



# Analyse de traces d'exécution des microcontrôleurs

Azzeddine Amiar  
[Azzeddine.Amiar@imag.fr](mailto:Azzeddine.Amiar@imag.fr)

Lydie du Bousquet  
[Lydie.Du-bousquet@imag.fr](mailto:Lydie.Du-bousquet@imag.fr)

Roland Groz  
[Roland.Groz@imag.fr](mailto:Roland.Groz@imag.fr)



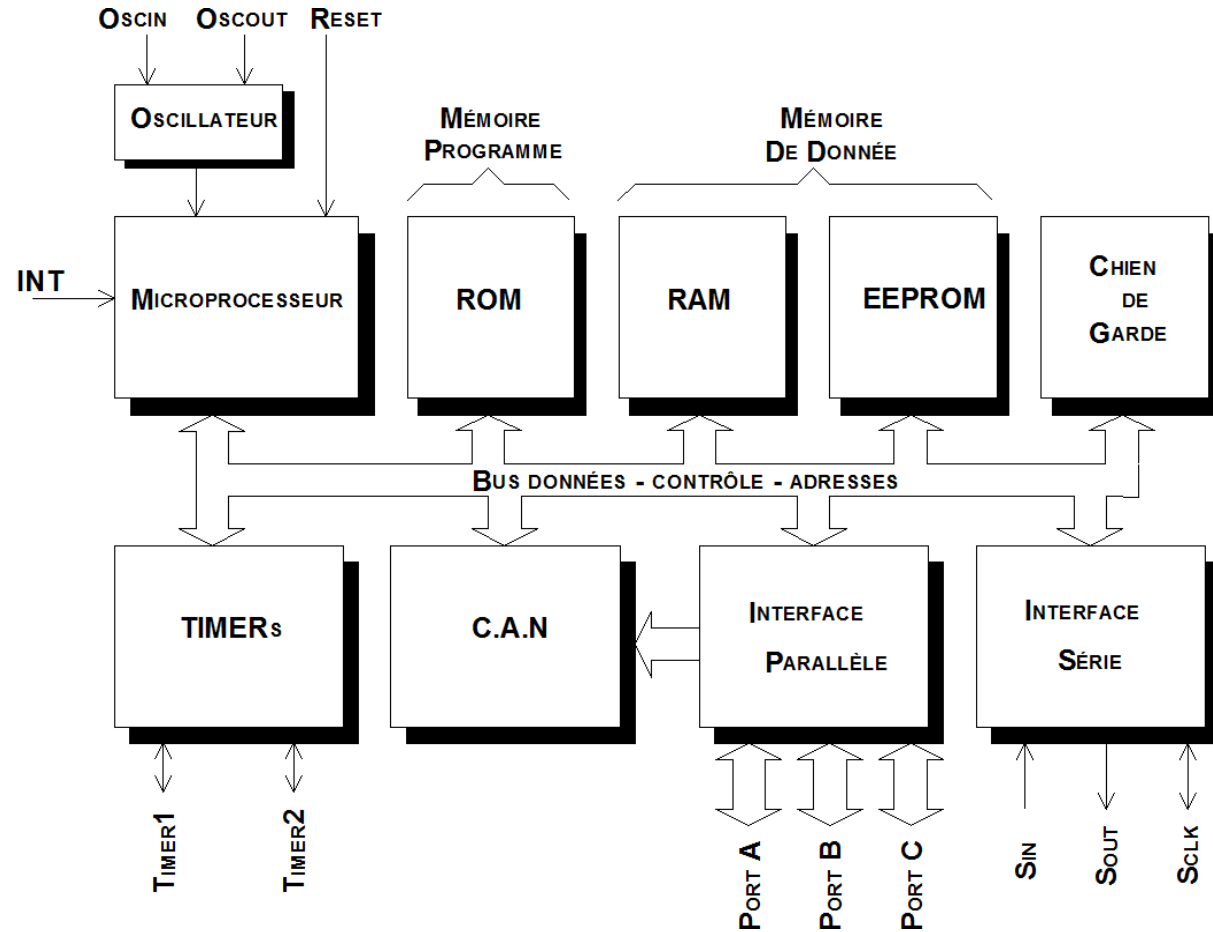
# SOMMAIRE

- 1. Microcontrôleurs et contexte de travail**
- 2. Résumé de la trace d'exécution**
- 3. Expérimentation**
- 4. Conclusion**

