ME 218 Class File Documentation

Revision 13: 8/11/20

About This Class File:

The `me218` class provided by this package deals with correctly typesetting project definitions for A, B, and C quarters, as well as for typesetting protocol specifications, such as that for the COM in B quarter, or if similar protocol specifications need to be written in future.

Use of appropriate class file options also allows typesetting of lab assignments, as well as assignments and projects for ME 210.

When used in an Overleaf project, this class file, wordmarks, and fonts should be added as linked documents from the Overleaf project for this documentation file. This allows for updates to the graphics or class file to be propagated by refreshing the linked version in the downstream projects.

If a prior project exists, this can be accomplished by making a copy.

About This Documentation:

This documentation is written entirely using the features provided by the `me218` class, except where additional code has been written in this document to properly render examples of title blocks.

Throughout the remainder of this document, code snippets will be provided in fixed-width font:

```
Example code snippet
```

followed by the equivalent render of the result where appropriate.

Commands

Class Options:

This class file is set up to generate both project descriptions and lab reports for both ME210 and ME218, which can be configured using the following options.

`lab`: Passing `lab` as an option when defining the document class will add the due date in the header, and adjust some of the language in the title block to reflect lab assignment organization. `11pt` is the recommended font size.

`project`: Passing `project` as an option when defining the document class will set up the document margins and title block for project specifications. `11pt` is the recommended font size.

`syllabus`: Passing `syllabus` as an option when defining the document class will automatically change the title block and display settings to render the detailed course syllabus. This option and `ataglance` can be used on the same source file to generate detailed or summary views. The only title block commands used should be `\quarter` and `\academicyear`. `10pt` or `11pt` is the recommended font size.

`ataglance`: Passing `ataglance` as an option when defining the document class will automatically change the title block and display settings to render the at-a-glance syllabus. This option and `syllabus` can be used on the same source file to generate detailed or summary views. The only title block commands used should be `\quarter` and `\academicyear`. `10pt` is the recommended font size.

`courseinfo`: Passing `courseinfo` as an option when defining the document class will automatically change the title block to render the course information document. The only title block commands used should be `\quarter` and `\academicyear`. `10pt` is the recommended font size.

`210`: Passing `210` as an option when defining the document class will change the wordmark and "218<a|b|c>" throughout the document to 210.

218: Passing 218 as an option when defining the document class will use the 218 wordmark and text throughout the document.

10pt|11pt|12pt: These class options select the document font size.

Title Block:

The title block uses a combination of macros that set title block element values, and macros that set flags to enable/disable certain display elements. The following title block definition macros are available:

\title{T}: Sets the value of the title to T. This is the portion of the title appearing directly after the colon.

\expandedtitle{E}: Sets the value of the expanded title to E. When set, the expanded title will appear as a second line in bold in the title block, immediately under the primary title. The expanded title will have the font scaled such that it takes up no more than one row. Typical use is to define the expanded acronym in cases where the game title is an acronym.

\headtitle{H}: Sets the value of the header title to H. The header title is the text appearing following the colon in the header on all but page one of the document. The following line of code produced the header in this document:

```
\headtitle{Class File Documentation}
```

\quarter{Q}: Sets the quarter to which the document refers. If Q is a, b, or c, then typesets the appropriate quarter into the first line of the title block and the header. This value is also used to determine which portion of the academic year specification is relevant to the given quarter. Defaults to a.

May be given as \quarter{} to remove quarter, season, and year references from the title and headers.

\academicyear{Y}: Defines the academic year corresponding to the document. The argument Y must be given in the form XX-XX where X is some digit. This class file assumes that the course is being taught in the 21st century.

If quarter was defined as blank, then the value of the argument to academicyear does not matter.

\preview{}: If present, this macro acts as a flag to enable the printing of information about the project preview in the title block. Enabling this option requires the day to be saved in preview, and the start and end times to be stored in previewstart and previewend respectively.

