

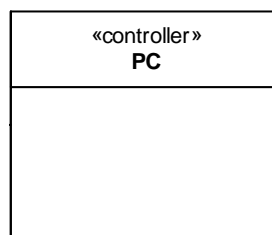
1 Softwaredesign

1.1 Version

Dato	Version	Initialer	Ændring
14. november	1	KT	Første udkast af dokumentet

1.2 PC blokken

Pc controller

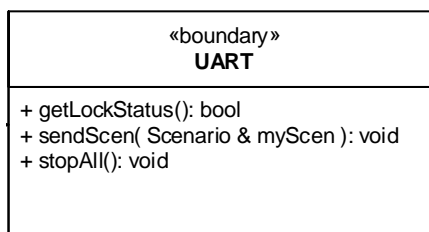


PC controlleren virker som et main program, og holder styr på følgende:

- Kodelås
- Data til UART klassen
- Led mellem Scenario og UI klassen

(Se sekvens diagram UC1 PC for yderligere information)

1.2.1 UART



Constructor

Operation: UART
Parametre: -
Returværdi: -
Beskrivelse: Constructoren opretter en forbindelse til UART'en på STK500 kittet ved brug af serialforbindelse. Til forbindelsen bruges Boost-biblioteket, og underbiblioteket 'asio'. <http://stackoverflow.com/questions/9917307/send-data-via-serial-port-in-cpp-using-com-ports> er et eksempel på opstart af IO port via USB.

getLockStatus

Operation: `bool getLockStatus(void)`
Parametre: -
Returværdi: bool: returnere `true` / `false` afhængeligt af kodelåsens status.
Beskrivelse: Sender et L over til microcontrolleren for at få kodelåsens status. funktionen venter på at få et svar tilbage fra microcontrolleren, hvilket enten er et U eller et L. Hvis UART'en får et 'L' returneres `false`. Hvis UART'en får et U returneres `True`.

stopAll

Operation: `void stopAll(void)`
Parametre: -
Returværdi: -
Beskrivelse: Funktionen bruges til at slukke for alle enheder tilsluttet systemet. et S sendes for til microcontrolleren. Se ref. protokol.

SendScen

Operation: `void SendScen(Scenario & Scenario)`

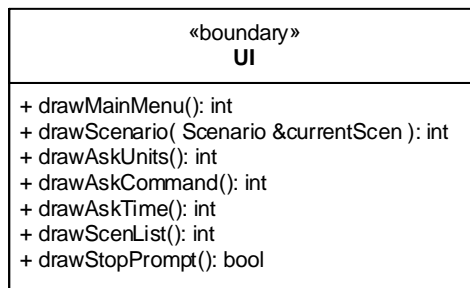
Parametre: Scenario & Scenario: reference til scenariet.

Returværdi: -

Beskrivelse: UARTen transmittere data over serial kommunikation. rækkefølgen på data der bliver sendt er følgende: Unit (1 char) CMD(1char) Time (2char). data-rækken bliver sendt 20 gange for at få alle kommandoer i scenariet overført til microcontroller.

NOTE: Tiden omreges iforhold til nuværende tid på systemet i computeren. f.eks. hvis klokken er 15.00 på computeren og første aktion sker kl 16.30 bliver tiden sat som 90.

1.2.2 UI



drawMainMenu

Operation: `int drawMainMenu(void)`
Parametre: -
Returværdi: `int` : Den returnerede integer er hvilken menu der er valgt.
Beskrivelse: Metoden udskriver en menu over de forskellige undermenuer på skærmen og giver brugeren muligheden for at vælge at gå ind i en af undermenuerne (opret nyt scenarie(1), kørs eksisterende scenarie(2), stop scenarie(3)). Den valgte menu's ID returneres.

drawScenario

Operation: `int drawScenario(Scenario & currentScen)`
Parametre: `Scenario & currentScen` : reference til klassen Scenario
Returværdi: `int` : Den returnerede værdi er den valgte action i Scenariet
Beskrivelse: Metoden udskriver en liste over actionerne i scenariet. Brugeren kan derefter vælge mellem de 20 aktioner, og ID'en på den valgte aktion returneres.

