### 1. Typical Design/Programming Tasks

- Configure and use digital and analog peripherals (ADC, DAC, GPIO, Timers, PWM).

- Write code for periodic sampling using timers and interrupts.

- Interface input/output pins for sensors and actuators.

- Use serial communication (USB UART) for data logging or debugging.

- Implement interrupt-driven event handling for time-sensitive tasks.

- Control actuators (e.g., motors) with PWM signals.

- Buffer data and transfer to other systems (e.g., PC via UART).

\*\*\*

### 2. Example Topics and Approaches

#### (A) Configuring and Using GPIO Pins

\*\*Digital Output: Toggling an LED\*\*

```c

#include <project.h>

int main(void)

{

CyGlobalIntEnable; // Enable interrupts

for(;;)

{

LED\_Write(1); // Turn ON

CyDelay(500);

LED\_Write(0); // Turn OFF

CyDelay(500);

}

}

```

- Place a digital output pin and name it "LED".

- Assign pin in the design schematic[5][6].

\*\*Digital Input with Output Mirror\*\*

```c

for(;;)

{

OutputPin\_Write(InputPin\_Read()); // Mirrors input to output

}

```

- Place both an input and output pin in the schematic[5].

\*\*\*

#### (B) Analog Input (ADC) and Output (DAC)

\*\*Periodic Analog Sampling via Timer Interrupt:\*\*

- Add a SAR ADC and a Timer. Tie Timer's terminal count (TC) output to ADC Start-of-Conversion (SOC) in schematic.

- Add Interrupt component connected to ADC End-of-Conversion (EOC).

\*\*Sample Timer Interrupt Handler:\*\*

```c

// Define buffer globally

uint16 adcBuffer[256];

uint8 idx = 0;

CY\_ISR(ADC\_ISR\_Handler) {

adcBuffer[idx++] = ADC\_Read16();

if(idx >= 256) idx = 0;

}

int main(void) {

CyGlobalIntEnable;

ADC\_Start();

Timer\_Start();

isr\_ADC\_StartEx(ADC\_ISR\_Handler);

for(;;) { /\* Main Loop \*/ }

}

```

- Start both ADC and Timer.

- CyGlobalIntEnable enables global interrupt handling.

- ADC samples are stored to buffer each interrupt[7][3].

\*\*Analog Output (DAC example):\*\*

```c

DAC\_Start();

DAC\_SetValue(adcValue >> 4); // Match resolution if needed

```

\*\*\*

#### (C) Using a Timer to Generate Periodic Interrupts

\*\*Common use-cases:\*\* creating sampling intervals, blinking an LED, or periodic tasks.

\*\*Schematic:\*\*

- Timer component outputs interrupt signal to Interrupt component.

- Interrupt triggers at terminal count (every period).

\*\*Timer Interrupt Example:\*\*

```c

CY\_ISR(Timer\_ISR\_Handler)

{

LED\_Write(!LED\_Read());

Timer\_ClearInterrupt(Timer\_INTR\_MASK\_TC);

}

int main(void)

{

CyGlobalIntEnable;

Timer\_Start();

isr\_Timer\_StartEx(Timer\_ISR\_Handler);

for(;;) { }

}

```

- Set up in Component Catalog: configure timer period for desired frequency (e.g., 1s for 1Hz blink)[2][8][1][9].

\*\*\*

#### (D) Generating PWM Signals

- PWM for motor speed control, signal generation, etc.

\*\*Typical setup:\*\*

- Place PWM component, assign output to pin.

- Configure: set frequency (period & clock divider), initial duty cycle in schematic.

\*\*PWM Control in Code:\*\*

```c

PWM\_Start();

PWM\_WriteCompare(128); // Sets duty cycle (0-255 for 8 bits)

```

To change duty cycle in response to sensor input:

```c

PWM\_WriteCompare(newValue);

```

- For continuous changes, run this logic in a timer interrupt or main loop[10][4].

\*\*\*

#### (E) Implementing Interrupt Service Routines (ISRs)

- Use `CY\_ISR` macro to define and link handler.

- In main, call `isr\_<ComponentName>\_StartEx(<ISR Handler Function>);`

- Use `CyGlobalIntEnable` to enable interrupts globally[11][2].

\*\*\*

#### (F) Serial Communication (USB UART)

- For sending debug or logged data to a PC.

