

Description des Tasks et Queues de :  
**WS\_CpplyRobot**

**tk\_Init**

S'exécute une fois!!  
Démarré les taches ds ordre 'yTkOrder'  
Auto delete

**tk\_Default**

Some debug tests

**EXTI15\_10\_IRQHandler(BP1)**

Sur Interrupt  
Envoie MsgX ds queue

**EXTI2\_IRQHandler(SWxy)**

Sur Interrupt  
Envoie MsgX ds queue

**tk\_CheckVR**

Toutes le temps  
Analyse le joystick VRx, VRy, SWxy  
Envoie MsgX ds queue

```
yTask.h:
typedef enum {
    TkNone = 0,
    TkInit,
    TkDefault,
    TkMoteurD,
    TkMoteurG,
    TkTrain,
    TkProcess,
    TkCheckVR,
    TkVTaffiche,
    TkAll,
} yTkOrder;
```

Les tâches sont démarrées  
ds l'ordre 'yTkOrder'

**tk\_Process**

\* recueillir un event ds la queue  
\* afficher le contenu de l'event (via queue de tk\_VTaffiche)  
\* switch Topic  
case BP1  
case SWxy  
case VRx  
case VRy  
case kbd

**qEvents**

MsgX

.Topic=BP1

.Topic=SWxy

.Topic=VRx, PayLoadF=VRx.PV

Topic=VRy, PayLoadF=VRy.PV

yMenu.c/  
yMenuGetTouche

aRxBuffer[0]