\grading{}: If present, this macro acts as a flag to enable the printing of information about the grading session in the title block. Enabling this option requires the day to be saved in grading, and the start and end times to be stored in gradingstart and gradingend respectively.

\gradingperiod{}: If present, this macro acts as a flag to enable the printing of information about the grading period in the title block. Enabling this option requires the beginning of thr grading period to be saved in preview, the end of thr grading period to be saved in grading, and the end time to be stored in gradingend.

\presentation{}: If present, this macro acts as a flag to enable the printing of information about the public presentation in the title block. Enabling this option requires the day to be saved in presentation, and the start time to be stored in presentationstart.

\revision{}: If present, this macro acts as a flag to enable the printing of information about the document revision history in the title block.

Revisions can be added using the \addrevision and related commands, detailed in the section on revision tracking.

Dates:

The me218 class uses the datetime2 package, along with a couple of customized time formats, to handle typesetting of dates and times. Storing and using dates and times can be accomplished with the following commands:

\DTMsavedate{N}{D}: This saves a date with the name N, and the date D. The name can be an alphanumeric string, and the date must be given in the format YYYY-MM-DD.

\DTMsavetime{N}{T}: This saves a time with the name N, and the time T. The name can be an alphanumeric string, and the time must be given in the format HH:MM:SS.

\DTMusedate{N}: This typesets a date with the name N that was previously stored with DTMsavedate, in the format m/d/YY.

\DTMmonthday{N}: This typesets a date with the name N that was previously stored with DTMsavedate, in the format Month d.

\DTMusetime{N}: This typesets a time with the name N that was previously stored with DTMsavetime, in the format h:mm am.

Revision Tracking:

By defining a series of revisions, the latest revision information will be automatically printed in the title block, and all revision history can be printed at a desired location within the document.

All revisions are zero-indexed.

The following commands are available:

\setrevision{K}{D}{T}: Set revision K to have associated date D and associated text T. Date must be given in the form YYYY-MM-DD.

\addrevision{D}{T}: Set the next available revision index to have associated date D and associated text T. Date must be given in the form YYYY-MM-DD. Next available revision index is calculated by counting uses of addrevision, so this command will overwrite in cases where setrevision was called on an index not yet allocated.

\getrevisiontext{K}: Typesets the text associated with the revision at index K.

\getrevisiondate{K}: Typesets the date associated with the revision at index K, in the format m/d/YY.

\printrevisions: Typesets the text and dates of all defined revisions in order. Only goes through the maximum index defined by the number of uses of addrevision.

Checkpoint Quotes:

The me218 class provides a command to typeset block quotes. The quote will be further indented from the parent text, with the attribution on a newline, inset from the right edge.

\checkpointquote{Q}{A}: Typesets a block quote where Q is the text of the quote, and A is the attribution.

Acronym and Term Definitions:

The me218 class provides a command to streamline definition of acronyms and terms.

\defineTerm[ARTICLE]{MACRO}{RENDER}: Defines a primary pair of macros, \MACRO and \MACROs, which will typeset as RENDER and RENDERs respectively. Additionally defines a pair of macros with attached article, \aMACRO and \AMACRO, which will typeset as ARTICLE MACRO, where the latter has ARTICLE capitalized. ARTICLE is an optional argument defaulting to "a". Also defines \MACRO and \MACROs with capitalized initial letter for non-acronym terms at the beginnings of sentences.

For example, \defineAcronym{test}{foo} will define macros \test, \tests, \Test, \Tests, \atest, and \Atest; typesetting as foo, foos, Foo, Foos, a foo, and A foo, respectively. Each term will automatically add a space after if necessary, and suppress one prior to punctuation.

The optional parameter is intended for use when the acronym begins on a vowel; such an acronym would be defined as, for example, \defineTerm[an]{device}{OBJECT}. This results in macros \device, \devices, \adevice, and \Adevice being defined, which typeset as OBJECT, OBJECTs, an OBJECT, and An OBJECT, respectively.

\defineSingularTerm[ARTICLE]{MACRO}{RENDER}: Provided for the case where the acronym is an irregular plural; defines the same macros as above, but none append an 's' to the rendered output.

