Ingénierie des µ-contrôleurs, 4A MT

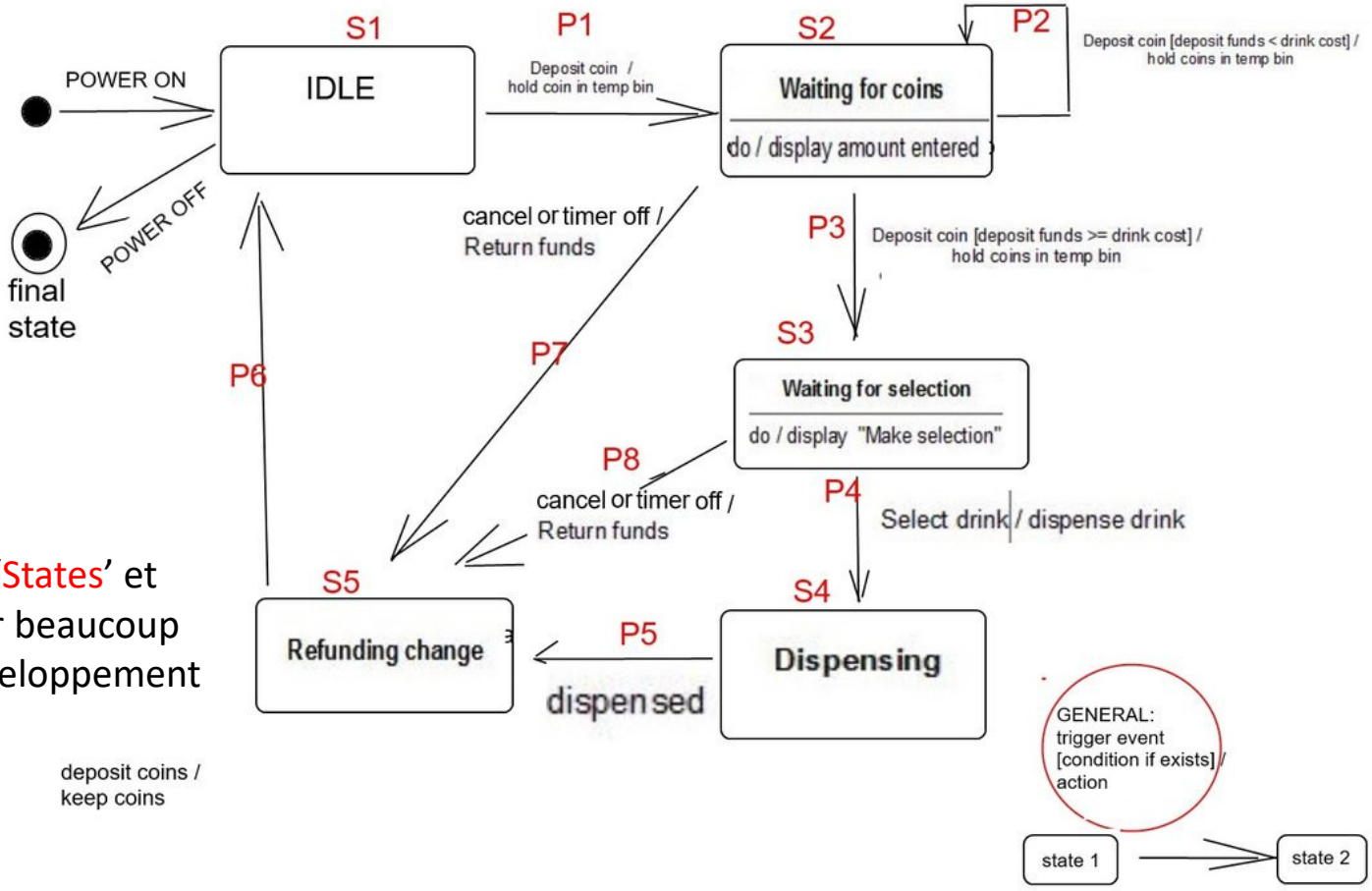
3ème séance : TP3 « passage des rapports atomatique »

Supplément de l'énoncé

2023

TP3.3 : Réalisation d’une machine à état avec les tableau de saut (jump tables)

