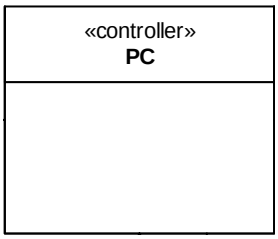
1 Softwaredesign

1.1 Version

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

1.2 PC blokken

Pc controller

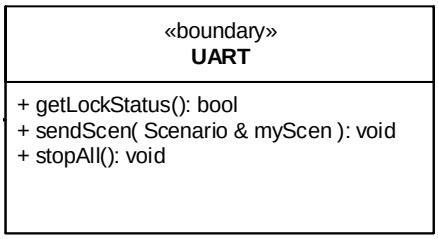


PC controlleren virker som et main program, og holder styr på følgende:

- 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 ASCII værdien for et `L` over til microcontrolleren for at få kodelåsens status. Funktionen venter på at få et svar tilbage fra microcontrolleren, hvilket enten er ASCII værdien for et `U` eller et `L`. Hvis UART'en får et `L` returneres `false`. Hvis UART'en får et `U` returneres `True`.

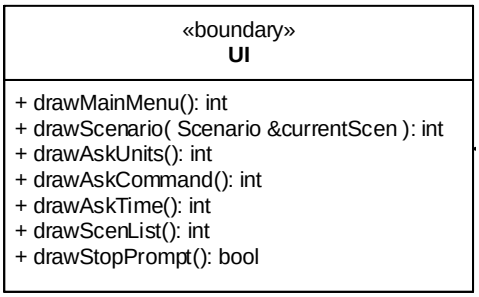
stopAll

Operation: `void stopAll(void)`
Parametre: -
Returværdi: -
Beskrivelse: Funktionen bruges til at slukke for alle enheder tilsluttet systemet. ASCII værdien for et `S` sendes for til microcontrolleren. Se evt. Protokol for UART side 23.

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: Methoden 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), kør eksisterende scenarie(2), stop scenarie(3)). Der fortages et tjek på det indtastede af brugeren, som skal valideres. validationen tjekker at det indtastede er enten 1,2 eller 3 som er de gyldige menu'er. hvis det indtastede er ugyldigt får brugeren en besked om ugyldigt input og får lov at forsøge igen. Dette sker indtil brugeren indtaster noget gyldigt.

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: Methoden udskriver en liste over de 20 aktioner der er i scenariet med at bruge ostream operatoeren der er lavet i klassen `Scenarie`. Brugeren kan derefter vælge hvilken af de 20 aktioner der skal redigeres ved at indtaste nummeret på aktionen. Når brugeren har indtastet noget, valideres dette. Hvis inputtet er den af de gyldige værdier (0-19), returneres ID'et/den gyldige værdi. Hvis inputtet er ugyldigt får brugeren en besked om ugyldigt input og får lov at forsøge igen. Dette sker indtil brugeren indtaster noget gyldigt.

drawAskUnits

Operation: `int drawAskUnits( void )`
Parametre: -
Returværdi: int : Den returnerede integer er hvilken unit der valgt
Beskrivelse: Methoden udskriver en liste over units. hvor unitsne er: Lampe1, Lampe2, TV og Radio.
brugeren kan derefter vælge hvilke af de 4 units der skal bruges, ved at indtaste nummeret på unit'en. Når brugeren har indtastet noget, valideres dette. Hvis inputtet er en af de gyldige værdier (1-4) returneres unit ID'en/den gyldige værdi. Hvis inputtet er ugyldigt får brugeren en besked om ugyldigt input og får lov at forsøge igen. Dette sker indtil brugeren indtaster noget gyldigt.

drawAskCommando

Operation: `int drawAskCommand( void )`
Parametre: -
Returværdi: int : Den returnerede integer er hvilken kommando
Beskrivelse: Methoden 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: Metoden beder brugeren om at indtaste den ønskede time-tal, hvor en ønsket kommando skal udføres. Brugeren kan derefter indtaste det ønskede time-tal. Metoden beder nu brugeren om at indtaste det ønskede minut-tal i den ønskede time. Brugeren kan nu indtaste det ønskede minut-tal.
Metoden validere nu time- og minut-tallene. hvis time-talet er indenfor det gyldige område (0-23) og minut-talet er indenfor dens gyldige område (0-59), udregnes tiden i minutter siden kl 00.00 og returneres. Hvis enten minut- eller time-tal er udenfor det gyldige område, køres metoden fra starten af, og dette sker indtil den gyldig tid indtastes.

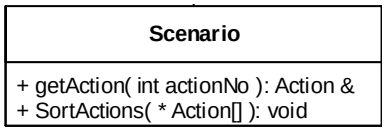
drawScenList

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 forudinstillet scenarier, ved at indtaste nummeret på det ønskede scenarie. Det indtastede valideres. Hvis det indtastede er indenfor området 1-3, returneres det indtastede, ellers bliver brugeren bedt om at prøve igen, indtil noget gyldigt er indtastet.