For example, \defineSingularTerm{test}{SHEEP} will define macros \test and \tests, but both will typeset as SHEEP.

Environments

The me218 class provides several custom environments for typesetting elements of the 218 specifications, including checklists and byte specifications. The following sections detail the available environments and any associated commands.

General Typesetting:

The indenting environment is used for typesetting most text within an ME218 document.

indenting: The indenting environment indents the text from the parent text, and sets the body font to be that which is used throughout the ME218 documents. Essentially, all non-header text will appear within an indenting environment.

Specification Lists:

The speclist environment is used for typesetting lists of project specifications.

speclist: This environment begins an itemize environment with customized formatting; the bullets are instead check boxes, and the hanging indent is adjusted appropriately.

\item: As in an itemize environment, used at the beginning of a line to mark a new entry. If additional paragraphs are required, adding a double break and additional text without an additional \item will correctly typeset the result including the hanging indent.

Byte Definitions:

There are two environments and associated commands for defining byte protocols and the meaning of bits within those bytes.

bytedefinition[N]: This environment typesets names of bits and bit indices in tabular form. This environment should only contain the commands singleBit, bitGroup, and unusedbits. Stringing these commands in sequence will add bits to the byte, pushing them on as low-order bits. A single use of this environment may be used to define a closely related group of bytes; the optional argument N defines the number of bytes to be typeset.

\suppressAccessRow: When used as the first command in a bytedefinition environment, suppresses typesetting of the access specifier row in the byte definition.

\singleBit[M,V]{N}: This command adds a single bit named N. [M,V] is an optional argument, and will also typeset the access mode and initial value of the bit, as M-V.

\bitGroup[M,V]{N}{K}: This command adds a group of K bits with name N, as NK N(K-1) ... N2 N1 N0. As for singleBit, [M,V] is an optional argument indicating the access mode of the bit group. Currently, the same initial value is set for all bits in the group.

\unusedBits[K][V]: With no optional argument specified, adds a single unused bit, typeset with access mode U-0. With the first optional argument specified, adds K such bits. With both optional arguments specified, typesets K bits with mode U-V.

bitdefinition: This environment typesets information about a single group of related bits within a byte. There should be one of these environments for each distinct group of bits within the byte. This environment should only contain the following commands:

\bitsinbyte{B}: Defines the bit positions within the byte.

B should be a string of the form n-m where n is the index of the highest bit in the byte, and m is the index of the lower bit in the byte.

This command should be called as the first thing in the bitdefinition environment.

\bits[R]{DEF}{DESC}: Defines name, position, and function of a set of named bits within a byte.

DEF is the short definition of the bit collection, as used in the bytedefinition environment. DESC is the English description of the name and/or function of this collection of bits. R is an optional argument; in the case that this bitdefinition refers to multiple bits, this argument should be given as X:Y where X and Y are the range of any numerical suffixes. See the examples below for usage. This argument may be omitted when defining a single bit.

This command should be called once per bitdefinition, immediately following bitsinbyte.

\bitpattern{P}{DESC}: Defines the meaning of a specific bit pattern.

P is the pattern of bits, and DESC is the English description of the meaning of that pattern of bits. This may be called as many times as necessary, defining a specific bit pattern at each call.

\note{T}: For use within a bitdefinition environment, to include a text description of the bit collection. May be used multiple times, although each use should contain only a single paragraph each.

Memory Map Definitions:

There is a provided environment and five associated commands for typesetting memory maps. These may be used in a floating table, or included directly in text; the former is preferred, with a [h] float specifier if desired.

mapdefinition: This environment typesets a series of bytes, formatted as registers in a memory map. Each bit may be inidividually named, and each row is automatically numbered. Assumes 8-bit word size.

\mmByteName{N}: This command should be used at the start of each row, and defined the name of that register in memory. This command should be followed by 8 bits worth of descriptions in the form of mmBit, mmBitGroup, and mmNote commands.