A

N

B

N

.Topic=kbd, PayLoadI=0U

.Topic=kbd, PayLoadI=1U

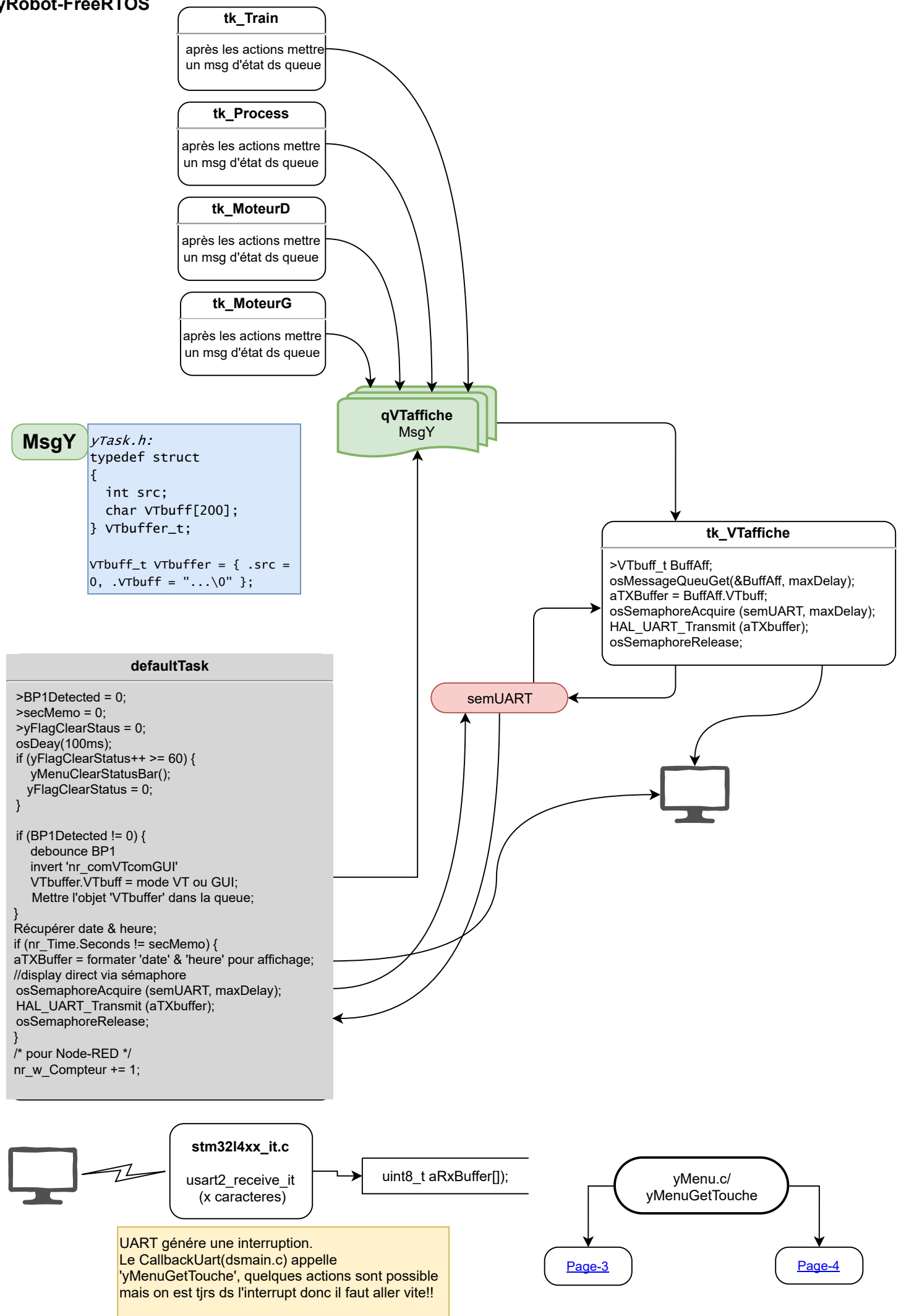
**MsgX**

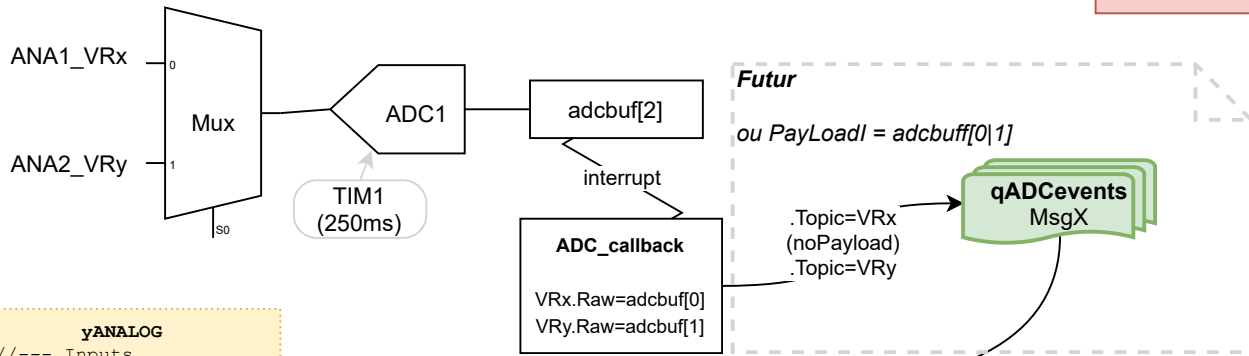
```
yTask.h:
typedef struct {
    uint8_t Topic;
    float   PayLoadF;
    uint8_t PayLoadI;
} yEvent_t;
```

```
yTask.h:
typedef enum {
    TopicNone = 0,
    BP1 = 1,
    SWxy = 2,
    kbd = 3,
    Mar = 4,
    Arr = 5,
    VRx = 11,
    VRy = 12,
    DIR = 13,
    SP = 14,
    MotD = 21,
    MotG = 22,
    RO = 31,
} yTopic;
```

```
yTask.h:
typedef enum {
    DestNone = 50,
    tkProcess = 51,
    VTaffiche = 52,
} yDestination;
```

```
yTask.h:
typedef enum {
    SrcNone = 0,
    SrcBP1 = 1,
    SrcSWxy = 2,
    SrcVRx = 5,
    SrcVRy = 6,
    SrcKbd = 11,
} ySource;
```





```

yANALOG
//--- Inputs
uint32_t Raw;
uint8_t Ri;
//--- Outputs
float PV;
float PVa;
int8_t sens;
uint8_t Ro;
//--- Parameters
float Ech_Mini;
float Ech_Maxi;
float A;
float B;
float Coef_Filtre;
float UnMoinsCoef;
float Trim;
uint32_t TrimRaw;
float Hysteresis;
//--- Memoires
float PVmemo;
float PVhyst;