- Place USBUART component, start in code:

```c

USBUART\_Start();

if(USBUART\_GetConfiguration())

USBUART\_PutString("Ready\r\n");

```

- Send values in ISR or main loop as needed.

\*\*\*

### 3. Potential Lab/Test Prompt Examples

- Create a schematic that periodically samples an analog input and writes digital/analog output.

- Implement code to blink an LED at 1 Hz using a timer and ISR.

- Change PWM duty cycle in response to a button press or ADC value.

- Log ADC readings via UART upon timer interrupt.

- Output the value on an input pin to an output pin each interrupt.

- Implement debouncing on a digital input using timer interrupts.

\*\*\*

### 4. Key PSoC Code Patterns

\*\*General Structure:\*\*

1. \*\*Place Hardware Components and Configure Parameters in Schematic\*\*

- Timers, PWM, ADC, Interrupt, IO Pins

2. \*\*Assign Pins\*\*

- Use “.cydwr” to assign physical pins

3. \*\*Write Code (main.c)\*\*

- Start peripherals: `Component\_Start()`

- Enable interrupts: `CyGlobalIntEnable`

- Attach ISRs: `isr\_Component\_StartEx(ISR\_Handler)`

- Main event loop

### Section 3: Light Sensor and Filter Questions

\*\*\*

#### 1. Fundamentals of Light Sensors

\*\*Photodiodes\*\*

- Semiconductor device converting light into current.

- Operates in reverse bias ("photoconductive mode").

- Fast response, low sensitivity; typical for high-speed optical applications.

- Output: Photocurrent proportional to intensity of incident light.

\*\*Phototransistors\*\*

- Transistor acting as a current amplifier controlled by incident light.

- Much higher sensitivity than photodiodes.

- Slower response; better suited for low light detection.

\*\*Practical Application\*\*

- Use sensors to detect the position of a projected line for robot navigation.

- Place sensor below robot facing the projected road; maximize signal-to-noise ratio[1].

\*\*\*

#### 2. Signal Conditioning and Filters

\*\*Transimpedance Amplifier (TIA)\*\*

- Converts photocurrent to voltage.

- Op-amp in inverting configuration with feedback resistor $$ R\_f $$.

$$ V\_{out} = I\_{ph} \times R\_f $$

- Feedback capacitor for noise reduction or to limit bandwidth.

\*\*Filters\*\*

- Used to remove unwanted frequency components (e.g., ambient light, power line noise).

- Key types:

- \*\*Low-Pass Filter:\*\* Retains DC and low frequencies, attenuates higher frequencies.

$$ f\_c = \frac{1}{2\pi RC} $$

- \*\*High-Pass Filter:\*\* Removes DC, passes higher frequencies.

$$ f\_c = \frac{1}{2\pi RC} $$

- \*\*Band-Pass Filter:\*\* Combination, passes a range near line’s modulation frequency.

\*\*Example Circuit\*\*

- Photodiode → TIA → Filter (RC or Sallen-Key, active) → ADC input on PSoC[2].

\*\*\*

#### 3. Measurement and Design Considerations

- \*\*Dark Current:\*\* Current under no illumination; minimize by using low-noise op-amps.

- \*\*Reverse Bias:\*\* Improves sensor speed, but increases dark current.

- \*\*Matching Spectral Response:\*\* Choose photodiode/phototransistor matching projector wavelength (e.g. red if using red lines).

- \*\*Multiple Sensors:\*\* Spatial array ("constellation") of detectors for robust line detection and turns negotiation.

- \*\*Breadboard Prototyping:\*\* Always prototype before PCB fabrication; duplicate circuits for reliability.

\*\*\*

#### 4. Lab Measurement Approach

- Design circuit on breadboard with provided sensor and op-amp.

- Use function generator and scope to drive/project light if simulating ambient conditions.

- Connect sensor output (after amplifier/filter) to PSoC analog input, digitize signal.

- Toggle LEDs when each sensor is over a line, confirming digital output.

\*\*Example PSoC Code for Sensor Detection\*\*

```c

int value = ADC\_Read();

if(value > threshold)

LED\_Write(1); // Over line

else

LED\_Write(0); // Not over line

```

Assign digital pins to LEDs and interface directly with PSoC analog input, setting appropriate threshold.

\*\*\*

#### 5. Typical Filter and Sensor Questions

- Why use an op-amp for light sensor signals instead of a direct connection to an ADC?

