User Manual

Stm32f4-discovery 320x240 VGA

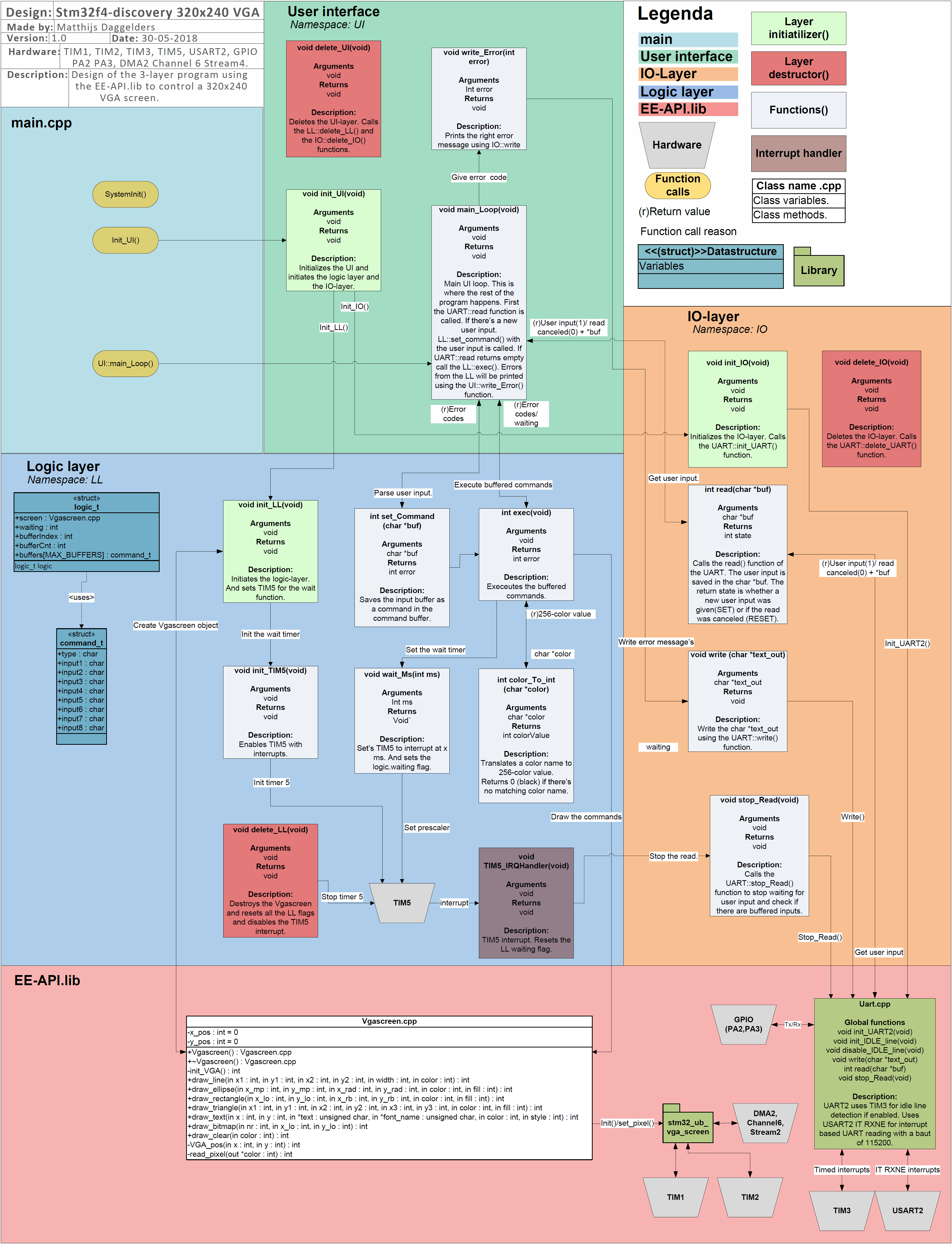
# Inleiding(beschrijving)

De stm32f4-discovery 320x240 VGA is een C/C++ project dat draait op de stm32f4-discovery en stuurt een 320x240 VGA scherm aan. Door verschillende input aan de stm32f4-discovery board via UART te geven. Worden verschillende dingen op het scherm getekend. Zo kan er tekst in verschillende stijlen, lijnen, ellipsen en andere vormen op het VGA scherm getekend worden. Dit kan per input gedaan worden maar er kan ook een loop gecreëerd worden. Een loop kan gemaakt worden door eerst een wait opdracht te sturen en vervolgens de opdrachten die herhaald moeten worden afsluiten met een *“repeat” opdracht.* In de loop worden de input achterelkaar uitgevoerd en op het einde wordt er weer bij het begin begonnen. Om de loop vervolgens te verlaten kan elke willekeurige opdracht gestuurd worden.