```

```

yMOTOR
//--- Inputs
uint8_t MarArr;
float Speed_SP;
//--- Outputs
uint8_t inRun;
float Velocity;
//--- Paramters
uint32_t DutyCycle;
float DeadBand;
//--- memories
float Speed_memo;
uint8_t Sens_memo;
uint8_t Run_memo;
float DB_memo;
uint8_t FeeWheel;
//--- Virtual outputs
uint32_t Speed_MV;
uint8_t Sens;
uint8_t _av;
uint8_t _ar;
uint32_t _pwm;
//--- Real outputs
uint32_t _gpioPortIN1;
uint16_t _gpioPinIN1;
uint32_t _gpioPortIN2;
uint16_t _gpioPinIN2;
TIM_HandleTypeDef _htimpwm;

```

H bridge	MC33886			
	IN1	IN2	D1	/D2
* Forward	H	L	0	pwm
* Reverse	L	H	0	pwm
* FreeWLow	L	L	0	pwm
* FreeWHigh	H	H	0	pwm

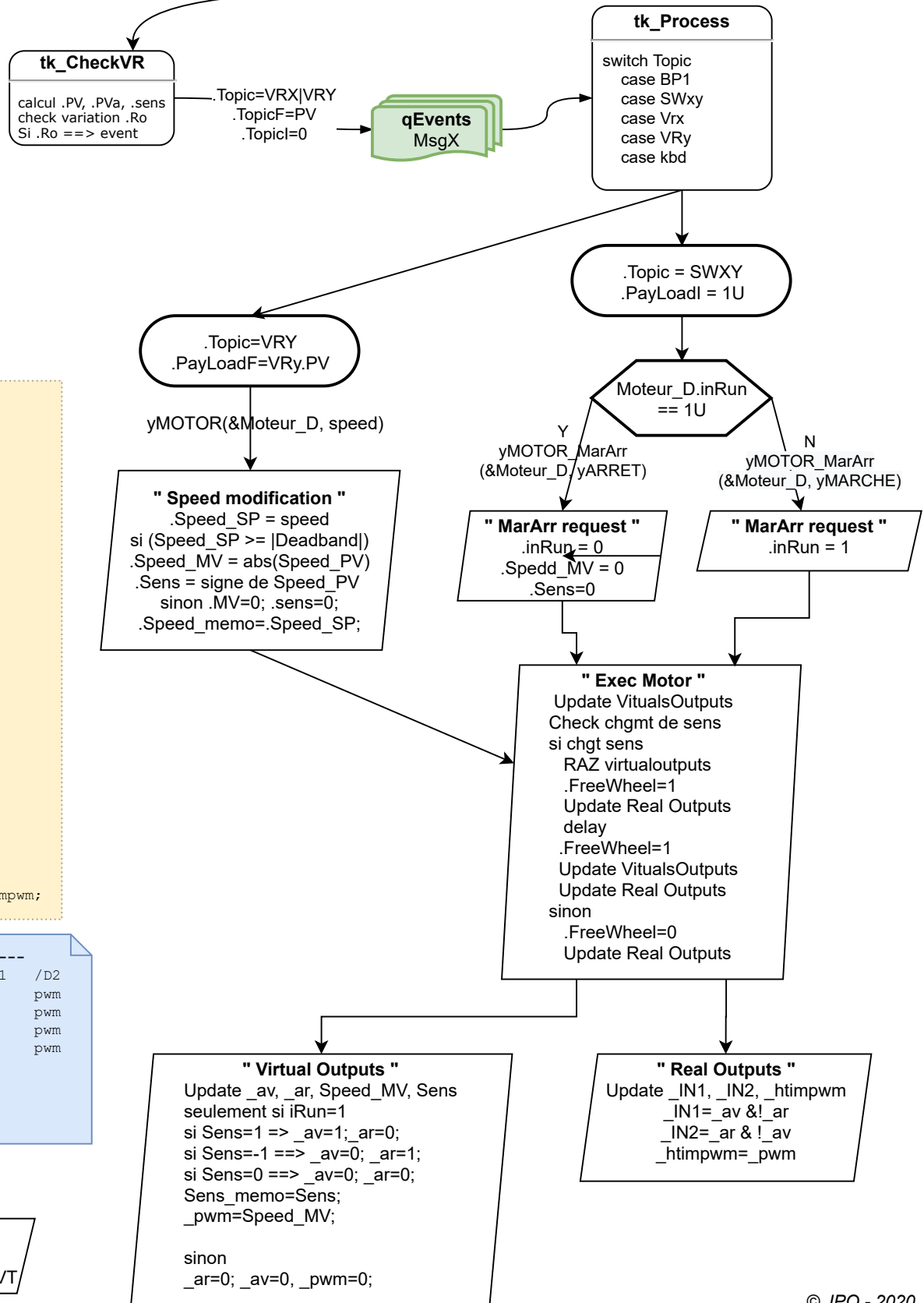
IN1 = H ==> OUT1 = H  
IN2 = H ==> OUT2 = H  
FreeWheelHigh = frein