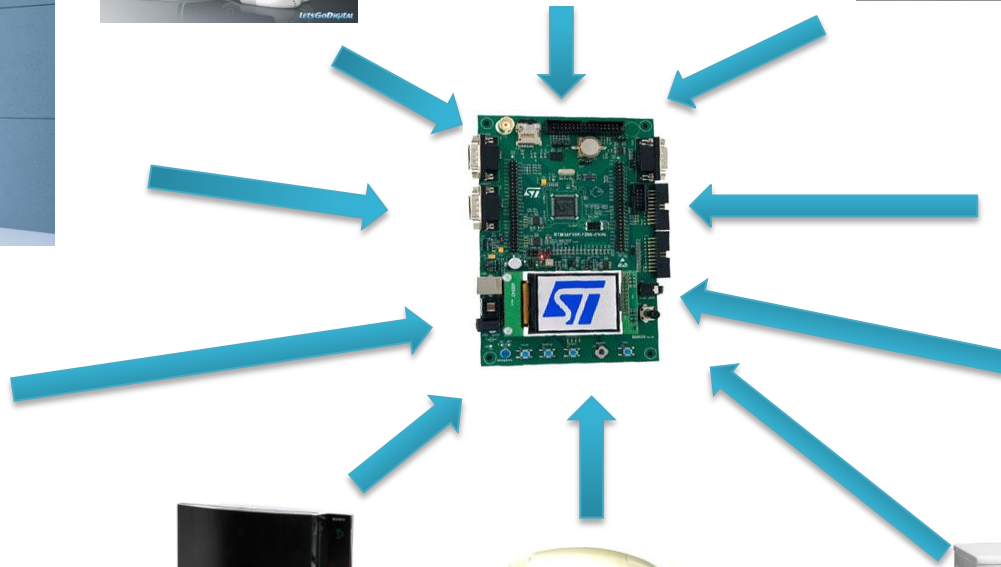
# SOMMAIRE

- 1. Microcontrôleurs et contexte de travail**
2. Résumé de la trace d'exécution
3. Expérimentation
4. Conclusion

