Définition de l'architecture
Compilateur de projet LDS

Documentation relatif au générateur de fichiers d'inclusion pour les processus écrit en L.D.S (Langage de Description Symbolique).

Tableau de mise à jour

Version	Date	Nom	Objet
01	12.05.2003	M. JULIENNE	Création du document

1 Préface

1.1 Documents

1.1.1 Documents de références

Acronyme	Titre / Désignation	Référence	version	Origine
DR1				

1.1.2 Documents applicables

Acronyme	Titre / Désignation	Référence	version	Origine
DA1				
DA2				
DA3				
DA4				

2 Présentation générale

Le fichier source regroupe les informations de l'architecture du projet.
L'architecture est constitué par :

- Le noyau (séquenceur).
- La liste des processus.
- Le niveau de priorité des processus.
- La liste des signaux d'échange entre les processus (le flux entre les processus).

2.1 Fichier source

Ce fichier se décompose en 2 parties.
La première partie, définie les processus, les handlers et leurs niveaux de priorité.
La deuxième partie définie les signaux d'échanges entre les processus.

2.2 Syntaxe de commande

Le programme écrit en langage C fonctionne sur un PC dans un environnement sous DOS (fonctionne sur WIN95 et NT).

Sequence.exe –i[fichierSrc] –o[FichierDebug] –e[FichierEvent] –p[FichierProcess]

- ***FichierSrc*** correspond au nom du fichier source décrivant le projet (paramètre obligatoire).
- ***FichierDebug*** correspond au nom du fichier de débugage généré par le programme projet (paramètre optionnel).
- ***FichierEvent*** correspond au nom du fichier généré par le programme décrivant tout les événement du projet et les équivalences (paramètre obligatoire).Ce fichier est utilisé par le séquenceur.
- ***FichierProcess*** correspond au nom du fichier généré par le programme décrivant tout les processus ou handler du projet et les équivalences (paramètre obligatoire).Ce fichier est utilisé par le séquenceur.

3 Définition du fichier source

Liste des mots clé :

- process
- etat
- signal
- event
- timer
- handler
- sequenceur

3.1 Syntaxe d'un process

le mot clé est : « process » en minuscule.
Syntaxe : *process (Proc, priorité, chemin)*
Proc correspond au nom du processus de 20 caractères maximum.
Priorité correspond au niveau de priorité du processus.
Chemin correspond au chemin et au nom du fichier à générer.

Exemple :

```
process   (Edit          , 100     ,def\\edit.inc)
process   (Gestion       , 50 ,def\\inout.inc)
process   (Message  , 100     ,def\\message.inc   )
process   (Flash         , 50 ,def\\flash.inc)
process   (Spi      , 50 ,def\\spi.inc)
process   (Fichier       , 200     ,def\\fichier.inc   )
```

3.1.1 Niveau de Priorité

Un processus peut avoir un niveau de priorité entre 1 et 255 où 1 est le niveau le plus haut.
Plusieurs processus peuvent avoir le même niveau de priorité.

3.2 Syntaxe d'un handler

le mot clé est : « handler » en minuscule.
Syntaxe *: handler (Hand, chemin)*
hand correspond au nom du handler de 20 caractères maximum.
Chemin correspond au chemin et au nom du fichier à générer.

Nota : Le nombre maximum de handler et de process est au total (handler + process) de 255.

Exemple :

handler (Bdf_232 ,def\\bdf_232.inc)

3.3 Syntaxe d'un séquenceur

Le mot clé est : « sequenceur » en minuscule.
Syntaxe *: sequenceur chemin*
Chemin correspond au chemin et au nom du fichier à générer.
Toutes les informations nécessaires au séquenceur seront stockées dans ce fichier.

Exemple :

Sequenceur def\\sequence.inc

3.4 Syntaxe d'un signal

Le mot clé est : « signal » en minuscule.
Syntaxe *: signal (type, procSrc,ProcDest NomEvenement)*
Type correspond au type de l'événement qui est généré. Le type peut être soit timer soit event.
ProcSrc correspond au nom du processus générateur du signal.
ProcDest correspond au nom du processus recevant le signal.
NomEvenenement correspond au nom de l'événement qui est généré.