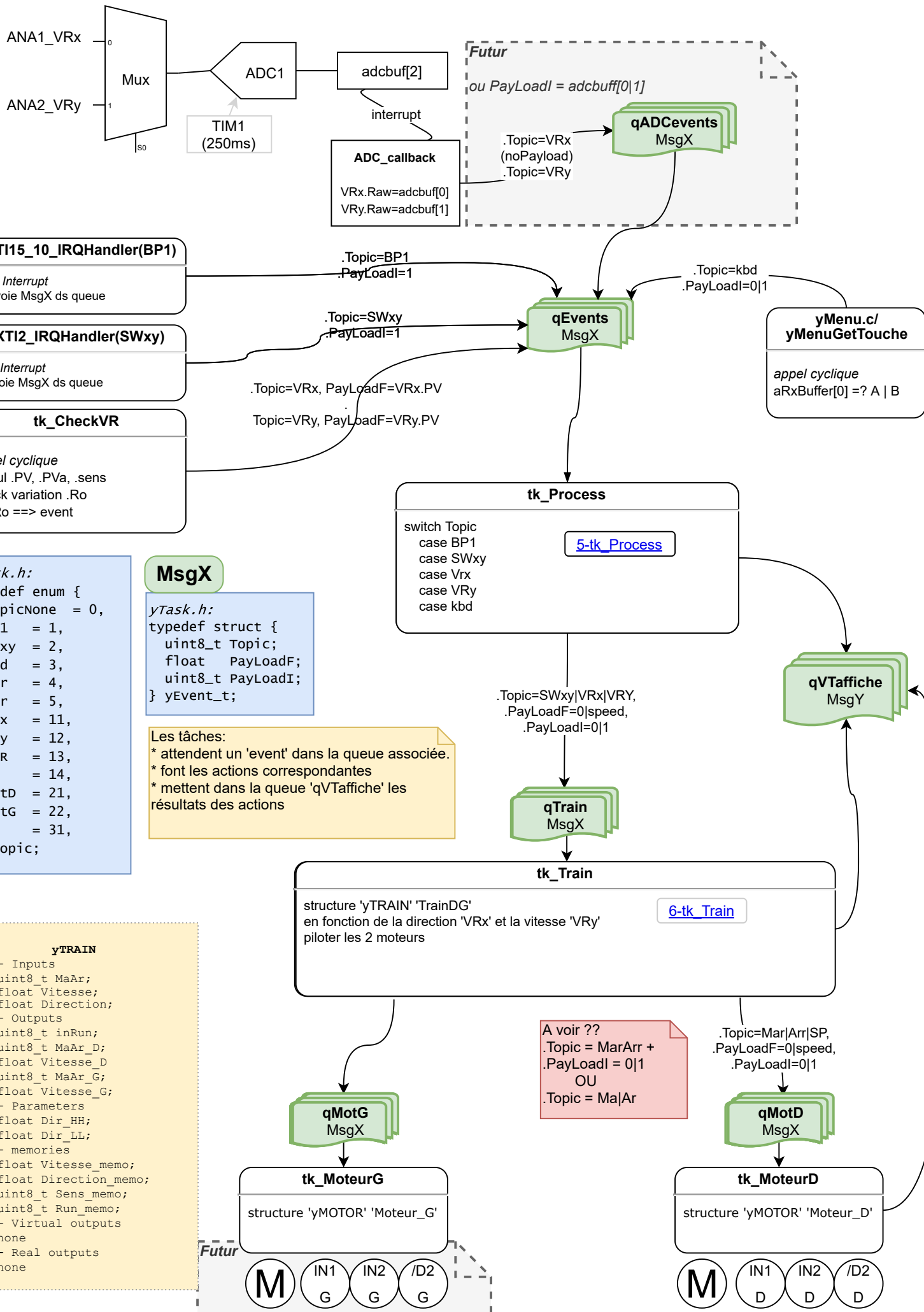
tk\_Default