- How does adjusting the feedback resistor $$ R\_f $$ affect amplifier sensitivity?

- What filter configuration best removes ambient flicker from fluorescent lighting (50/60Hz)?

- How do you characterize sensor linearity and response time?

- How does the use of multiple sensors improve robot navigation accuracy?

- Why is matching sampling rate in ADC important for light sensor data?

\*\*\*

#### 6. Practical Troubleshooting Tips

- Always confirm circuit orientation and pinout for photodiodes/transistors.

- Ground loops and stray light sources can inject noise; shield and use proper cable routing.

- Test each filter stage separately: sweep function generator, observe frequency response.

- Use oscilloscope to fine-tune threshold settings before coding PSoC logic.

\*\*\*

#### 7. Example Calculation and Circuit Setup

- Given values for $$ R $$ and $$ C $$, calculate cutoff frequency for low-pass filter.

$$ f\_c = \frac{1}{2\pi RC} $$

- For photodiode current of $$ 1 \, \mu\text{A} $$ and feedback resistor $$ 100\,k\Omega $$:

$$ V\_{out} = 1\,\mu\text{A} \times 100\,k\Omega = 0.1\,V $$

\*\*\*

### Photodiode Calculations

#### 1. Photocurrent from Incident Light

- Given: Photodiode responsivity $$ R $$ (A/W), incident light power $$ P\_{in} $$ (W)

- Formula: $$ I\_{ph} = R \times P\_{in} $$

- Example:

- Photodiode responsivity: $$ R = 0.6 $$ A/W (typical for Si at 850 nm)

- Incident light power: $$ P\_{in} = 50 $$ μW $$ = 50 \times 10^{-6} $$ W

- $$ I\_{ph} = 0.6 \times 50 \times 10^{-6} = 30 $$ μA

#### 2. Transimpedance Amplifier Output

- Given feedback resistor $$ R\_f $$, photocurrent $$ I\_{ph} $$:

- Formula: $$ V\_{out} = I\_{ph} \times R\_f $$

- Example:

- $$ I\_{ph} = 30 $$ μA

- $$ R\_f = 100 $$ kΩ $$ = 100,000 $$ Ω

- $$ V\_{out} = 30 \times 10^{-6} \times 100,000 = 3 $$ V

#### 3. Bandwidth of Transimpedance Amplifier

- Op-amp gain-bandwidth product $$ GBW $$, feedback resistor $$ R\_f $$, total capacitance $$ C\_T $$ (parasitic + photodiode)

- Formula for bandwidth: $$ f\_{BW} = \frac{GBW}{2\pi R\_f C\_T} $$

- Example:

- $$ GBW = 1 $$ MHz

- $$ R\_f = 100 $$ kΩ

- $$ C\_T = 20 $$ pF $$ = 20 \times 10^{-12} $$ F

- $$ f\_{BW} = \frac{1\times10^6}{2\pi \times 100,000 \times 20\times10^{-12}} \approx 80 $$ kHz

#### 4. RC Filter Cutoff Calculation

- Attenuate high-frequency noise at output.

- Formula: $$ f\_c = \frac{1}{2\pi RC} $$

- Example:

- $$ R = 10 $$ kΩ, $$ C = 1 $$ nF

- $$ f\_c = \frac{1}{2\pi \times 10,000 \times 1 \times 10^{-9}} \approx 16 $$ kHz

\*\*\*

### Phototransistor Calculations

#### 1. Output Current

- Gain is $$ \beta $$ (typically 100–1000)

- Base photocurrent calculated as per photodiode: $$ I\_{ph} $$

- Collector current: $$ I\_C = \beta \times I\_{ph} $$

- Example:

- $$ I\_{ph} = 0.5 $$ μA, $$ \beta = 200 $$

- $$ I\_C = 0.5 \times 10^{-6} \times 200 = 0.1 $$ mA $$ = 100 $$ μA

#### 2. Output Voltage with Load Resistor

- Resistor $$ R\_L $$ connected at collector to Vcc.

- Output voltage drop: $$ V\_{out} = V\_{cc} - I\_C \times R\_L $$

- Example:

- $$ V\_{cc} = 5 $$ V, $$ R\_L = 10 $$ kΩ

- $$ V\_{out} = 5 - (100 \times 10^{-6} \times 10,000) = 5 - 1 = 4 $$ V

#### 3. Response Time (Rise/Fall Time)

- Determined largely by collector load and capacitances.

