glenn Denton, based on ProgEEPROM.c by Todd Morton</td>
</tr>
<tr>
<td>uC/OS</td>
<td>--</td>
<td>--</td>
<td>Jean Lebrosse</td>
</tr>
</tbody>
</table>
Headlight Module ControlTask Dataflow

main.c

CANRxTask() → Control Group

OSSemPend()

Control Task

25LC010A Serial EEPROM

EEPROMWrite() *EEPROMRead()

*EEPROMRead() only occurs at startup.

Motor Group

Motor Task

Motor Data

ChangeAngle()

PWM.c

Stepper Motor

*EEPROMWrite()