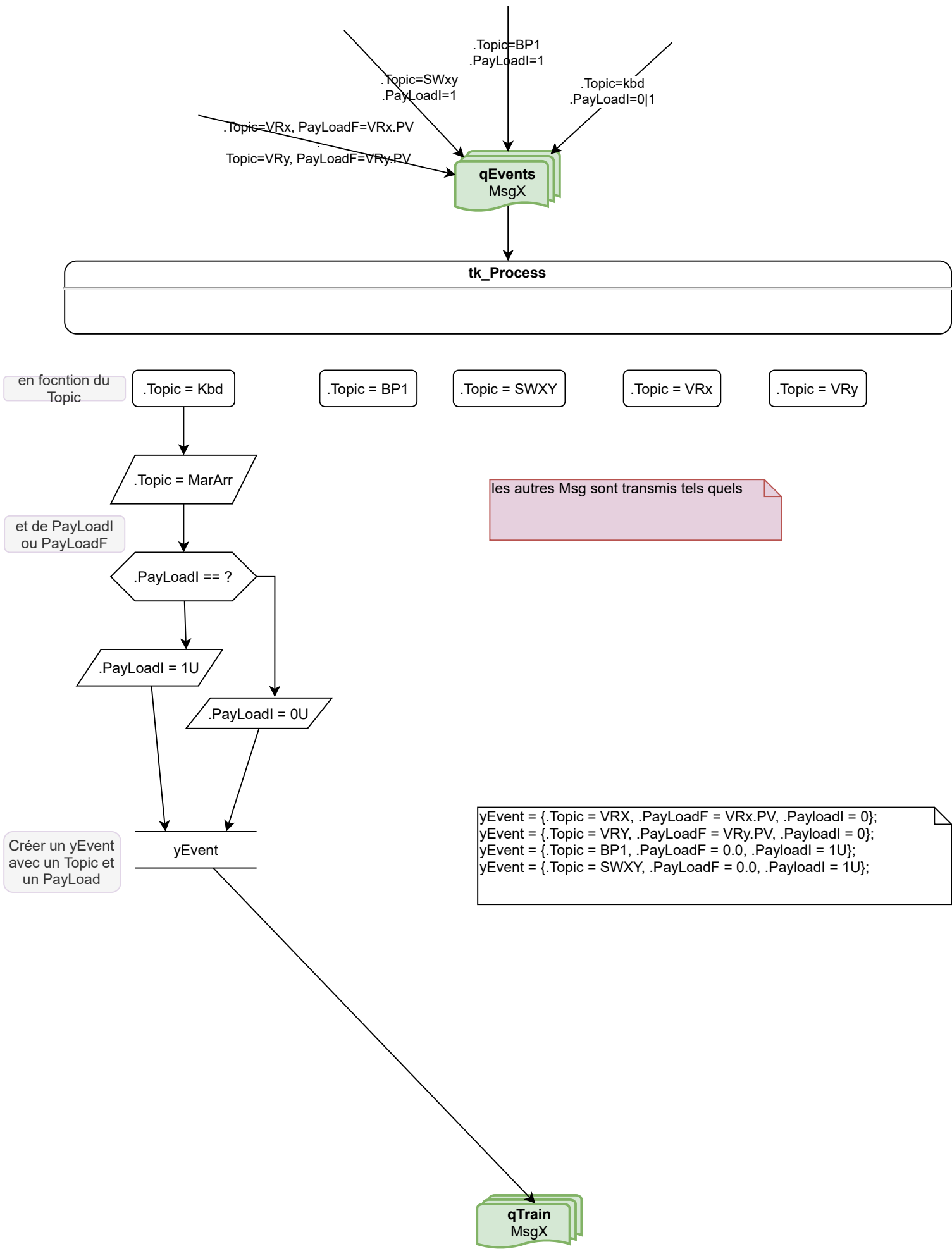
```

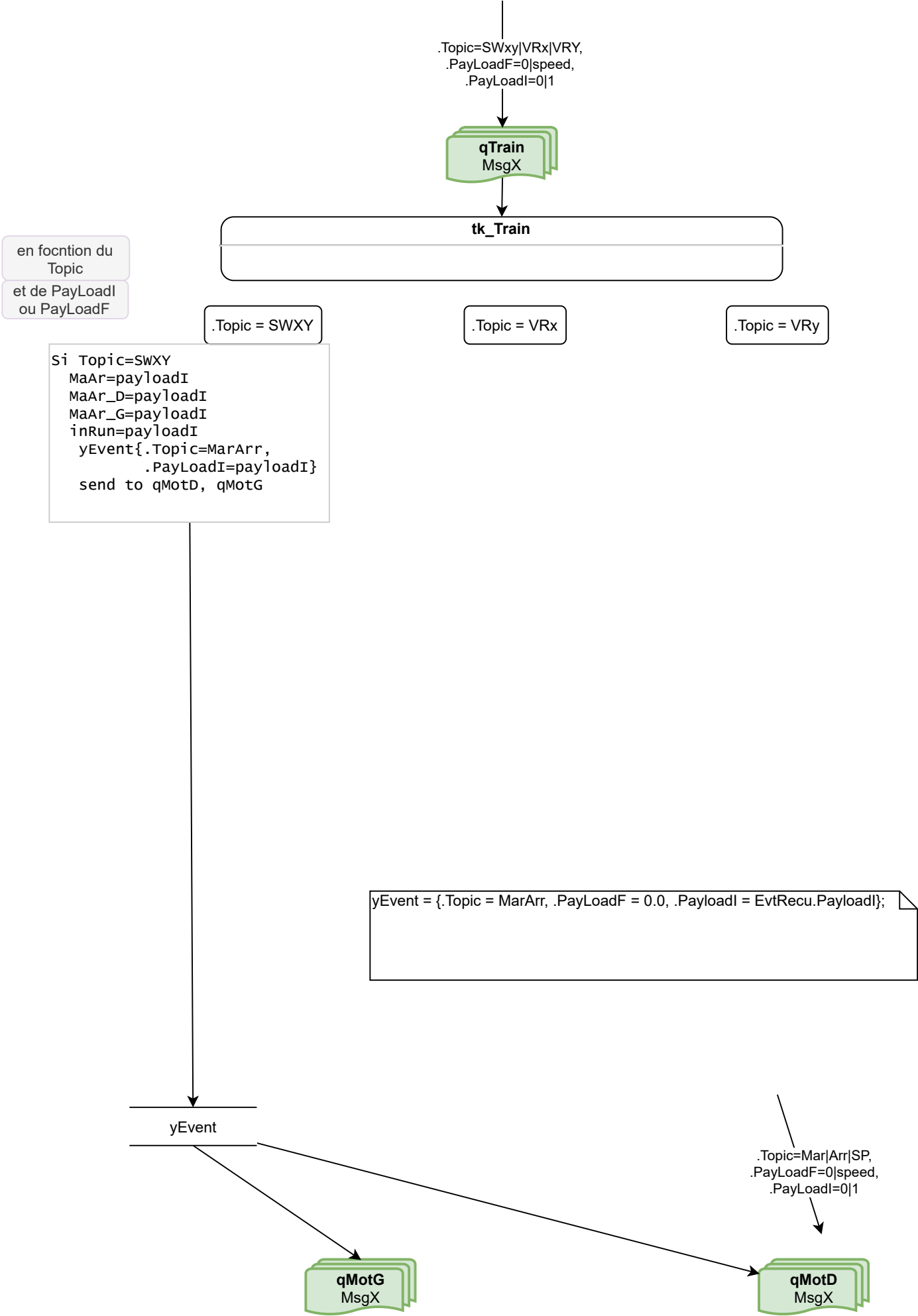
//--- traces
LD2 = Motor_D.inRun
Display états Moteur_D via tk_VT