3.4.1 Exemple 1

```
signal (timer Edit, Edit,    Reveil )
signal (event Bdf_232,             Edit,   TempoAttenteByte)
signal (event Bdf_232,             Edit,   TempoFichierEdit,)
```

3.4.2 Exemple 2

```
/*  définition du séquenceur      */
sequenceur ..\t\inc\sequence.inc

/*  définition des processus      */

process    ( Gestion              , 10    ,..\t\inc\Gestion.inc)
process    ( Acces                , 20    ,..\t\inc\Acces.inc)
process    ( Memoire              , 50    ,..\t\inc\Memoire.inc)
process    ( Usine                , 100   ,..\t\inc\Usine.inc)
process    ( SurveillanceSignal   , 150   ,..\t\inc\Surv_Sig.inc)
process    ( TraitementRF         , 200   ,..\t\inc\Trait_RF.inc)

/*  définition des handlers       */
handler TransfertUsine          ,..\t\inc\Trans_Us.inc)
handler TransfertGestion ,..\t\inc\Trans_Ge.inc)


/* signaux du processus Acces */
signal(event, Acces, Memoire    FinTelechargement
signal(event, Acces, Memoire    FinChargementLut )
signal(event, Acces, Memoire    FinChargementAle )

signal(event, Acces, Memoire    SauveRegistreEtatCudc)
signal(event, Acces, Memoire    SauveRegistreEtat8Vsb)
signal(event, Acces, Memoire    SauveRegistrePPInOutEtat )
signal(event, Acces, Memoire    SauveRegistreCtrl8Vsb )
signal(event, Acces, Memoire    SauveRegistreCtrlCudc )
signal(event, Acces, Memoire    SauveRegistreAutomaticRfLevel )

signal(event, Acces, Acces      ,PrepareChargeLut)
signal(event, Acces, Acces      ,ChargeLut)
signal(timer, Acces, Acces      ,TempoScrutation )
```

```
signal event      Acces  Gestion                 FinTelechargement
signal event      Acces  Gestion                 Acq8Vsb
signal event      Acces  Gestion                 Horloge8Vsb
signal event      Acces  Gestion                 SignalEntre
signal event      Acces  Gestion                 EtatAlimM12
signal event      Acces  Gestion                 EtatAlimP12
signal event      Acces  Gestion                 SignalSortieRF
signal event      Acces  Usine              ConfigurationUart

signal event      Acces  SurveillanceSignal   EtatCarteCudcConfigure
signal event      Acces  SurveillanceSignal   EtatCarteModulateurConfigure
signal event      Acces  SurveillanceSignal   EtatCarteAleConfigure
signal event      Acces  SurveillanceSignal   EtatCarteClipConfigure

signal event      Acces  SurveillanceSignal   EtatCarteCudcNonConfigure
signal event      Acces  SurveillanceSignal   EtatCarteModulateurNonConfigure
signal event      Acces  SurveillanceSignal   EtatCarteAleNonConfigure
signal event      Acces  SurveillanceSignal   EtatCarteClipNonConfigure


/* signaux du processus Gestion */
signal (event, Gestion,    Usine,  ModeOperationnel)
signal (event, Gestion,    Usine,  ModeMaintenance )
signal (event, Gestion,    Usine,  ModeReglage     )
signal (event, Gestion,    Acces,  SourceMPEG2     )
signal (event, Gestion,    Acces,  SourceUC        )
signal (event, Gestion,    Acces,  Frequence       )

signal (event, Gestion,    Acces,  NiveauPuissance)
signal (event, Gestion,    SurveillanceSignal  NiveauPuissance)

signal (event, Gestion,    Acces,  Retard,     )
signal (event, Gestion,    Acces,  Control8Vsb )
signal (event, Gestion,    Acces,  RfLevel     )
signal (event, Gestion,    Memoire, SauveRegistreMDR          )
signal (event, Gestion,    Memoire, SauveRegistreUTR_Freq     )
signal (event, Gestion,    Memoire, SauveRegistreUTR_Retard   )
signal (event, Gestion,    Memoire, SauveRegistreUTR_NivPuis  )
signal (event, Gestion,    Memoire, SauveRegistreCR8          )
signal (event, Gestion,    Memoire, SauveRegistreOFR      )
signal (event, Gestion,    Memoire, SauveRegistreGCR      )
signal (event, Gestion,    Memoire, SauveRegistreCFR      )

/* signaux du processus Memoire */
signal (event, Memoire,Acces,   FinChargementConfigFlash )
signal (event, Memoire,Acces,   ChargementLut )
signal (event, Memoire,Acces,   ZoneLutNok )
signal (event, Memoire,Acces,   ChargementAleAcuScale )
signal (event, Memoire,Acces,   ChargementAleAcuScaleBis )
```

```
signal (event, Memoire,Acces,    ZoneAleNok )

signal (timer, Memoire,Memoire,          TempoProgramMax )
signal (event, Memoire,Gestion, DefautInterne )
signal (event, Memoire,Memoire,          ProgrammeRegistre )

signal (event, Memoire,Memoire,          ProgrammeTableau )

signal (timer, Memoire,Memoire,          TempoEffaceZone )
signal (timer, Memoire,Memoire,          TempoEffaceMax )
signal (event, Memoire,Memoire,          ProgrammeFormate )
signal (event, Memoire,Memoire,          ProgrammeFichier )

signal (event, Memoire,   Usine,         ProgrammationFichierOK )
signal (event, Memoire,   Usine,         TimeOutProgrammationFichier )
signal (event, Memoire,   Usine,         ErreurProgrammationFichier )
signal (event, Memoire, Usine,           MessageErreur )

/* signal (event, Memoire, TraitementRF,        InitTraitementRF )
signal event,       Memoire,      SurveillanceSignal,   RestaurationConfigUsine
signal event,       Memoire,      SurveillanceSignal,   RestaurationConfigParDefaut

/* signaux du processus Usine */
signal (event, Usine, Memoire            , SauvegardeFichier )
signal (event, Usine, Memoire            , DebugAfficheEtatFlash )
signal (timer, Usine, Usine              , TempoAttenteFichier )
signal (event, Usine, SurveillanceSignal, DemandeUtilisationBuffer )
signal (event, Usine, SurveillanceSignal, FinUtilisationBuffer )
signal (event, Usine, SurveillanceSignal, ModeAdaptatif       )
signal (event, Usine, SurveillanceSignal, ModeFixe     )

/* signaux du handler TransfertUsine */
signal (event, TransfertUsine,     Usine, Commande )
signal (timer, TransfertUsine,     Usine, TempoAttenteByte )
signal (event, TransfertUsine,     Usine, ErreurFichierTropGros )
/* signaux nouveaux de du handler TransfertUsine      */
signal (event, TransfertUsine,     Usine, ErreurDebordement )
signal (event, TransfertUsine,     Usine, TrameCorrecte )

/* signaux du handler TransfertGestion */
signal (event, TransfertGestion, Gestion, ErreurAbsenceEntete )
signal (event, TransfertGestion, Gestion, ErreurAbsenceFinTrame )
signal (event, TransfertGestion, Gestion, ErreurDebordement )
signal (event, TransfertGestion, Gestion, TrameCorrecte )

/* signaux du processus SurveillanceSignal */
signal (timer, SurveillanceSignal , SurveillanceSignal  , DemandeUtilisationBuffer )
signal (event, SurveillanceSignal , Gestion              , Qualite )
signal (event, SurveillanceSignal , Usine                , UtilisationBufferPermis )
```

```
signal (event, SurveillanceSignal        , Acces                , ReglageManuelRfLevel      )
signal (event, SurveillanceSignal        , Acces                , ReglageAutomaticRfLevel   )
signal (event, SurveillanceSignal        , Acces                , ReglageOffsetId           )
signal (event, SurveillanceSignal        , Acces                , ReglageOffsetQd           )
signal (event, SurveillanceSignal        , Acces                , ReglageQuadM              )
signal (event, SurveillanceSignal        , Acces                , ReglageQuadD              )
signal (event, SurveillanceSignal, SurveillanceSignal   , CalculShoulder )
signal (timer, SurveillanceSignal, SurveillanceSignal   , CalculShoulder )

/* signaux du processus TraitementRF */
signal (event, TraitementRF, TraitementRF              , Bidon
```