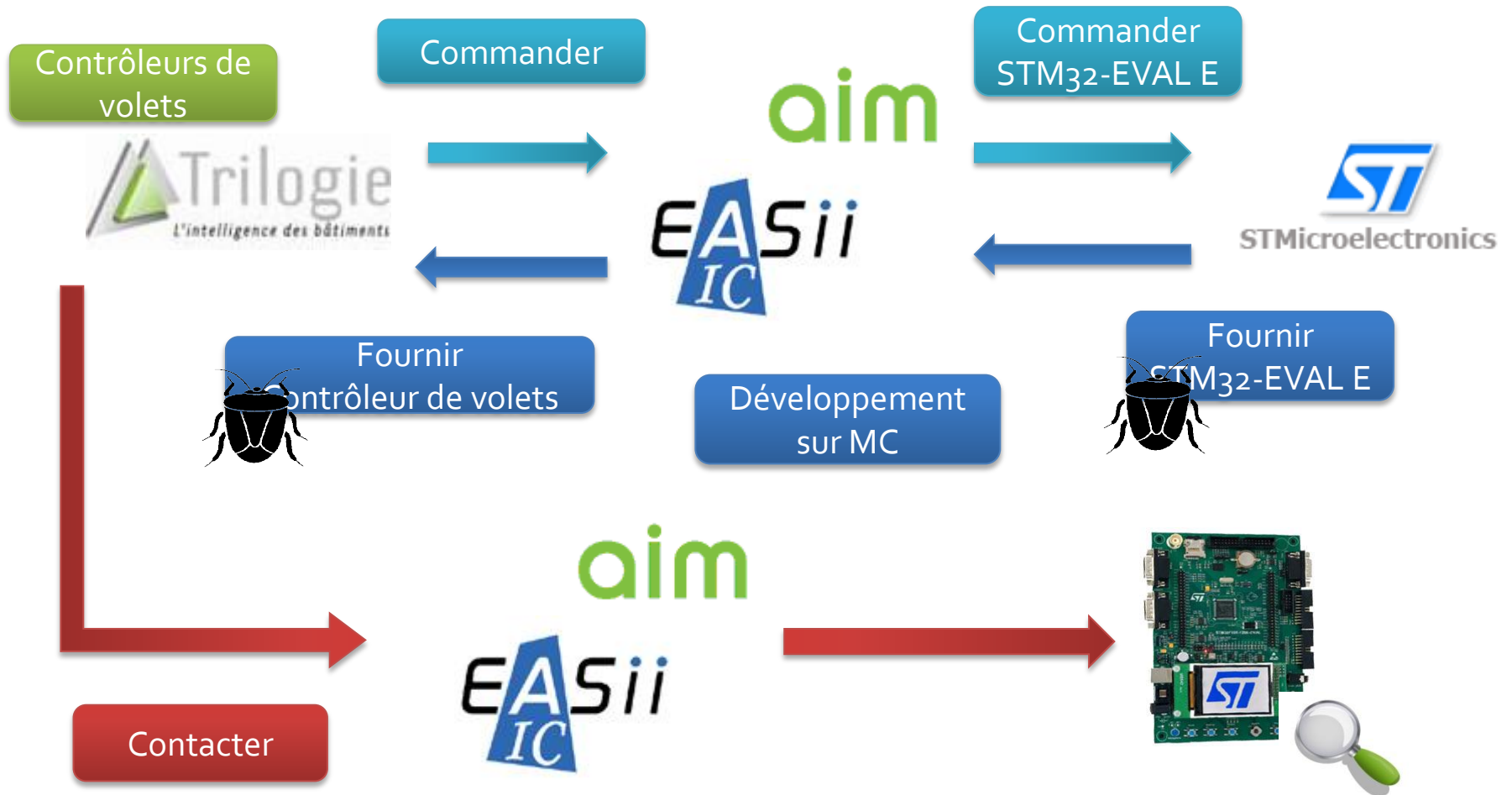
# Microcontrôleur



# Contexte



# Débogage d'applications embarquées





# Débogage d'applications embarquées

- Analyse de signaux électriques
- Simulation (mais pas toujours possible)

=>Coûteuse en temps

=>Difficulté de compréhension élevée



# Débogage d'applications embarquées

- Développement d'applications embarquées à la portée de non-spécialistes
- Nombreux environnements de développement pour des applications
- Peu d'outils dédiés à la mise au point  
=>Travail de validation/diagnostic longs et fastidieux

# Motivation

- Microprocesseurs *ARM Cortex-Mx*
- Partie dédiée (sur microprocesseur) à la trace, ETM (Embedded Trace Macrocell)
- Extraction de la trace en utilisant une sonde JTAG
- Projet FUI IO32 vise à développer un outil d'aide au diagnostic en phase d'exploitation