drawAskUnits

Operation: `int drawAskUnits(void)`
Parametre: -
Returværdi: `int` : Den returnerede integer er hvilken unit der valgt
Beskrivelse: Metoden udskriver en liste over units. Brugeren kan derefter vælge hvilken unit der skal bruges, og ID'en på denne unit returneres.

drawAskCommando

Operation: `int drawAskCommand(void)`
Parametre: -
Returværdi: `int` : Den returnerede integer er hvilken kommando
Beskrivelse: Metoden udskriver en liste over kommandoer. Brugeren kan derefter vælge hvilken kommando der skal bruges, og ID'en på denne kommando returneres.

drawAskTime

Operation: `int drawAskTime(void)`
Parametre: -
Returværdi: `int` : tiden i minutter indtastet af brugeren siden kl 00.00
Beskrivelse: Brugeren bliver bedt om at indtaste tiden for hvornår en kommando skal udføres. tiden udregnes til antal minutter siden kl 00.00 og returneres.

drawScenList

Operation: `int drawScenList(void)`
Parametre: -
Returværdi: `int` : ID'et på det valgte scenarie
Beskrivelse: Brugeren bliver præsenteret for 3 forudstillede scenarier, med beskrivelser af hvad de gør. Brugeren vælger en af de 3 forudindstillet scenarier.

drawStopPromt

Operation: `bool drawStopPromt(void)`
Parametre: -
Returværdi: `bool`: returnere `true` for valgt 'ja' og `false` for valgt nej
Beskrivelse: Brugeren bliver præsenteret med muligheden om brugeren er sikker på at scenariet skal afsluttes med valgmulighederne: 'ja' og 'nej'.

1.2.3 Scenario

Scenario
+ <code>getAction(int actionNo)</code> : <code>Action &</code> + <code>SortActions(* Action[])</code> : <code>void</code>

Constructor

Operation: `void Scenario()`
Parametre: -
Returværdi: -
Beskrivelse: constructoren opretter et array med 20 objekter af klassen `action` i.

`getAction`

Operation: `Action & getAction(int actionNo)`
Parametre: `actionNo` - ID'et på den `action` der skal bruges
Returværdi: `Action &` - Reference til den valgte `action`.
Beskrivelse: Metoden finder den valgte `Action` ud fra parameteren `actionNo`, og returnere en reference til den `action`.

`sortActions`

Operation: `Action[] & sortActions(Action[])`
Parametre: Modtager en pointer til et array af `Action` objekter.
Returværdi: Returnerer en reference til det sorterede array af `Action` objekter.
Beskrivelse: Sorterer et array af `Action` objekter efter udførelsestidspunkt. Den der skal udføres først lægges i starten af arrayet. Alt skal være relativt til det nuværende tidspunkt.

operator<<

- Operation: `ostream & operator<<(ostream & theStream, Scenario & theScen)`
- Parametre: Modtager det `ostream` objekt, som der skal streames til samt hvilket `Scenario` objekt, der skal streames.
- Returværdi: Returnerer en reference til det `ostream` objekt, der modtages som parameter. (tillader cascading).
- Beskrivelse: Skal udskrive "Aktionsnummer Tidspunkt Enhed Kommando" som titler med et bestemt mellemrum på én linie og herefter udskrive nummeret på den første aktion, den første aktion på den næste linie. Herefter udskrives nummeret på den anden aktion samt den anden aktion på den næste linie osv. Se evt. Tabel ?? på side ?? i kravspecifikationen.

1.2.4 Action

Action
+ setUnit(int unitNo): bool + setCommand(int commandNo): void + setTime(int time): void
- seconds: int - houseCode: int - unitCode: int - commandNo: int

Figur 1: Action klassen

setCommand

Operation: `bool setCommand(int commandNo)`

Parametre: Modtager nummeret på den kommando der skal eksekveres.

1 = Tænd

2 = Sluk

3 = Dim 5%

4 = Dim 15%

...

12 = Dim 95%

Returværdi: Returnerer **TRUE**, hvis input var gyldigt og **FALSE** hvis ikke.