4 Annexe

4.1 Algorithme d'analyse du fichier source

4.1.1 1ère passe

La 1ère analyse du fichier va consister à :

- Initialise une variable « NbErreur » à 0.
- Initialise une variable « NbWarning » à 0.
- calculer le nombre de ***process*** max.
- Calculer le nombre ***handler*** max.
- Allouer un tableau (« TableauProcessHandler ») pour stoker le nom des handler et des process suivant la structure suivant :
  Char Nom [DimMaxProcessHandler] ;
  Char NomFichier[DimMaxNomFichier] ;
  Unsigned char Type ; où type peut etre soit handler soit process.(Niveau de priorité par défaut le niveau est de 255 pour un process).
- Initialise une variable « NbProcess » à 0.
- Initialise une variable « NbHandler » à 0.
- calculer le nombre d'***event*** max.
- Calculer le nombre ***timer*** max.
- Allouer un tableau (« TableauEventTimer ») pour stoker le nom des event et des timer suivant la structure suivant :
  Char Nom [DimMaxEventTimer] ;
  Unsigned char Type ; où type peut être soit event soit timer.
- Initialise une variable « NbEvent » à 0.
- Initialise une variable « NbTimer » à 0.
- Initialise une variable « NbEventTimer» à 0.
- Calculer le nombre ***signal*** max.
- Allouer un tableau (« TableauSignal ») pour stoker le nom des signaux suivant la structure :
  short ProcSrc ;
  short ProcDest ;
  short Event ;
- Initialise une variable « NbSignal » à 0.