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. Hvis brugeren trykker på `Y / y` tasten returneres dette som et `true`, altså et 'ja'. hvis alt andet end `Y / y` trykkes returneres `false`, svarende til et 'nej'.

1.2.3 Scenario



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 er der skal bruges
Returværdi:	Action & - Reference til den valgte action.
Beskrivelse:	Methoden finder den valgte Action ud fra parameteren `actionNo`, og returnere en reference til den action.

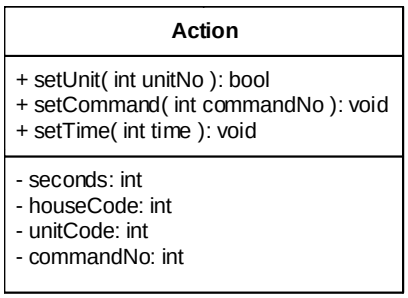
sortActions

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



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 variablen `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 linieskift.

Explicit constructor

Operation: `Action( int time, int unit, int commandNo )`

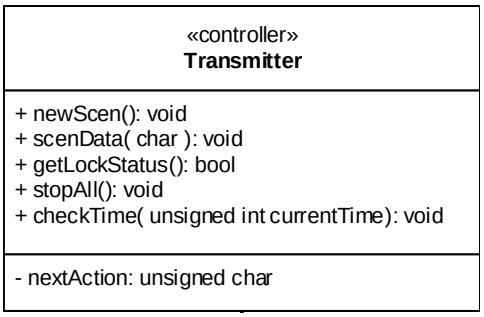
Parametre: Modtager parametre for hhv. tid, enhed og kommandonummer.

Beskrivelse: Initierer attributter i objektet, skal sætte attributterne til passende defaultværdier, hvis ingen værdier gives.

Attribut	Beskrivelse
`int seconds`	Tiden fra kl `00:00` til tiden for at den pågældende aktion skal udføres i sekunder.
`int houseCode`	Huskoden for den enhed, som aktionen skal manipulere.
`int unitCode`	Enhedskoden for den enhed, som aktionen skal manipulere.
`int commandNo`	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: Methoden 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: Methoden har ansvaret for status angående kodelåsen.

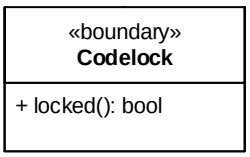
`stopAll`

Operation: `void stopAll( void )`
Parametre: –
Returværdi: –
Beskrivelse: Methoden 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: Methoden bliver kaldt fra TimeKlassen hver 5. sekund, og ansvaret for om den næste aktion skal afvikles. Dette sker kun vis den givende tid er større end den næste aktion's tid.

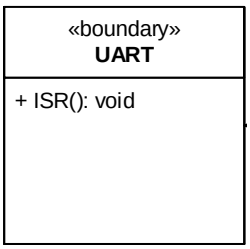
1.3.2 Codelock



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



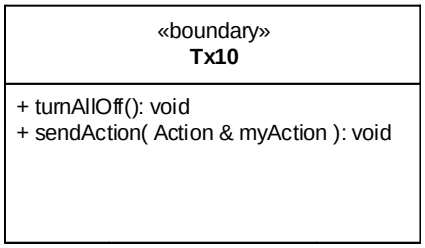
constructor

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

ISR

Operation:	`ISR()`
Parametre:	-
Returværdi:	-
Beskrivelse:	Ved interupt 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()`.

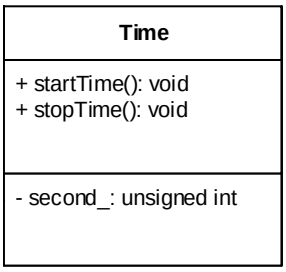
sendAction

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

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

startTime

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

stopTime

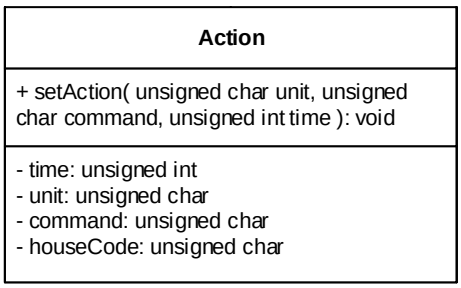
Operation: `Void stopTime ( void )`
Parametre: -
Returværdi: -
Beskrivelse: -stopper tidstælling på timeren.

operator++

Operation: `operator++()`
Parametre: -
Returværdi: -
Beskrivelse: -inkremere atributten second med 1.

Attribut	Beskrivelse
unsigned int second	indeholder tiden i sekunder.

1.3.6 Action



setAction

Operation: `void setAction( unsigned char unit, unsigned char command, unsigned int time )`

Parametre: Modtager enhed, kommando og tidspunkt relativt til starttidspunktet for aktionen.