# Inhoudsopgave

[Inleiding(beschrijving) 4](#_Toc515458555)

# Ontwerp 3-lagen

Hieronder is het 3-lagen model van de software weergegeven.  


Figuur . Ontwerp 3-lagen model met de EE-API.lib.

## Toelichting 3-lagen model

In figuur 1 is het ontwerp van het 3-lagen model met de EE-API.lib weergegeven. Het onderste licht rode vlak is de EE-API.lib. Daarboven zijn de 3-lagen van het programma en de main.cpp weergegeven. De drie lagen van het model zijn de:

* **User interface (UI)**De user interface bepaald wat de gebruiker te zien krijgt en is de basis van het programma. De UI heeft een *main\_Loop()* functie waarin de rest van het programma werkt. Eerst wordt de input van de gebruiker opgevraagd via de IO-layer *(IO::read()).* Als er een input binnen is gekomen wordt deze door gestuurd naar de LL. De LL verwerkt de input van de gebruiker en geeft een error code terug als de input niet herkent wordt of als de input buffer vol zit. De *write\_Error()* functie vertaald de error code naar een output string en stuurt deze door naar de IO-layer. De IO-layer stuurt het op zijn beurd door naar de UART, die de error melding op de terminal weergeeft.

Het is ook mogelijk dat de *IO::stop\_Read()* wordt aangeroepen. Wanneer dit gebeurd stopt het programma met wachten op gebruiker input en wordt in de UI de *LL::exec()* aangeroepen om te kijken of er gebufferde opdrachten( commands) uitgevoerd kunnen worden.

* **Input/output-layer (IO-layer)**

De IO-layer functioneert als de laag tussen het programma en de UART library functionaliteit. Als de LL-layer of de UI iets met de UART willen doen gaat dit door de IO-layer heen. Doordat alles door de IO-layer heen gaat. Hoeft alleen deze layer aangepast te worden als een andere input bron gebruikt wordt (bijvoorbeeld I2C). De UI en de LL kunnen die IO-layer op de zelfde functies aanspreken alleen wat erna gebeurd veranderd. Hierbij moet rekening gehouden worden met het programma verloop.

Als de *UART::read()* wordt aangeroepen blijft het programma net zolang wachten tot er een user input gegeven is of de *UART::stop\_Read()* wordt aangeroepen. Deze functies worden aangeroepen met de zelfde functie naam alleen dan in de IO namespace.

* **Logic-Layer (LL)**

De LL is waar de user input wordt opgesplitst in losse woorden. Dit gebeurd in de *set\_command()* functie. De opdracht wordt opgeslagen in *logic.buffers[].* Met de LL::exec() functie kan de huidige command of alle gebufferde commands uitgevoerd worden.

De LL::exec() functie bepaald welk VGAscreen functie aangeroepen moet worden. En vertaald de command naar de juiste format voor de VGAscreen functie.

De wait\_Ms() functie van de LL zet de waiting timer hoog en zet de prescaler van de timer zo in dat het gewenste aantal ms gewacht wordt. Tijdens het wachten kunnen er wel commands in de buffer worden opgeslagen maar de exec() command voert niks uit en keert terug naar de UI om vervolgens weer te wachten op de gebruiker input. De timer 5 interrupt geeft roept de IO::stop\_Read() functie op. Door het stoppen van de IO::read() zorgt ervoor dat de LL::exec() functie wordt aangeroepen in de UI::main\_Loop(). De LL::exec() voert alle commands uit de buffer uit totdat er een nieuw wait command gegeven is. Als het laatste command in de buffer “repeat” is dan worden alle commands die opgeslagen zijn herhaald. Dit gebeurd net zolang tot er een nieuwe gebruiksinput gegeven is.

De logic-layer staat als een mediator tussen het programma en de VGA functies. Hierdoor als er iets veranderd in de VGAscreen klasse hoeft alleen de LL aangepast te worden. Zelfde geld als bijvoorbeeld de UI veranderd zal de LL het zelfde blijven. Dit komt doordat de nieuwe UI zonder problemen de LL functies aan kan spreken, zonder wijzigingen aan de LL kant.

## EE-API.lib

# Doxygen

## Doxygen 3-lagen model

## Doxygen EE-API.LIB