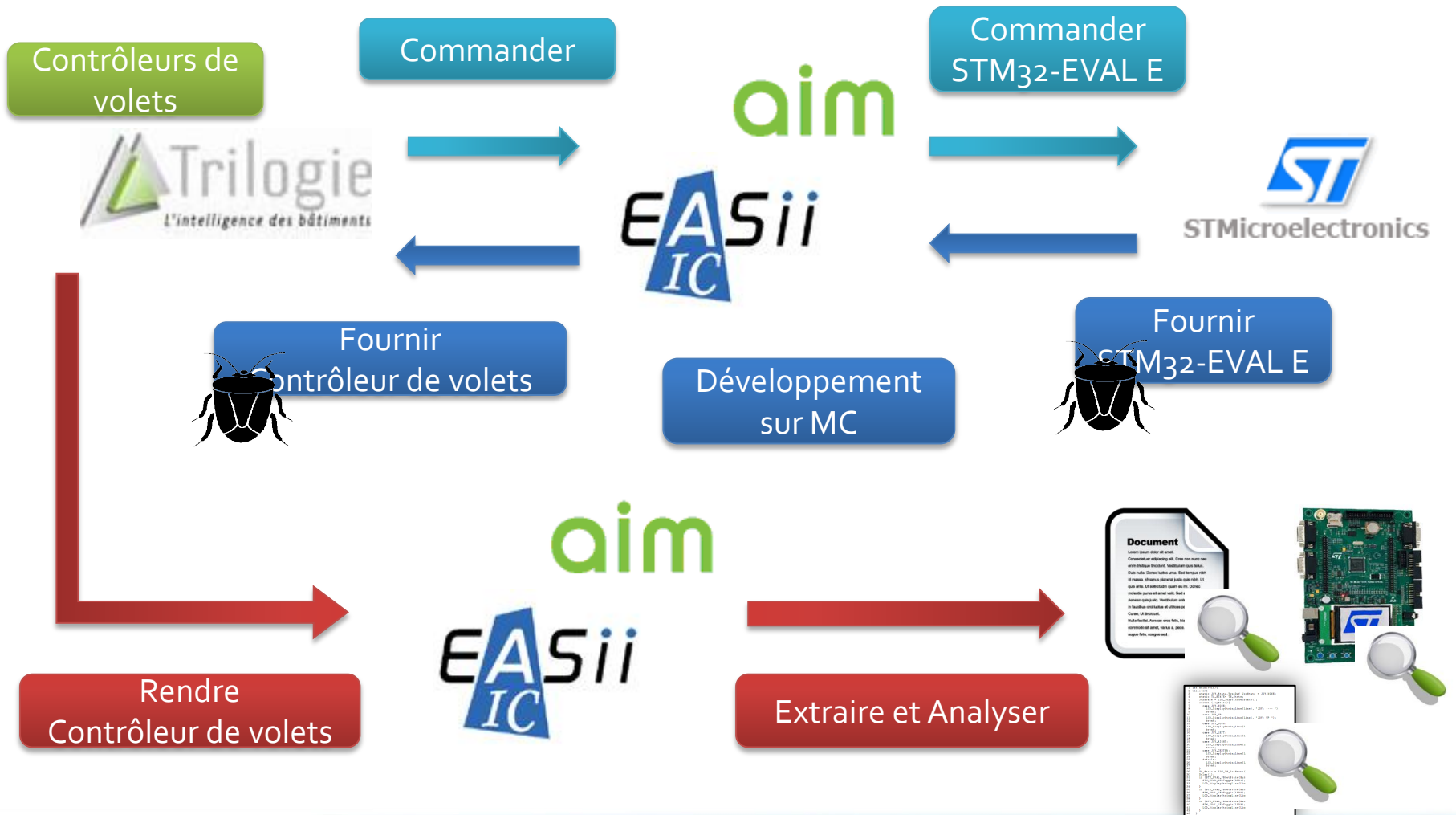




# Objectifs

- Proposer des méthodes pour décrire les traces de façon abstraite
  - Trace de taille réduite => analyse plus rapide
- Proposer des méthodes de diagnostique