Beskrivelse: Set-metode til at vælge hvilken kommando, den givne aktion skal indeholde. Metoden ændrer udelukkende på variablerne `commandNo`.

setUnit

Operation: `bool setUnit(int unitNo)`

Parametre: Modtager hvilken enhed aktionen skal manipulere.

1 = Lampe 1

2 = Lampe 2

3 = TV

4 = Radio

Alle andre værdier er ugyldige.

Returværdi: Returnerer **TRUE**, hvis input var gyldigt og **FALSE** hvis ikke.

Beskrivelse: Set-metode til at vælge hvilken enhed, den givne aktion skal manipulere. Metoden ændrer udelukkende på variablerne `houseCode` og `unitCode`. `houseCode` skal iøvrigt sættes i forhold til den enhed, som vælges. Lampe 1 og Lampe 2 er under huskode 1, TV og Radio er under hhv. 2 og 3.

setTime

- Operation: `bool setTime(int time)`
- Parametre: Modtager tidspunktet for hvornår en aktion skal udføres i minutter fra kl 00:00. Dvs hvis en aktion skal starte kl 15:30 skal værdien $15 \times 60 + 30 = 930$ indsættes.
- Returværdi: Returnerer `TRUE`, hvis input var gyldigt og `FALSE` hvis ikke.
- Beskrivelse: Set-metode til at vælge hvilket tidspunkt, den givne aktion skal udføres. Metoden ændrer udelukkende på variabelen `minutes`.

operator<<

- Operation: `ostream & operator<<(ostream & theStream, Action & theAction)`
- Parametre: Modtager et `ostream` objekt, som der skal streames til samt en reference til et objekt af klassen `Action`, som skal udskrives.
- Returværdi: Returnerer en reference til det `ostream` objekt, som metoden blev kaldt med (tillader cascading).
- Beskrivelse: Udskriver en linie med tidspunkt i formattet `HH:MM`, enhedsnavn og kommando beskrevet i tekstform, afsluttet med et lineskift.

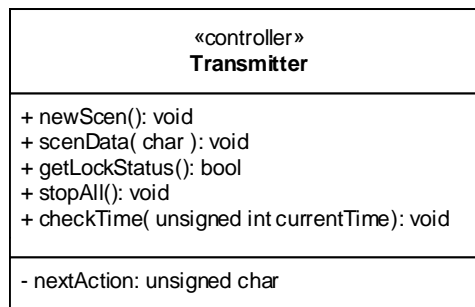
Explicit constructor

- Operation: `Action(int time, int unit, int commandNo)`
- Parametre: Modtager parametre for hhv. tid, enhed og kommandonummer.
- Beskrivelse: Initierer attributter i objektet, skal sætte attributterne til passende default-værdier, hvis ingen værdier gives.

Attribut	Beskrivelse
<code>int seconds</code>	Tiden fra kl 00:00 til tiden for at den pågældende aktion skal udføres i sekunder.
<code>int houseCode</code>	Huskoden for den enhed, som aktionen skal manipulere.
<code>int unitCode</code>	Enhedskoden for den enhed, som aktionen skal manipulere.
<code>int commandNo</code>	Nummeret på den kommando, som skal eksekveres.

1.3 Transmitter blokken

1.3.1 Transmitter



newScen

Operation: void newScen(void)
Parametre: -
Returværdi: -
Beskrivelse: Stopper det igangværende scenarie og sætter nextAction til 0.

scenData

Operation: void ScenData(char)
Parametre: char: Indeholder enten information om enhed, kommando eller tidpunkt.
Returværdi: -
Beskrivelse: Metoden har ansvaret for at indsætte char'en i det pågældende aktionsobjekt (se protokol om UART).

getLockStatus

Operation: bool getLockStatus(void)
Parametre: -
Returværdi: bool : Returværdien fortæller om kodelåsen er indstilt korrekt.
Beskrivelse: Metoden har ansvaret for status angående kodelåsen.

`stopAll`

Operation: `void stopAll(void)`

Parametre: —

Returværdi: —

Beskrivelse: Metoden har ansvaret for at stoppe det igangværende scenarie ved at sende stop-kommandoen ud på Tx10.