- Formula: $$ t\_{r} \approx 0.35 \times \frac{R\_L \times C\_{tot}} $$

- Example:

- $$ R\_L = 10 $$ kΩ, $$ C\_{tot} = 50 $$ pF

- $$ t\_r = 0.35 \times (10,000 \times 50 \times 10^{-12}) = 0.35 \times 0.5 \times 10^{-6} = 0.175 $$ μs

\*\*\*

### Measurement and Threshold Examples

#### 1. Setting ADC Threshold for Line Detection

- Typical circuit: output from TIA is digitized by ADC, threshold chosen to distinguish “on line” vs background.

- If dark current output = 0.2 V, full light output = 2.0 V, choose $$ V\_{threshold} = 1.0 $$ V.

#### 2. Combining Multiple Sensors

- Assume four sensors spaced across a robot.

- To detect line edge, compare outputs; for center line, majority should be above threshold.

#### 3. Noise Suppression

- If observed output signal is noisy, lower RC filter cutoff or add differential pair measurement.

\*\*\*

### Real Circuit Calculation Example

\*\*Given:\*\*

- Photodiode responsivity: 0.5 A/W

- Incident intensity: 100 μW

- Feedback resistor: 220 kΩ

- Parasitic capacitance: 30 pF

- Op-amp GBW: 2 MHz

\*\*Calculations:\*\*

- Photocurrent: $$ I\_{ph} = 0.5 \times 100 \times 10^{-6} = 50 $$ μA

- Output voltage: $$ V\_{out} = 50 \times 10^{-6} \times 220,000 = 11 $$ V (will likely saturate, so scale down R\_f)

- Bandwidth: $$ f\_{BW} = \frac{2 \times 10^6}{2\pi \times 220,000 \times 30 \times 10^{-12}} \approx 48 $$ kHz

Here are comprehensive notes for Question 4, covering your LTSPICE simulations—how to complete a partial schematic, run frequency (AC) and time (TRANSIENT) domain simulations, and make/interpret circuit measurements. These notes give detailed steps, example settings, and troubleshooting methods for typical design test questions[1].

\*\*\*

### LTSPICE Simulation & Measurement Notes

\*\*\*

#### 1. Completing Partial Circuits

- \*\*Identify missing elements:\*\* Resistors, capacitors, sources, or connections as indicated by question or schematic comments.

- Use typical values if not specified (e.g., R = 1 kΩ, C = 0.1 μF).

- Check node labels: Ensure all required connections are made to ground, sources, and measurement points.

\*\*\*

#### 2. Time Domain Simulation (.TRAN)

- \*\*Purpose:\*\* Analyze how voltages and currents in the circuit change over time for given inputs.

- \*\*Typical Setup:\*\*

- Use `.tran tstep tstop` (e.g., `.tran 0.01m 10m` for 10 ms total, 10 μs step).

- SINE or SQUARE source for input:

- `SINE(0 1 1k)` (1 V amplitude, 1 kHz)

- `PULSE(0 5 0 1u 1u 1m 2m)` (5 V, 1 ms width, 2 ms period)

- \*\*How to simulate:\*\*

1. Set or uncomment `.tran` directive.

2. Set input wave (verify type/parameters).

3. Run simulation.

4. Plot input and output nodes.

- \*\*Measurements:\*\*

- Rise/fall time: Use cursors to get time between 10% and 90% of signal.

- Peak voltage: Peak positive/negative voltage from graphs.

- Delay: Compare input/output crossing points for time lag.

- See RC circuit example in your labs for shape of step and sine response[1].

\*\*\*

#### 3. Frequency Domain Simulation (.AC)

- \*\*Purpose:\*\* Examine how your circuit responds at different input frequencies (gain, phase).

- \*\*Typical setup:\*\*

- Use `.ac dec N fstart fstop` (e.g., `.ac dec 100 10 1Meg`)

- AC amplitude set in voltage source: e.g., `AC 1`

- \*\*How to simulate:\*\*

1. Set or uncomment `.ac` directive.

2. Ensure AC amplitude is nonzero in source.

3. Run simulation.

4. Plot output (e.g., `V(out)`), use dB scale for gain (`20\*log10(V(out)/V(in))`).

- \*\*Measurements:\*\*

- -3dB bandwidth: Frequency where amplitude falls to 0.707 of max.