# Débogage d'applications embarquées



# SOMMAIRE

1. Microcontrôleurs et contexte de travail
2. **Résumé de la trace d'exécution**
3. Expérimentation
4. Conclusion

# Trace - Définition

```
4 Index,Time (in s),Address/Port,Instruction/Data,Function,Source Code/Extra Info
5 3731,7351.866943208 s,X : 0x08001638," LDR r0,[pc,#4] ; @0x08001640,"TPIU_ReadITATBCTR0"," return(TPIU->ITATBCTR0);"
6 3732,7351.866943208 s,X : 0x0800163A," LDR r0,[r0,#0x00],"TPIU_ReadITATBCTR0",""
7 3733,7351.866943342 s,X : 0x0800163C," BX lr","TPIU_ReadITATBCTR0"," }"
8 17391,7351.867195592 s,X : 0x080018F0," POP {r4,pc},"led_check"," }"
9 17392,7351.867195592 s,X : 0x08001990," BL.W TPIU_ReadITATBCTR0 (0x08001638)","main"," Value = TPIU_ReadITATBCTR0();"
10 17393,7351.867195592 s,X : 0x08001638," LDR r0,[pc,#4] ; @0x08001640,"TPIU_ReadITATBCTR0"," return(TPIU->ITATBCTR0);"
11 17394,7351.867195592 s,X : 0x0800163A," LDR r0,[r0,#0x00],"TPIU_ReadITATBCTR0",""
12 17395,7351.867195725 s,X : 0x0800163C," BX lr","TPIU_ReadITATBCTR0"," }"
13 17396,7351.867195725 s,X : 0x08001994," LDR r1,[pc,#136] ; @0x08001A20","main",""
14 17397,7351.867195725 s,X : 0x08001996," STR r0,[r1,#0x00],"main",""
15 17398,7351.867195725 s,X : 0x08001998," BL.W TPIU_ReadETMData (0x08001620)","main"," Value = TPIU_ReadETMData();"
16 17399,7351.867195725 s,X : 0x08001620," MOV r0,#0x02","TPIU_ReadETMData"," TPIU->ITCTRL = 2; //0b10 = Integration data tes
17 17400,7351.867195725 s,X : 0x08001624," LDR r1,[pc,#8] ; @0x08001630","TPIU_ReadETMData",""
18 17401,7351.867195725 s,X : 0x08001626," STR r0,[r1,#0x00],"TPIU_ReadETMData",""
19 17402,7351.867195725 s,X : 0x08001628," LDR r0,[pc,#8] ; @0x08001634","TPIU_ReadETMData"," return(TPIU->FIFO_DATA_0);"
20 17403,7351.867195725 s,X : 0x0800162A," LDR r0,[r0,#0x00],"TPIU_ReadETMData",""
21 17404,7351.867196008 s,X : 0x0800162C," BX lr","TPIU_ReadETMData"," }"
22 17405,7351.867196008 s,X : 0x0800199C," LDR r1,[pc,#128] ; @0x08001A20","main",""
23 17406,7351.867196008 s,X : 0x0800199E," STR r0,[r1,#0x00],"main",""
24 17407,7351.867196008 s,X : 0x080019A0," BL.W TPIU_SetITATBCTR2 (0x08001644)","main"," TPIU_SetITATBCTR2();"
25 17408,7351.867196008 s,X : 0x08001644," MOV r0,#0x01","TPIU_SetITATBCTR2"," TPIU->ITCTRL = 1; //0b01 = Integration test mo
26 17409,7351.867196008 s,X : 0x08001648," LDR r1,[pc,#16] ; @0x0800165C","TPIU_SetITATBCTR2",""
27 17410,7351.867196008 s,X : 0x0800164A," STR r0,[r1,#0x00],"TPIU_SetITATBCTR2",""
28 17411,7351.867196008 s,X : 0x0800164C," LDR r1,[pc,#16] ; @0x08001660","TPIU_SetITATBCTR2"," TPIU->ITATBCTR2 = 1; //0b01
29 17412,7351.867196008 s,X : 0x0800164E," STR r0,[r1,#0x00],"TPIU_SetITATBCTR2",""
30 17413,7351.867196008 s,X : 0x08001650," MOV r0,#0x00","TPIU_SetITATBCTR2"," TPIU->ITCTRL = 0; //0b00 = Normal mode"
31 17414,7351.867196008 s,X : 0x08001654," LDR r1,[pc,#4] ; @0x0800165C","TPIU_SetITATBCTR2",""
32 17415,7351.867196008 s,X : 0x08001656," STR r0,[r1,#0x00],"TPIU_SetITATBCTR2",""
33 17416,7351.867196342 s,X : 0x08001658," BX lr","TPIU_SetITATBCTR2"," }"
34 17417,7351.867196342 s,X : 0x080019A4," BL.W TPIU_ReadITATBCTR0 (0x08001638)","main"," Value = TPIU_ReadITATBCTR0();"
35 17418,7351.867196342 s,X : 0x08001638," LDR r0,[pc,#4] ; @0x08001640,"TPIU_ReadITATBCTR0"," return(TPIU->ITATBCTR0);"
36 17419,7351.867196342 s,X : 0x0800163A," LDR r0,[r0,#0x00],"TPIU_ReadITATBCTR0",""
37 17420,7351.867196525 s,X : 0x0800163C," BX lr","TPIU_ReadITATBCTR0"," }"
38 17421,7351.867196525 s,X : 0x080019A8," LDR r1,[pc,#116] ; @0x08001A20","main",""
39 17422,7351.867196525 s,X : 0x080019AA," STR r0,[r1,#0x00],"main",""
40 17423,7351.867196525 s,X : 0x080019AC," BL.W TPIU_ReadETMData (0x08001620)","main"," Value = TPIU_ReadETMData();"
41 17424,7351.867196525 s,X : 0x08001620," MOV r0,#0x02","TPIU_ReadETMData"," TPIU->ITCTRL = 2; //0b10 = Integration data tes
42 17425,7351.867196525 s,X : 0x08001624," LDR r1,[pc,#8] ; @0x08001630","TPIU_ReadETMData",""
43 17426,7351.867196525 s,X : 0x08001626," STR r0,[r1,#0x00],"TPIU_ReadETMData",""
44 17427,7351.867196525 s,X : 0x08001628," LDR r0,[pc,#8] ; @0x08001634","TPIU_ReadETMData"," return(TPIU->FIFO_DATA_0);"
45 17428,7351.867196525 s,X : 0x0800162A," LDR r0,[r0,#0x00],"TPIU_ReadETMData",""
46 17429,7351.867196525 s,X : 0x0800162C," BX lr","TPIU_ReadETMData"," }"
47 17430,7351.867196525 s,X : 0x0800162E," MOVs r0,r0,"???",""
48 17431,7351.867196525 s,X : 0x08001630," LSRs r0,r0,#28,"???",""
49 17432,7351.867196642 s,X : 0x08001632," B 0x0800163E","???",""
50 17433,7351.867196642 s,X : 0x080019A4," BL.W TPIU_ReadITATBCTR0 (0x08001638)","main"," Value = TPIU_ReadITATBCTR0();"
51 17434,7351.867196642 s,X : 0x08001638," LDR r0,[pc,#4] ; @0x08001640,"TPIU_ReadITATBCTR0"," return(TPIU->ITATBCTR0);"
52 17435,7351.867196642 s,X : 0x0800163A," LDR r0,[r0,#0x00],"TPIU_ReadITATBCTR0",""
```