\mmBit{B}: This command adds a single bit named B to the current register.

\mmBitGroup{B}{N}: This command definesa group of bits named B, and numbered from N-1 through 0.

\mmUnusedBits{N}: This command defines a block of N unused bits. These bits will be shaded gray in the memory map.

`\mmNote[N]{T}`: This command defines a written comment with text `T` over a set of `N` bits.

Packet Definitions:

There are two environments and associated commands for defining byte protocols and the meaning of bits within those bytes.

`packetdefinition`: This environment typesets packet formats and annotations for bytes within those packets in tabular form. This environment should only contain the command `byteGroup`. Each use of this command will add bytes to the packet, added them to the end.

`packetbytedefinition`: This environment typesets information about a single group of related bytes within a packet. There should be one of these environments for each distinct group of bytes within the packet. Long packets will automatically be line-broken to fit in the available text area. This environment should only contain `bytesinpacket`, `bytes`, and `note` commands.

`\bytegroup{A}{B0,...,BN}`: This command adds a group of bytes `B0` to `BN` within the packet with annotation `A`. The second argument is a comma-separated list of byte names; this is rendered in text mode, so use `\textsubscript{}` for subscripts, or math mode to enclose symbols. See the examples for usage details.

`\bytesinpacket{R}`: This command should be the first item in a `packetbytedefinition` environment. `R` is the range of byte indices, and is printed with no further processing.

`\bytes{A}`: Used to set a short description of the meaning of these bytes. Should be used once per `packetbytedefinition`, immediately following `bytesinpacket`.

`\note{T}`: For use within a `packetbytedefinition` environment, to include a text description of the byte collection. May be used multiple times, although each use should contain only a single paragraph each.

Examples

Acronym and Term Definitions:

Acronym and term definitions are accomplished with the `defineTerm` macro. The following lines of code:

```
\defineTerm{robot}{ROBOT}
One \robot, ah, ha, ha! Two \robots, ah, ha, ha!
```

will typeset as:

One ROBOT, ah, ha, ha! Two ROBOTs, ah, ha, ha!

While the following lines of code:

```
\defineTerm[an]{device}{OBJECT}
This is \adevice, one of many \devices.
```

will typeset as:

This is an OBJECT, one of many OBJECTs.

For non-acronym terms, the following:

```
\defineTerm[an]{field}{end zone}
\Fields everywhere, with \afield here and many \fields there.
```

will typeset as

End zones everywhere, with an end zone here and many end zones there.

Title Block:

Typical example of a title block for an ME218 project:

```
\title{SPACE JAM}
\expandedtitle{Superior Planetary Ambassador's Competitive Evaluation of Jus-
tifiably Astronomical Magnitude}
\headtitle{SPACE JAM}
\quarter{b}
\academicyear{17-18}
\preview{}
\grading{}
\presentation{}
```

Resulting in the following typeset title block:

ME 218b SPACE JAM

Superior Planetary Ambassador's Competitive Evaluation of Justifiably Astronomical Magnitude

Project Preview on March 3 from 1-5 pm. **Grading Session** on March 5 from 1-5 pm.
Project Presentation on March 7 starting at 7:00 pm.

where the `expandedtitle` can be left undefined if the name of the project is not an acronym. Any of the `grading`, `preview`, and `presentation` may be left undefined to suppress typesetting of those specific elements.

The title block for this document uses the following code:

```
\title{Class File Documentation}
\headtitle{Class File Documentation}
\quarter{}
\revision{}
```

Resulting in the following typeset title block:

ME 218 Class File Documentation

Revision 13: 8/11/20

Specification Lists:

Specification list environments should be included directly under the subsection, and not within an `indenting` environment.

The following code:

```
\begin{speclist}
\item The first specification.
\item The second specification.

      With two paragraphs.
\end{speclist}
```

results in the following specification list:

- ☐ The first specification.
- ☐ The second specification.
  With two paragraphs.