```

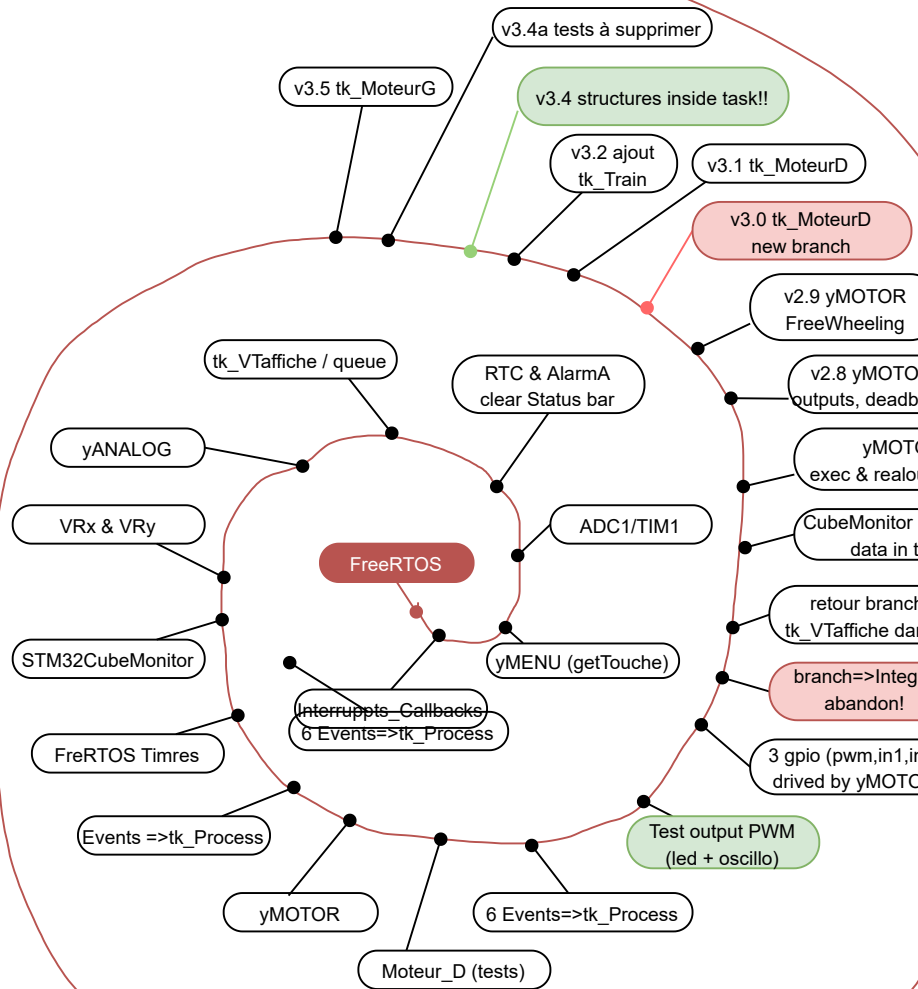


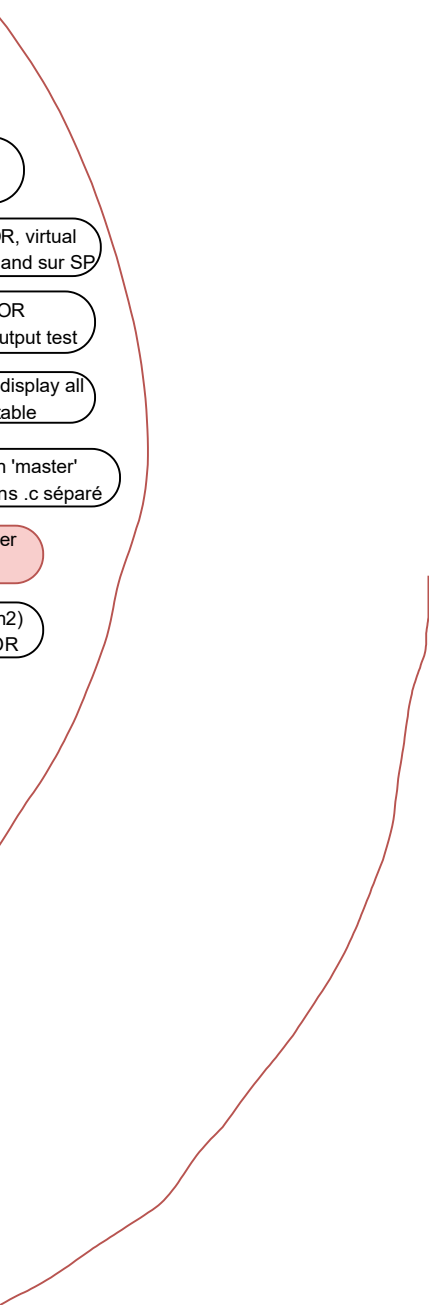






Conception en spirale!!





OR, virtual  
and sur SP

OR  
output test

display all  
table

h 'master'  
ns .c séparé

er

h2)  
OR



## yRobot-FreeRTOS

Détails de la gestion des touches saisies.  
S/prog appelé depuis une interruption UART.  
Mais qqs pbs!

```
uint8_t aRxBuffer[];
```

yMenu.c/  
yMenuGetTouche

OK

nOK

```
//traiter les touches spéciales
*** W-w //saisie 1 ou 2 char
    preparer texte et mettre ds queue'qVTaffiche'
*** C-c //effacer écran
    call yMenuClearVT
*** M-m //clear screen, afficher menu
    call yMenu Clear, Welcome, Display
    mettre ds queue 'retour curseur'
*** Q-q //quitter
    enter standby power
*** S-s //test blocage semaphore
```

uart2NbCar

= 1

= 2

aRxBuffer[0]  
= 0 a 9

Y

N

aRxBuffer[0]  
= F-h

N

VTbuffer.VTbuff = 'screen1'  
Mettre l'objet 'VTbuffer' dans la queue;

VTbuffer.VTbuff = 'Cde erronée'  
Mettre l'objet 'VTbuffer' dans la queue;

VTbuffer.VTbuff = chiffre en string  
Mettre l'objet 'VTbuffer' dans la queue;

aRxBuffer[0]  
= 1 a 9

Y

H

N

aRxBuffer[1]  
= 0 a 9

VTbuffer.VTbuff = chiffres en string  
Mettre l'objet 'VTbuffer' dans la queue;

VTbuffer.VTbuff = 'Cde erronée'  
Mettre l'objet 'VTbuffer' dans la queue;