`checkTime`

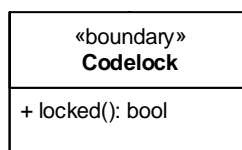
Operation: `void checkTime(unsigned int)`

Parametre: `unsigned int` : Nuværende tid i sekunder fra Time-klassen.

Returværdi: —

Beskrivelse: Metoden bliver kaldt fra TimeKlassen hver 5. sekund, og ansvaret for om den næste aktion skal afvikles. Dette sker kun hvis den givende tid er større end den næste aktion's tid.

1.3.2 Codelock



unlocked

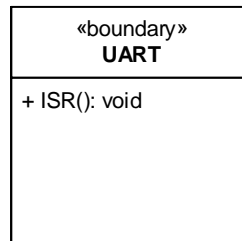
Operation: bool unlocked(void)

Parametre: ingen parametre

Returværdi: TRUE hvis den er åben. FALSE hvis den ikke er åben.

Beskrivelse: Tjekker om koden er tastet rigtigt ind på vores DE2 board.

1.3.3 Transmitter UART



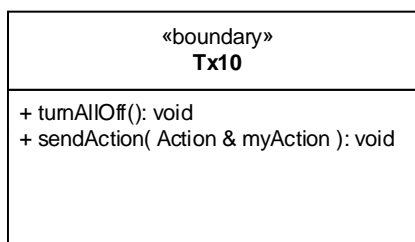
constructor

Operation: UART
Parametre: -
Returværdi: -
Beskrivelse: opsætter forbindelse på RX og TX til UART, sætter op til brug af interrupt.

ISR

Operation: ISR()
Parametre: -
Returværdi: -
Beskrivelse: Ved interrupt på UARTen går transmitteren i 'read mode'. Hvis UART'en læser et L som første char, tjekkes Lockstatus på kodelåsen og returneres til PCen. Hvis et N læses går UARTEN i 'receiving mode' og begynder at aflæse data på reciever pinen. se ref. (state-machine transmitter UART), for lagring af modtagne data. Hvis S modtages sættes transmitteren til 'stop mode'.

1.3.4 Tx10



turnAllOff

Operation: `void turnAllOff()`

Parametre: -

Returværdi: -

Beskrivelse: Når denne metode kaldes, sendes slukke-kommandoer ud til samtlige enheder på netværket via `sendAction()`.

sendAction

Operation: `void sendAction(Action & myAction)`

Parametre: Modtager en reference til det pågældende objekt af klassen `Action`, som skal eksekveres.

Returværdi: -

Beskrivelse: Skal ved hjælp af protokollen for kommunikation over X.10 afsende kommandoer. Der skal sendes én bit hver anden interrupt/nulgennemgang på PD2 (INT0) (interrupt i toggle mode). Ved hver bit skal der ved det efterfølgende interrupt sendes det modsatte af det foregående. Hver gang et 1-tal skal sendes, skal dette være HIGH i 1 ms fra nulgennemgangen/interruptet.

Constructor

Operation: `Tx10(void)`

Parametre: Ingen

Beskrivelse: Skal initiere `Timer0` samt interrupt på PD2 (INT0), men dog ikke aktivere disse.

1.3.5 Time

Time
+ startTime(): void + stopTime(): void
- second_: unsigned int

Time

Operation: Void Time()
Parametre: -
Returværdi: -
Beskrivelse: -initialisere timer 1 til at interrupt hvert sekund.

startTime

Operation: Void startTime (void)
Parametre: -
Returværdi: -
Beskrivelse: -starte tidstælling på timeren. Tiden tælles op hvert minut.

stopTime

Operation: Void stopTime (void)
Parametre: -
Returværdi: -
Beskrivelse: -stopper tidstælling på timeren.

operator++Operation: `operator++()`

Parametre: -

Returværdi: -

Beskrivelse: -inkremere attributten second med 1.

Attribut	Beskrivelse
unsigned int second	indeholder tiden i sekunder.