- Trace réduite
- > 60 000 de lignes

- Trace : suite d'événements (ruptures de séquence) correspondant à l'exécution
- Une *rupture de séquence* : possibilité que la prochaine instruction à exécuter ne soit pas directement la suivante de celle actuellement traitée

# Problématique

- Trace d'exécution : {3, 8, 11, 14, 17, 20, 23, 26, 32, 36, 40}
  - Contient une boucle infinie (Ligne 2)
- => Une *explosion* de la taille de la trace
- Résumer la trace d'exécution

```
1 int main(void){
2 while(1){ ←
3     static JOY_State_TypeDef JoyState = JOY_NONE;
4     static TS_STATE* TS_State;
5     JoyState = IOE_JoyStickGetState();
6     switch (JoyState){
7         case JOY_NONE:
8             LCD_DisplayStringLine(Line5, "JOY: ---- ");
9             break;
10        case JOY_UP:
11            LCD_DisplayStringLine(Line5, "JOY: UP ");
12            break;
13        case JOY_DOWN:
14            LCD_DisplayStringLine(Line5, "JOY: DOWN ");
15            break;
16        case JOY_LEFT:
17            LCD_DisplayStringLine(Line5, "JOY: LEFT ");
18            break;
19        case JOY_RIGHT:
20            LCD_DisplayStringLine(Line5, "JOY: RIGHT ");
21            break;
22        case JOY_CENTER:
23            LCD_DisplayStringLine(Line5, "JOY: CENTER ");
24            break;
25        default:
26            LCD_DisplayStringLine(Line5, "JOY: ERROR ");
27            break;
28    }
29    TS_State = IOE_TS_GetState();
30    Delay(1);
31    if (STM_EVAL_PBGetState(Button_KEY) == 0){
32        STM_EVAL_LEDToggle(LED1);
33        LCD_DisplayStringLine(Line4, "Pol: KEY Pressed ");
34    }
35    if (STM_EVAL_PBGetState(Button_TAMPER) == 0){
36        STM_EVAL_LEDToggle(LED2);
37        LCD_DisplayStringLine(Line4, "Pol: TAMPER Pressed ");
38    }
39    if (STM_EVAL_PBGetState(Button_WAKEUP) != 0){
40        STM_EVAL_LEDToggle(LED3);
41        LCD_DisplayStringLine(Line4, "Pol: WAKEUP Pressed ");
42    }
43 }
44 }
```