Byte Definitions:

Byte definitions should be used within the `indenting` environment of the parent section. Typically, the byte itself will be defined by a `subsubsection*`, as in the following code defining a status byte:

```
\subsubsection*{SS}
Status Information Byte

\begin{bytedefinition}
\suppressAccessRow
\unusedBits[2]
\bitGroup[R,0]{POS}{2}
\unusedBits
\bitGroup[R,0]{GS}{3}
\end{bytedefinition}
```

This should be immediately followed by, in this case, two `bitdefinitions`, one for each group of bits. Code for the first of these is as follows:

```
\begin{bitdefinition}
\bitsinbyte{2-0}
\bits[2:0]{GS}{Game status bits}
\bitpattern{000}{waiting for start}
\bitpattern{001}{face-off}
\bitpattern{011}{playing}
\bitpattern{100}{sudden death}
\bitpattern{101}{game over}
\end{bitdefinition}
```

This results in the following complete byte definition for this status byte:

SS: Status Information Byte

—	—	POS1	POS0	—	GS2	GS1	GS0
bit 7							bit 0

bit 2-0 **GS<2:0>:** Game status bits
`000` = waiting for start
`001` = face-off
`011` = playing
`100` = sudden death
`101` = game over

bit 5-4 **POS<1:0>:** Possession status bits
`00` = Neither team in possession
`01` = RED in possession
`10` = BLUE in possession

Multi-byte definitions can be created using code like the following example, which also uses a `note` in the `bitdefinition`:

```
\begin{bytedefinition}[2]
\singleBit[R,1]{STATUS}
\unusedBits[2][x]
\singleBit[R,1]{RAND}
\unusedBits
\bitGroup[R/W,0]{POS}{3}
\singleBit[W,0]{CFLAG}
\singleBit{MODE}
\unusedBits[5]
\singleBit[W,x]{LAST}
\end{bytedefinition}


\begin{bitdefinition}
\bitsinbyte{7}
\bits{CFLAG}{Clear flag}
\note{Write a \texttt{1} to this bit to clear the flag.}
\end{bitdefinition}
```

Which results in the following typeset byte definition:

DBYTE: Demo Byte

R-1	U-x	U-x	R-1	U-0	R/W-0	R/W-0	R/W-0
STATUS	—	—	RAND	—	POS2	POS1	POS0
bit 15							bit 8

W-0		U-0	U-0	U-0	U-0	U-0	W-x
CFLAG	MODE	—	—	—	—	—	LAST
bit 7							bit 0

bit 7 **CFLAG:** Clear flag
Write a `1` to this bit to clear the flag.

Packet Definitions:

Packet definitions should occur within the `indenting` environment of the parent section.[1] Typically, the packet itself will be defined by a `subsubsection*`, as in the following code defining a packet:

```
\subsubsection*{802.15.4 Frame}
\begin{packetdefinition}
\byteGroup{Start Delimiter}{0x7E}
\byteGroup{Length}{MSB,LSB}
\byteGroup{Frame Data}{API-specific structure}
\byteGroup{Checksum}{CHK}
\end{packetdefinition}
```

[1] Use of an `MEsubsection` environment includes the `indenting` environment.

This should be immediately followed by, in this case, three `packetbytedefinitions`, one for each group of bytes requiring explanation. In this case, we leave out an explanation for the frame delimiter. Code for the frame data is as follows:

```
\begin{packetbytedefinition}
\bytesinpacket{4-n}
\bytes{Payload}
\note{Contains data associated with this packet}
\end{packetbytedefinition}
```

This results in the following complete packet definition:

802.15.4 Frame:

Start Delimiter	Length		Frame Data	Checksum
0x7E	MSB	LSB	API-specific structure	CHK

byte 2-3 — Packet length: MSB,LSB
Packet length is number of bytes in packet, not including checksum

byte 4-n — Payload
Contains data associated with this packet

byte n+1 — Checksum
Checksum calculated using all preceding bytes

The following definition of a longer packet demonstrates the automatic line-breaking for long packets:

```
\subsubsection*{Dynamixel Protocol 2.0 Message Format}
\begin{packetdefinition}
\byteGroup{Header}{0xFF,0xFF,0xFD}
\byteGroup{Reserved}{0x00}
\byteGroup{Device ID}{ID}
\byteGroup{Length}{LEN\textsubscript{L},LEN\textsubscript{H}}
\byteGroup{Instruction}{CMD}
\byteGroup{Instruction Address}{ADDR\textsubscript{L},ADDR\textsubscript{H}}
\byteGroup{Data}{D\textsubscript{1},$\cdots$,D\textsubscript{N}}
\byteGroup{CRC}{CRC\textsubscript{L},CRC\textsubscript{H}}
\byteGroup{CRC}{CRC\textsubscript{L},CRC\textsubscript{H}}
\byteGroup{CRC}{CRC\textsubscript{L},CRC\textsubscript{H}}
\end{packetdefinition}
```

Which typesets as:

Dynamixel Protocol 2.0 Message Format:

Header			Reserved	Device ID	Length		Instruction	Instruction Address	
0xFF	0xFF	0xFD	0x00	ID	LEN_L	LEN_H	CMD	$ADDR_L$	$ADDR_H$

Data			CRC		CRC		CRC	
D_1	$\cdots$	D_N	CRC_L	CRC_H	CRC_L	CRC_H	CRC_L	CRC_H

Memory Maps:

A memory map can be defined using a `mapdefinition` environment, with the following code:

```
\begin{mapdefinition}
  \mmByteName{STATUS} \mmUnusedBits{6} \mmBitGroup{MOS}{2}
  \mmByteName{C1MLOC1} \mmBit{LG} \mmBit{WFLG} \mmBit{BFLG} \mmBit{UFLG} \mmBitGroup{LOC}{4
  \mmByteName{C1MLOC2} \mmBit{LG} \mmBit{WFLG} \mmBit{BFLG} \mmBit{UFLG} \mmBitGroup{LOC}{4
  \mmByteName{C2MLOC1} \mmBit{LG} \mmBit{WFLG} \mmBit{BFLG} \mmBit{UFLG} \mmBitGroup{LOC}{4
  \mmByteName{C2MLOC2} \mmBit{LG} \mmBit{WFLG} \mmBit{BFLG} \mmBit{UFLG} \mmBitGroup{LOC}{4
  \mmByteName{C1RESH} \mmNote{Corporation 1 Resources High Byte}
  \mmByteName{C1RESL} \mmNote{Corporation 2 Resources Low Byte}
  \mmByteName{C2RESH} \mmNote{Corporation 1 Resources High Byte}
  \mmByteName{C2RESL} \mmNote{Corporation 2 Resources Low Byte}
  \mmByteName{PUR1} \mmBitGroup{C2PMT}{4} \mmBitGroup{C1PMT}{4}
  \mmByteName{PUR2} \mmBitGroup{N2PMT}{4} \mmBitGroup{N1PMT}{4}
\end{mapdefinition}
```

Which results in the following memory map:

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00h	STATUS	—	—	—	—	—	—	MOS1	MOS0
01h	C1MLOC1	LG	WFLG	BFLG	UFLG	LOC3	LOC2	LOC1	LOC0
02h	C1MLOC2	LG	WFLG	BFLG	UFLG	LOC3	LOC2	LOC1	LOC0
03h	C2MLOC1	LG	WFLG	BFLG	UFLG	LOC3	LOC2	LOC1	LOC0
04h	C2MLOC2	LG	WFLG	BFLG	UFLG	LOC3	LOC2	LOC1	LOC0
05h	C1RESH	Corporation 1 Resources High Byte							
06h	C1RESL	Corporation 2 Resources Low Byte							
07h	C2RESH	Corporation 1 Resources High Byte							
08h	C2RESL	Corporation 2 Resources Low Byte							
09h	PUR1	C2PMT3	C2PMT2	C2PMT1	C2PMT0	C1PMT3	C1PMT2	C1PMT1	C1PMT0
0Ah	PUR2	N2PMT3	N2PMT2	N2PMT1	N2PMT0	N1PMT3	N1PMT2	N1PMT1	N1PMT0

The `mapdefinition` environment may also be placed in a floating table as follows:

```
\begin{table}[tb]
  \begin{mapdefinition}
    .
    .
    .
  \end{mapdefinition}
  \caption{Caption}
  \label{tab:mapDefinitionExample}
\end{table}
```

Currently Undocumented

The following content is examples for typesetting lab assignments which are not yet fully documented.