1.3.6 Action

Action
+ setAction(unsigned char unit, unsigned char command, unsigned int time): void
- time: unsigned int - unit: unsigned char - command: unsigned char - houseCode: unsigned char

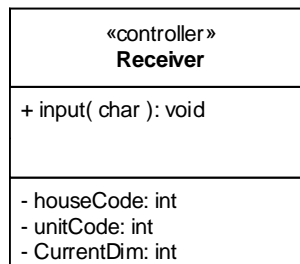
setAction

- Operation: `void setAction(unsigned char unit, unsigned char command, unsigned int time)`
- Parametre: Modtager enhed, kommando og tidspunkt relativt til starttidspunktet for aktionen.
- Returværdi: -
- Beskrivelse: Metodens formål er primært at gemme informationer i en aktion. Metoden skal selv sætte houseCode baseret på hvilken enhed, den kaldes med. De forskellige houseCode værdier kan ses nedenfor:
Lamper: 1
TV: 2
Radio: 3

Attribut	Beskrivelse
<code>unsigned int time</code>	Gemmer tiden i minutter for den pågældende aktion relativt til det tidspunkt scenariet sættes igang.
<code>unsigned char unit</code>	Gemmer hvilken enhed, aktionen skal gælde for.
<code>unsigned char command</code>	Gemmer hvilken kommando, aktionen skal sende.
<code>unsigned char houseCode</code>	Gemmer huskoden, for den enhed der skal manipuleres.

1.4 Receiver blokken

1.4.1 Receiver



Receiver

Operation: void input(char)

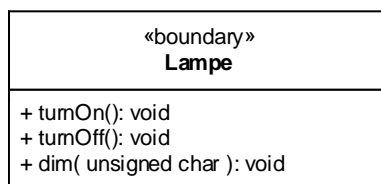
Parametre: Modtager en char fra X.10 som bruges til at finde om lampen skal. Tænde, slukke, dime op, eller dime ned.

Returværdi: ingen retur værdi.

Beskrivelse: Controller for receiveren. Holder styr på lampen nuværerne dimning samt dens houseCode og unitCode.

Attribut	Beskrivelse
houseCode	Indeholder hus koden for denne Receiver
unitCode	Indeholder unit koden for denne Receiver
CurrentDim	Den nuværerne dim værdi.

1.4.2 Lampe



turnOn

Operation: void turnOn(void)
Parametre: Ingen.
Returværdi: Ingen.
Beskrivelse: Sætter PWM for lampen til 100%.

turnOff

Operation: void turnOff(void)
Parametre: Ingen
Returværdi: Ingen
Beskrivelse: Sætter PWM for lampen til 0%.

Dim

Operation: void dim (char)
Parametre: Modtager en char med en char med den ønskede dimnings værdi.
Returværdi: Ingen
Beskrivelse: Modtager en char som har en værdi mellem 0 og til og med 9, hvor 0 = 5%,
1 = 15% osv. Skal herefter ændre pulsbredden på det ben der er forbundet
til lampen i forhold til den ovenfor angivne procent.

1.4.3 Rx10

«boundary» Rx10
+ ISR(): void

ISR

Operation: `ISR(INT0_vect)`

Parametre: Denne servicerutine er indbygget i Atmega32's IO bibliotek og modtager ingen parametre.

Beskrivelse: Denne funktion kaldes ved interrupts på PD2 INT0 og skal være **Friend** af klassen. Ved interrupts skal denne skifte den nuværende værdi på PA0 ind i **temp**. Afhængig af hvilket stadie systemet er i, skal der udføres forskellige handlinger.

checkData

Operation: `bool checkData(unsigned char)`

Parametre: Modtager 8 nulgennemgange.

Returværdi: Returnerer **TRUE**, hvis de 8 nulgennemgange overholder protokollen for 4 bits og **FALSE**, hvis den ikke gør.

Beskrivelse: Denne metode sørger for at kontrollere om en given 4-bit størrelse overholder X.10 protokollen. Der modtages 8 nulgennemgange, som deles op i 4 par. Hvert par udgør 1 bit. Hvert par skal bestå af ét 1-tal og ét 0 for at der kan returneres **TRUE**.

translate

Operation: `unsigned char translate(unsigned char)`

Parametre: Modtager 8 nulgennemgange.