# Résumé de trace

- Réduire la taille/complexité des traces d'exécution
- Exploiter la présence de la boucle active
  - Construire le résumé de la trace autour de cette boucle
  - *Origine de la boucle*, l'événement qui définit la boucle



# Patrons de trace - Définition

- Une trace  $\sigma$  est un mot sur  $\Sigma$
- L'ensemble des traces est dénoté  $\Sigma^*$
- Un événement  $d \in \Sigma$  est identifié comme *origine de la boucle*
  - L'ensemble des traces possibles du programme devient  $(d. \Sigma \setminus \{d\})^*$
- $Patrons(\sigma, d) = \{ \sigma' \in \Sigma^* \mid \sigma'(1) = d$   
 $\wedge \exists i, j \leq |\sigma| : \sigma' = \sigma_{i..j}$   
 $\wedge \forall k \in [2..|\sigma'|] : \sigma'(k) \neq d \}$

# Patrons de trace - Exemple

•  $\sigma_1 = \text{AAAAAAAAAA}$



– Le seul patron de  $\sigma_1$  est : [A] qui se répète neuf fois

•  $\sigma_2 = \text{ABCABADAA}$



– Les patrons de  $\sigma_2$  sont : [ABC], [AB], [AD] et [A] x 2

•  $\sigma_3 = \text{ABCBCABDB.}$



– Les patrons de  $\sigma_3$  sont : [ABCBC] et [ABDB]



# Approche

- Soit une trace  $\sigma = ABC ABC AD AC AD AD$
- Chercher des Patrons similaires (voisins) commençant par l'origine de la boucle active
  - $\sigma = [ABC] [ABC] [AD] [AC] [AD] [AD]$
- Générer des résumés sans/avec perte d'information

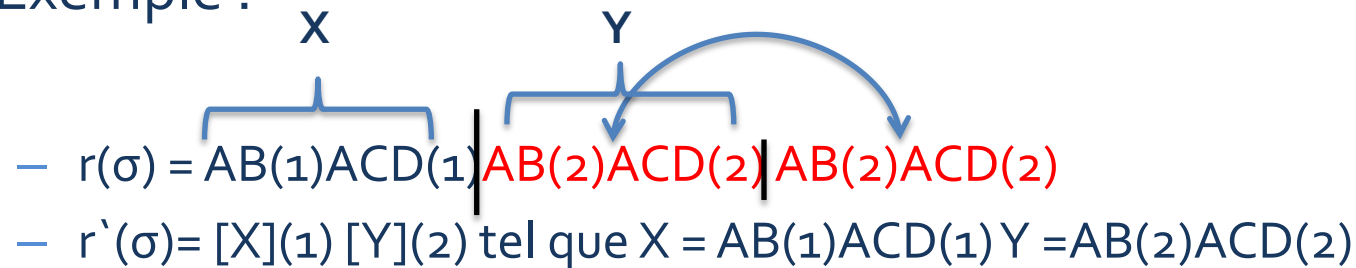
# Résumé de trace sans perte d'information

- Conserver le séquençement des événements de la trace d'exécution
  - Reconstruire la trace initiale à partir du résumé
  - Rester utilisable et compatible dans les techniques d'analyse dynamique comme le test
- Exemple:
  - $\sigma = \text{ABACDABABACDACDABABACDACD}$   
          ↑   ↑    ↑   ↑    ↑    ↑    ↑   ↑    ↑    ↑
  - $r(\sigma) = \text{AB}(1)\text{ACD}(1)\text{AB}(2)\text{ACD}(2) \text{AB}(2)\text{ACD}(2)$