- Phase shift: Difference between input/output phases.

- Resonance, cutoff observation for filters.

- Compare with analytic expectations for $$ f\_c = \frac{1}{2\pi RC} $$[1].

\*\*\*

#### 4. Measurement (.MEAS) Statements

- Place in netlist or schematic text.

- Examples:

- `.meas AC BW WHEN mag(V(out))=0.707`

- `.meas TRAN Vpeak MAX V(out)`

- `.meas TRAN RiseTime TRIG v(out) VAL=0.1 FALL=0 TARG v(out) VAL=0.9 RISE=1`

- Results displayed in text output after simulation ("Draft1.txt" or log).

\*\*\*

#### 5. Parameter Sweeps

- To study effect of R or C variation:

- `.step param R 1k 10k 1k`

- `.param R=5k`

- Will result in multiple traces, showing sensitivity to components.

\*\*\*

#### 6. Typical Troubleshooting

- Ensure all nodes have a ground.

- Use correct units (e.g., 1k, 1u, 1n—not 1000, 0.000001).

- If simulation fails: Check for floating nodes, large time steps, or unrealistic component values.

- For convergence errors, relax tolerances:

- `.options reltol=0.01 abstol=1u vabstol=1u`

\*\*\*

#### 7. Example Test Prompts

- Complete a circuit and simulate response to 1 kHz sine and a step.

- Plot and measure -3dB frequency, time constant, or gain for a given filter.

- Add a missing RC/Opamp stage to a partial schematic as specified.

- Do a parameter sweep to show how output changes with varying R or C.

- Compare simulated and theoretical cutoff frequencies, time constants, or response times.

\*\*\*

#### 8. General Advice

- Always check signal polarity and scale before interpreting plots.

- Use both .ac and .tran simulations to fully characterize circuit.

- Save output data (table or plot) for lab report as required.

- Cross-verify with analytic calculations (RC $$ \tau $$, cutoff, etc.).

\*\*\*

Here is a concise cheatsheet of key formulas for LTSPICE transient (time domain) and frequency domain (AC) simulations, focusing on common circuit measurements relevant to your test and lab tasks[1][2].

\*\*\*

### LTSPICE Simulation Cheatsheet

\*\*\*

#### Transient (Time Domain) Simulation Formulas

- \*\*Time Constant, RC Circuit:\*\*

$$

\tau = R \times C

$$

- \*\*Cutoff Frequency (High / Low Pass Filter):\*\*

$$

f\_c = \frac{1}{2 \pi R C}

$$

- \*\*Rise Time (10% to 90% of step response):\*\*

Approximate for first order system:

$$

t\_r \approx 2.2 \times \tau

$$

- \*\*Settling Time:\*\*

Time taken for output to stay within ±5% of final value (approx.)

$$

t\_s \approx 3 \times \tau

$$

- \*\*Voltage Gain (Voltage Ratio):\*\*

$$

A\_v = \frac{V\_{out}(t)}{V\_{in}(t)}

$$

- \*\*Overshoot:\*\*

$$

\text{Overshoot} = \frac{V\_{peak} - V\_{final}}{V\_{final}} \times 100\%

$$

\*\*\*

#### Frequency Domain (AC) Simulation Formulas

- \*\*Cutoff Frequency:\*\*

Same as above, for −3 dB frequency of filters.

- \*\*Gain in Decibels (dB):\*\*

$$

G\_{dB} = 20 \log\_{10} \left( \frac{V\_{out}}{V\_{in}} \right)

$$

- \*\*Phase Shift:\*\*

Measured as difference in phase angle (degrees or radians) between output and input.

- \*\*Bandwidth (BW):\*\*

Frequency range where gain is within −3 dB of maximum.

- \*\*Resonant Frequency in RLC Circuits:\*\*

$$

f\_0 = \frac{1}{2 \pi \sqrt{L C}}

$$

\*\*\*

#### Measurement Tips for LTspice

- Use `.meas` to automatically find max, min, average, rise time, fall time, frequency, etc.

- Example commands:

```

.meas TRAN Vmax MAX V(out)

.meas TRAN Vmin MIN V(out)

.meas TRAN RiseTime TRIG v(out) VAL=0.1 FALL=1 TARG v(out) VAL=0.9 RISE=1

.meas AC GainAtFreq FIND mag(V(out)) AT=1e3

```

\*\*\*