Part 0: Investigations

Reading:

COK 9.9.2, Scherz Section 2.3, Tutorial: Simulating the design.

Background:

Each lab station is equipped with a small circuit board with an RC circuit (low pass filter) similar to that shown in Figure 1.

Assignment:

☐ **0.1)** The value of the resistor can be determined from the color bands. Your task is to determine the value of the capacitor. We want you to use the principles of an RC filter to do this determination.

a) Apply a square wave to the input of the RC and measure the 63% rise time[2]. Using that information, you can determine what the value of C is. You can also determine the corner frequency (or -3 dB point) of this filter. Practice using the cursors and frequency measurement functions on the digital scope to do this.

b) Apply a sine wave with a frequency 10 times the corner frequency, a frequency equal to the corner frequency, and a frequency $1/10$ the corner frequency and determine the ratio of V_{in} to V_{out}. As you fill out this table, think about what you would expect the ratio of V_{in} to V_{out} to be at the corner frequency and well above and well below the corner frequency. If your data doesn't match your expectations, figure out what is wrong **before you turn in the lab report**.

☐ **0.2)** When you are finished with the "mystery" circuit board, please replace it on the shelf under the monitor so that the next person can easily find it. Also refer to Task 0.1

In the report:

Show the value of C in the RC filter and all of your calculations. List the amplitude ratios determined in 0.1.b. **List the number on the back of the circuit that you used** (without this we can not grade this section).

Board #: _____

	Frequency	V_{in}	V_{out}	V_{out} / V_{in}
$1/10\ F_c$				
F_c				
$10\ F_c$				

Part 1: Dummy Section for Count

Section text

Time Spent:

Preparing **outside** of the lab: _____

In the lab working on this part: _____

Part 2: Dummy Section for Count

Section text

Part 1 Preparing **outside** of the lab: _____

In the lab working on this part: _____

Part 2 Preparing **outside** of the lab: _____

In the lab working on this part: _____

Report Preparing **the report**: _____

[2] Hint: see online O'scope tutorial

Test Section:

Here's the text

- ☐ **2.1)** item one

Figure 1: The caption

- ☐ **2.2)** item two see fig. 1.

Appendix A: The test appendix

test:

some text

Revision History

Revision 0: Start writing documentation. (2/10/18)
Revision 1: Finish documentation, add command to streamline acronym definitions. (3/10/18)
Revision 2: Adjust bit definition spacing for multiple-byte entries. Minor grammar corrections. (9/18/18)
Revision 3: Add capitalized versions of term macros for beginnings of sentences (9/18/18)
Revision 4: Adjust header to use PostScript wordmarks. (9/20/18)
Revision 5: Add functions for typesetting of packets. (11/26/18)
Revision 6: Rework functions for typesetting of packets. (11/27/18)
Revision 7: Fix for extra blank line in title block when not a lab assignment. (1/25/19)
Revision 8: Add option of typesetting for ME 210. (1/7/20)
Revision 9: Update display of revisions at bottom of file. (2/18/20)
Revision 10: Add code for memory maps, adjusted array rules and glue for better PDF output. (2/20/20)
Revision 11: Add documentation for memory maps and packets, adjusted spacing of packets. (2/27/20)
Revision 12: Add code to implement Syllabus/At a Glance, documentation pending. Added partial header logic for Course Info sheet, table definitions and documentation pending. (7/26/20)
Revision 13: Minor update to title block when generating course info sheet. Table generation code for course info sheet has been moved into the class file. Updated the Class Options documentation. (8/11/20)