4.1.2 2ème passe

La 2éme passe d'analyse du fichier consiste à lecture d'une ligne du fichier et en fonction du premier mot correspondant à une commande :

- vérifier par rapport au tableau (« TableauProcessHandler ») l'existence du process où du handler.
  Si non, le stock dans le tableau et incrémente un variable NbProcess ou NbHandler.
  Si oui, on informe l'utilisateur par un message d'erreur.
- vérifier par rapport au tableau (« TableauEventTimer ») l'existence de l'événement ou du timer.
  Si non, le stock dans le tableau et incrémente un variable Nbevent ou NbTimer et incrémente la variable NbEventTimer.
  Si oui, on informe l'utilisateur par un message d'erreur.

4.1.3 3ème passe

La 3éme passe d'analyse du fichier consiste à lecture d'une ligne du fichier et en fonction du premier mot correspondant à une commande « signal » :

- vérifier par rapport au tableau (« TableauProcessHandler ») l'existence du process où du handler pour le process source et pour le process destinataire.
- vérifier par rapport au tableau (« TableauEventTimer ») l'existence de l'événement ou du timer.
- vérifier par rapport au tableau (« TableauSignal ») l'existence de l 'état.
  Si non, informe l'utilisateur par un message d'erreur.
  Si oui, le stock dans le tableau et incrémente un variable NbSignal.

4.1.4 Message d'erreur.

Le message d'erreur consiste à la génération d'un « bip » et une indication de l'erreur (ligne et type d'erreur) dans le fichier de trace.
De plus, à la fin de l'analyse un rapport totalisant le nombre d'erreur est affiché.

4.2 Génération des fichiers

Génère un fichier avec la liste des processus et de sa constante équivalente.
Génère un fichier avec la liste des événement ou timer et sa constante équivalente.

4.2.1 Fichier associé au séquenceur

Génération d'une constante « NB_PROCESS « correspondant au nombre de processus max.
Génération d'une constante « NB_TIMER « correspondant au nombre de timer max.

Génération d'un tableau :

```
StructProcess
{
        const unsigned char NiveauPriorité ;
        unsigned char Etat ;
} ;

StructProcess TableauProcessus[NB_PROCESS] = {
        {NiveauPrioritéProcess0 , EtatProcess0 },
        {NiveauPrioritéProcess1 , EtatProcess1 },
        {NiveauPrioritéProcess2 , EtatProcess2 },
        {…, …},
        {NiveauPrioritéProcess(n-1) , EtatProcess(n-1) },
        {NiveauPrioritéProcess(n) , EtatProcess(n) } } ;
```

Lors de la génération du tableau la variable « EtatProcess » est initialisé à 0.
Les niveaux de priorité sont redéfinis pour qu'ils commencent à 0 et sans trou dans les priorités.
Génération d'un tableau :

```
StructTimer
{
        unsigned char FlagEtatTimer ;
        const unsigned char ProcessSrc;
        const unsigned char ProcessDest ;
        const unsigned char Evenement ;
        unsigned long Param ;
        unsigned short CptTimer ;
} ;

StructTimer TableauTimer[NB_TIMER] = {
{} } ;
```

4.2.2 Fichiers associés aux processus

Une liste des processus utilisé par ce processus lors de la déclaration des signaux.
Une liste des événements utilisés par ce processus lors de la déclaration des signaux.
Une déclaration d'équivalence de type :

```
#define PROCESS     x
```

4.2.3 Fichier associés aux handler

```
StructProcess
{
        const unsigned char NiveauPriorité ;
        unsigned char Etat ;
} ;

TableauProcessus[NB_PROCESS] =
        {
        {NiveauPrioritéProcess0 , EtatProcess0 },
        {NiveauPrioritéProcess1 , EtatProcess1 },
        {NiveauPrioritéProcess2 , EtatProcess2 },
        {…, …},
        {NiveauPrioritéProcess(n-1) , EtatProcess(n-1) },
        {NiveauPrioritéProcess(n) , EtatProcess(n) }
        } ;
```