# Abstraction du résumé

- Résumer un résumé de trace  
=> Niveau d'abstraction

- Exemple :



–  $r(\sigma) = AB(1)ACD(1)AB(2)ACD(2)AB(2)ACD(2)$

–  $r'(\sigma) = [X](1) [Y](2)$  tel que  $X = AB(1)ACD(1) Y = AB(2)ACD(2)$

- Niveau d'abstraction : nombre d'exécutions de l'algorithme d'abstraction sur une trace ou son résumé



# Résumé avec perte d'information

- Généralisation de l'approche du résumé sans perte d'information
- Cette généralisation ignore le nombre d'occurrence des patrons
  - $AB(2) \approx AB(8)$ , et seront tout les deux remplacé par  $AB(*)$
- Exemple:
  - $\sigma = ABACDABABACDACDABABACDACD$
  - Sans perte :  $r(\sigma) = [AB(1)ACD(1)](1)[AB(2)ACD(2)](2)$
  - Avec perte:  $r(\sigma) = [AB(*)ACD(*)](*)$

# SOMMAIRE

1. Microcontrôleurs et contexte de travail
2. Résumé de la trace d'exécution
3. **Expérimentation**
4. Conclusion



# Expérimentation - Etapes

- Développement d'un prototype (Python) pour résumer les traces.
- Sept programmes :
  - Horloge
  - Trois programmes mutants d'Horloge
  - Aquarium
  - Le Pendu
  - Jeu de LEDs
- Raccordement sonde JTAG (KEIL Ulink pro)
- Chargement / Exécution du programme sur le microcontrôleur
- Récupération de la trace

# Expérimentation

- Sans perte d'information:

Trace	Taille T	Taille R	Lignes T	Lignes R	Niveau	$\Delta$
Horloge 1	129 Ko	11 Ko	32887	2041	2	93,79%
Horloge 2	13 Ko	2 Ko	3190	322	2	89,91%
M1 Horloge	8 Ko	1 Ko	1942	105	2	94,59%
M2 Horloge	20 Ko	1 Ko	4874	167	2	96,57%
M3 Horloge	15 Ko	2 Ko	3570	238	2	99,33%
Aquarium	127 Ko	1 Ko	29368	6	2	99,98%
Pendu	10 Ko	1 Ko	1965	42	2	97,86%
LEDs	185 Ko	1 Ko	62239	24	3	99,96%

Lignes T - Lignes R

- Avec perte d'information:

Trace	Taille T	Taille R	Lignes T	Lignes R	Niveau	$\Delta$
Horloge 1	129 Ko	11 Ko	32887	27	4	99,92%
Horloge 2	13 Ko	1 Ko	3190	4	4	99,87%
M1 Horloge	8 Ko	1 Ko	1942	22	2	99,87%
M2 Horloge	20 Ko	1 Ko	4874	4	4	99,92%
M3 Horloge	15 Ko	1 Ko	3570	4	3	99,89%
Aquarium	127 Ko	1 Ko	29368	3	3	99,99%
Pendu	10 Ko	1 Ko	1965	42	2	97,86%
LEDs	185 Ko	1 Ko	62239	26	3	99,96%

$$\Delta = \frac{\text{Lignes T} - \text{Lignes R}}{\text{Lignes T}} \times 100$$

Lignes T



# Expérimentation - Conclusion

- Efficacité des deux approches à fournir une vue plus abstraite de l'information à traiter
    - Taux d'abstraction  $\geq 89\%$
  - L'approche sans perte d'information semble plus intéressante
    - Permet la sauvegarde du séquençement des événements, et le nombre de leurs occurrences
- =>Meilleur analyse/compréhension du système

# SOMMAIRE

1. Microcontrôleurs et contexte de travail
2. Résumé de la trace d'exécution
3. Expérimentation
4. **Conclusion**

## Méthode Standard

- Analyse de signaux électriques
- Simulation
- Couteuse en temps
- Difficulté de compréhension élevée

## Nouvelle Méthode

- Analyse de trace, du code statique, microcontrôleur
- Analyse post-mortem
- Analyse plus rapide
- Difficulté réduite

**Merci de votre attention!**

Questions?