Returværdi: Returnerer en **char** bestående af 0000 efterfulgt af de 4 bits, som inputtet oversættes til.

Beskrivelse: Denne metode returnerer 0000 efterfulgt af bit 7, 5, 3 og 1. Dvs hvis input er: 10011010 vil returværdien være: 00001011.

Attribut	Beskrivelse
<code>bool start</code>	Sættes til TRUE , hvis den er igang med at modtage en kommando, ellers sættes den til FALSE .
<code>unsigned char count</code>	Tæller hvor mange nulgennemgange der har været siden sidste hele bit/mønster.
<code>unsigned char temp</code>	Variabel som agerer skifteregister ved modtagelse af nulgennemgange.

1.5 Protokol

1.6 Protokol for UART

For kommunikation mellem Transmitter og PC bruges UART med 8 bits bredde. I de to forskellige stadier kan der sendes forskellige ting, under Receiving stadiet skal der sendes i rækkefølgen: Unit, Command, TimeL, TimeH.

Idle state	
Mode:	Value
Lockstatus	L,ret: L U
Newsцен	N
Stop	S

Tabel 1: Tilgængelige værdier i 'Idle' state.

Receiving state					
Unit:	Value	Command:	Value	TimeL	Value
Arne (L1)	'A'	Dim	'A' - 'J'	bin_val	0x00 - 0xFF
Carl (L2)	'C'	Tænd	'T'	TimeH:	Value
Per (TV)	'P'	Sluk	'S'		
Nils (Radio)	'N'				
				bin_val	0x00 - 0x05

Tabel 2: Tilgængelige værdier i 'Receiving' state.

1.7 Protokol for X10

Ved kommunikation over X10, skal en bit bestå af to nulgennemgange hvor de er hinandens omvendte. Dvs: 1 = 10 og 0 = 01. Udover dette er der et specifikt startmønster "1110" og et specifikt stopmønster "1111", samt en ventekode "000000". Hver kommando skal sendes to gange i stræk, adskilt af ventekoden. I Tabel 3 ses hvordan en kommando er opbygget i X10.

Startcode	4 bit	4 bit	4 bit	waitcode	4 bit	4 bit	4 bit	Stopcode
1110	House	Unit	Command	000000	House	Unit	Command	1111

Tabel 3: Mønster for hvordan en kommando sendes via X.10

4 bit værdien 'House' erstattes med en af nedenstående huskoder. Hvis koden "All off" modtages, kaldes der i hver receiver en off funktion.

House	Value	Kommentar
All off	0	Slukker alt
Lamps	1	
TV	2	
Radio	3	
Resrved	4.. 15	ikke taget i brug.

Tabel 4: Tilgængelige huskoder i X10.

Ligeledes erstattes 'Unit' med en af de nedenstående enheder:

Unit	Value
Lampe 1	1
Lampe 2	2
TV	4
Radio	8

Tabel 5: Tilgængelige huskoder i X10.

Hver enhed har hver sit sæt af kommandoer, da vi kun vælger at implementere lamper, er her en liste over kommander for lamper:

CMD_L	Value
Sluk	0
Tænd	1
Dim 5%	2
DIm 15%	3
sim ...%	...
Dim 95%	11
Reserved	12... 15

Tabel 6: Tilgængelige kommandoer for lamper.

1.8 Porte

Transmitter

- D0 (RXD) & D1 (TXD): Bruges til at kommunikere via UART med PC blokken.
- D2 (INT0): Betragtes som `ZeroCrossDetect: bool` fra Figur ?? side ??.
- D7 (OC2): Forbindes til `X.10Data: bitstream`, bruges til at sende data fra transmitter via X10.
- A0: Henter status fra kodelåsen.