Détails de la gestion des touches saisies.  
S/prog appelé depuis une interruption UART.  
Version sans passer par la queue!

```
uint8_t aRxBuffer[];
```

yMenu.c/  
yMenuGetTouche

```
//traiter les touches spéciales
*** W-w //saisie 1 ou 2 char
preparer texte et mettre ds queue'qVTaffiche'
*** C-c //effacer écran
call yMenuClearVT
*** M-m //clear screen, afficher menu
call yMenu Clear, Welcome, Display
mettre ds queue 'retour curseur'
*** Q-q //quitter
enter standby power
*** S-s //test blocage semaphore
```

uart2NbCar

aRxBuffer[0]

aRxBuffer[0]

48 - 57  
0 - 9

yes

aTxBuffer = chiffre en string  
Take 'semUART'  
Transmit to UART  
Release 'semUART'

no

H - h

yes

aTxBuffer = 'screen1'  
Take 'semUART'  
Transmit to UART  
Release 'semUART'

default

aTxBuffer = "Cde erronée"  
Take 'semUART'  
Transmit to UART  
Release 'semUART'

0

1 - 9

y

aRxBuffer[1]

0 - 9

default

aTxBuffer = chiffres en string  
Take 'semUART'  
Transmit to UART  
Release 'semUART'

aTxBuffer = "Cde erronée"  
Take 'semUART'  
Transmit to UART  
Release 'semUART'

1

YY - yy

y

aTxBuffer = "2char YY"  
Take 'semUART'  
Transmit to UART  
Release 'semUART'

default

aTxBuffer = "Cde erronée"  
Take 'semUART'  
Transmit to UART  
Release 'semUART'

0