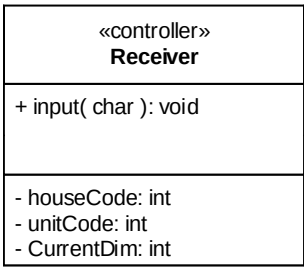
Returværdi: -

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

1.4 Receiver blokken

1.4.1 Receiver



Receiver

Operation: void input(char)

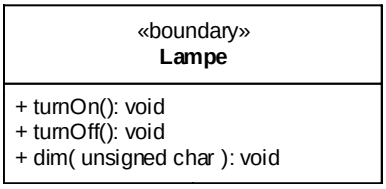
Parametre: Modtager en char fra X.10 som bruges til at finde om lampen skal. Tænde, slukke, dime op, eller dime ned.

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%.

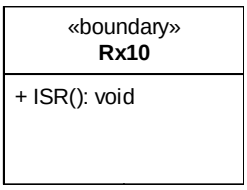
turnOff

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



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 en `char`, som repræsenterer 8 nulgennemgange.

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

Beskrivelse: Metoden tager parameteren og oversætter den til 4 bits. For at fylde `char`'en sættes bit 7-4 til `0` i returværdien. Bit 3-0 på returværdien sættes til bit 7, 5, 3 og 1 fra parameteren.
Dvs hvis metoden kaldes med en char `10011010` vil returværdien være: `00001011`.

Attribut	Beskrivelse
`bool start`	Sættes til `TRUE`, hvis den er igang med at modtage en kommando, ellers sættes den til `FALSE`.
`unsigned char count`	Tæller hvor mange nulgennemgange der har været siden sidste hele bit/mønster.
`unsigned char temp`	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
Newscen	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'		
Per (TV)	'P'	Sluk	'S'	TimeH:	Value
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 kommanoder 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

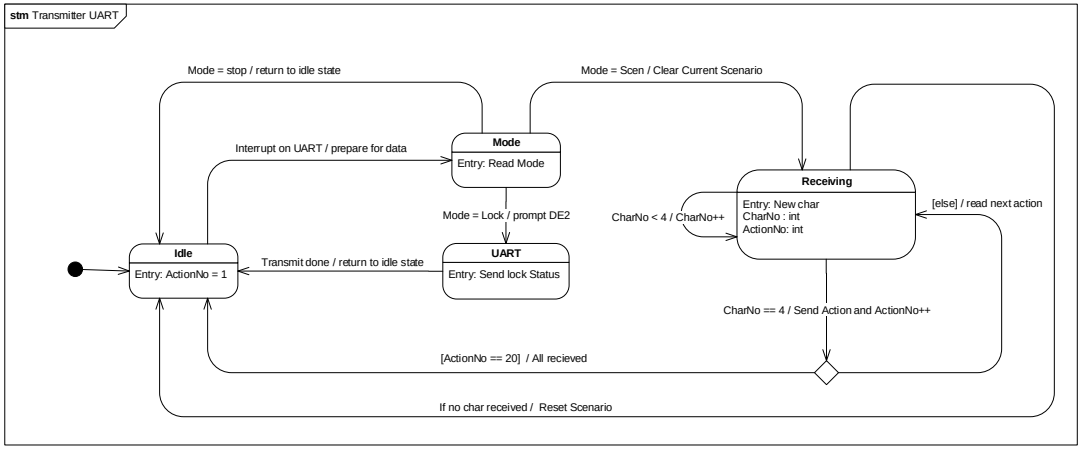
Receiver blokken ejer følgende porte på STK500:

- 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

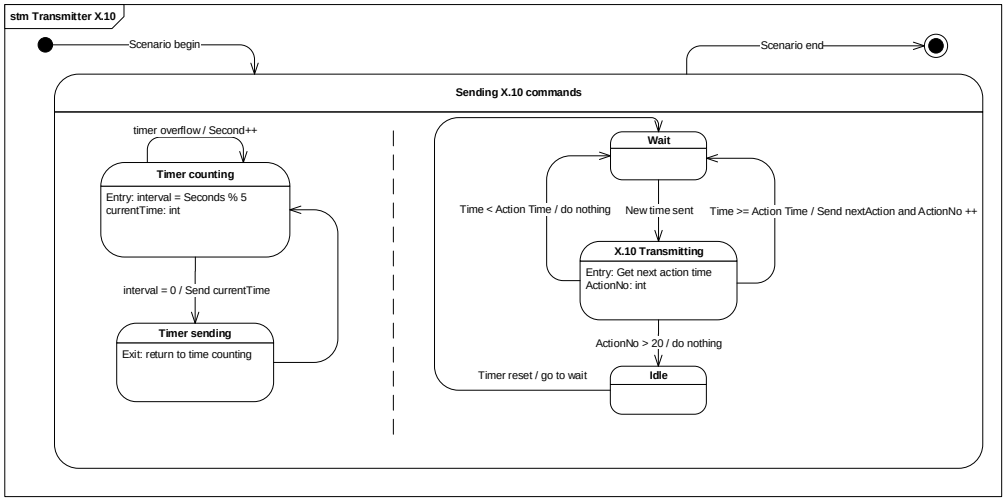
1.9.1 Statemachines

Statemachine over UART/transmitter



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



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