Receiver

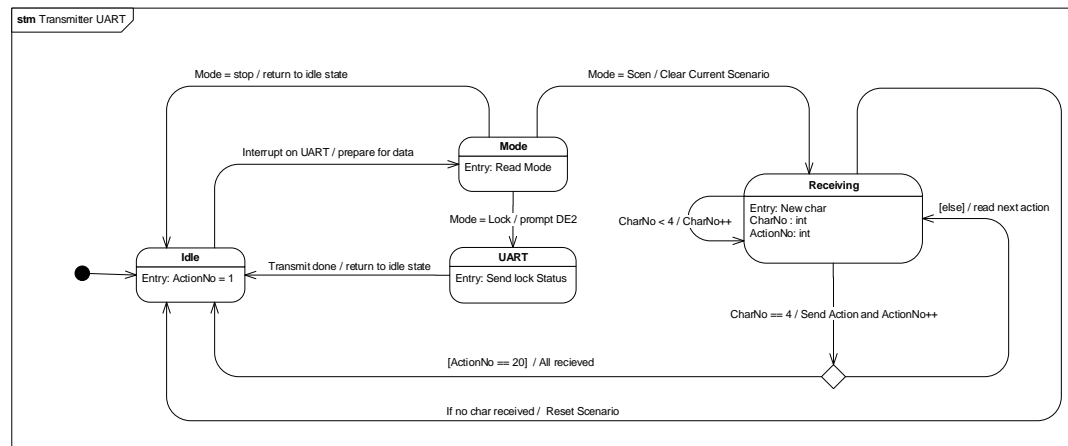
Receiver blokken ejer følgende porte på STK500:

- D0 (RXD) & D1 (TXD): Bruges til debugging af Receiveren ved at sende input fra X10 ud via UART.
- A0: Betragtes som `X.10Data: bitstream` fra Figur ?? på side ??.
- D2 (INT0): Betragtes som `ZeroCrossDetect: bool` fra Figur ?? side ??.
- D7 (OC2): Forbindes til `LightCntrl: CntrlPWM` fra ovenstående figur.

1.9 Stm diagrammer

1.9.1 Statemachines

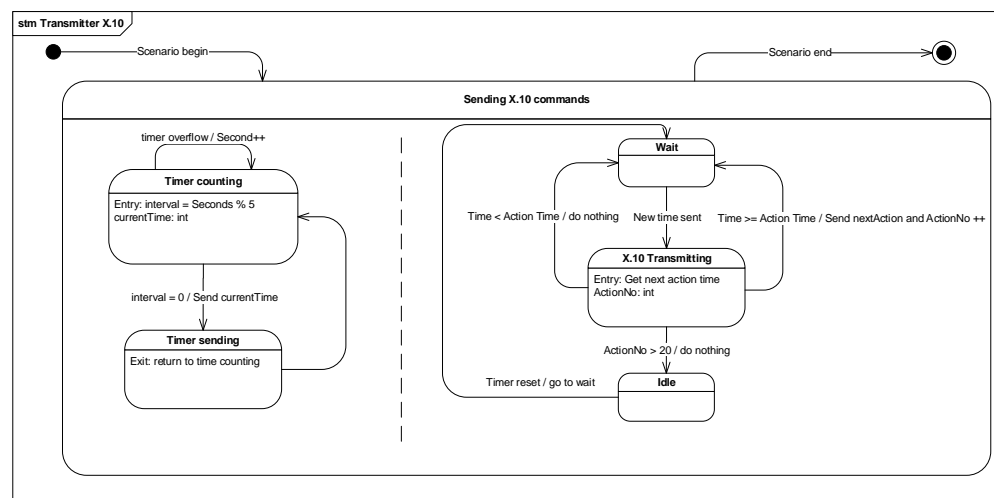
Statemachine over UART/transmitter



26

Viser Transmitteren's tilstande når PC'en sender data over UART'en. Den første block Mode er et af følgende: Lock, Stop, og Scen(Scenarie). Hvis Scen er sendt, gør transmitter klar til at modtage flere data om det nye scenarie. Efter alt data er modtaget, starter transmitteren det nye scenariet og returner til idle tilstanden.

Statemachine over X.10/transmitter



27

Den statemachine viser det parallelle forløb mellem Time, som holder styr på tiden, og transmitteren, der modtager tid fra Time, samt sender X.10 data ud til Tx10 Klassen.