Exemple : diagramme d'état de distributeur de café

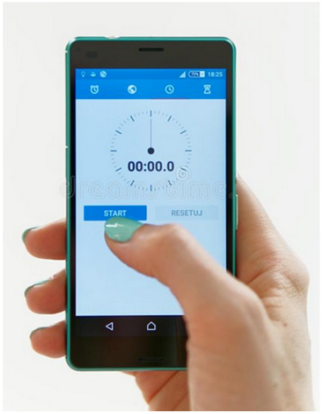


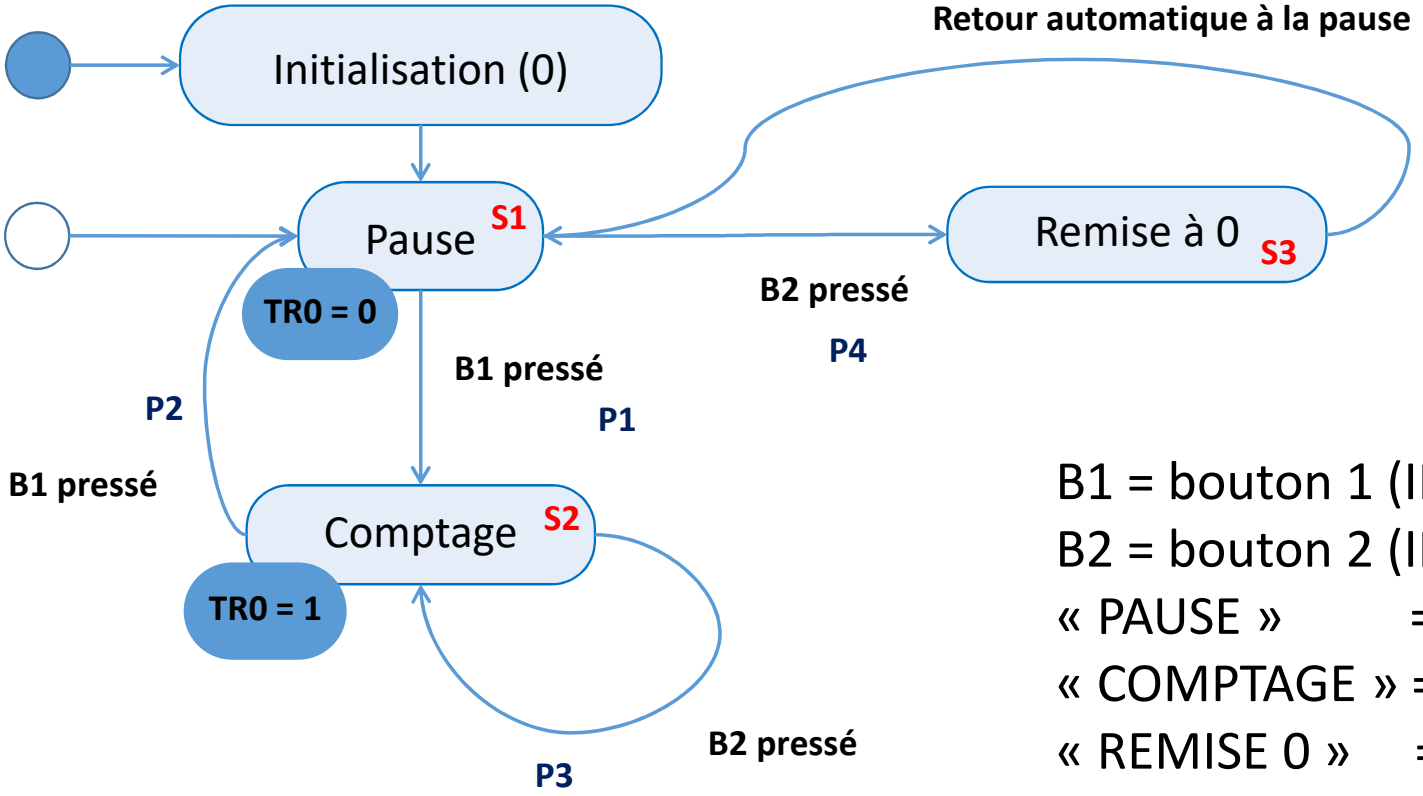
La formalisation des 'States' et des 'Pass' fait gagnier beaucoup de ressources de développement

Pris ici : https://stackoverflow.com/questions/52890834/uml-state-for-vending-machine-branches-dilemma

Diagramme « machine à états » de sablier

Observez la fonctionnalité de votre chronomètre de smartphone





B1 = bouton 1 (INT0)
B2 = bouton 2 (INT1)
« PAUSE » = ETAT 1
« COMPTAGE » = ETAT 2
« REMISE 0 » = ETAT 3

TP3.3 : Façon ‘pro’ de programmer une machine à état

3.3.5.1 Jump Tables The JMP @A+DPTR instruction supports case-dependent jumps for jump tables. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the accumulator. Typically the DPTR is loaded with the address of a jump table, and the accumulator acts as an index. If, for example, five "cases" are desired, a value from 0 through 4 is loaded into the accumulator and a jump to the appropriate case is performed as follows:

```
MOV   DPTR,#JUMP_TABLE
MOV   A,#INDEX_NUMBER
RL    A
JMP   @A+DPTR
```

Remplacer par ‘ETAT’ sans le ‘#’

The RL A instruction above converts the index number (0 through 4) to an even number in the range 0 through 8 because each entry in the jump table is a 2-byte address:

```
JUMP_TABLE:  AJMP CASE0
             AJMP CASE1
             AJMP CASE2
             AJMP CASE3
```

État #1, État #2…

Page 74

Another application of this instruction is in "greater than" or "less than" comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, the carry flag is set. If the first is greater than or equal to the second, the carry flag is cleared. For example, if it is desired to jump to BIG if the value in the accumulator is greater than or equal to 20H, the following instructions could be used:

```
CJNE A,#20H,$+3
JNC  BIG
```

The jump destination for CJNE is specified as "$+3." The dollar sign ($) is a special assembler symbol representing the address of the current instruction. Since CJNE is a 3-byte instruction, "$+3" is the address of the next instruction, JNC. In other words, the CJNE instruction follows through to the JNC instruction *regardless* of the result of the compare. The sole purpose of the compare is to set or clear the carry flag. The JNC instruction decides whether or not the jump takes place. This example is one instance in which the 8051 approach to a common programming situation is more awkward than with most microprocessors; however, as we shall see in Chapter 7, the use of macros allows powerful instruction sequences, such as the example above, to be constructed and executed using a single mnemonic.

Annexe

If (a ≥ b) {} else {} en Assembleur (utilisable pour machine à état)

```
COMPARER:
        mov         A, #variable_a
        clr         C ; C = 1, si a – b < 0
        subb        A, #variable_b
        jnc         LABEL_IF
        CODE ELSE….
        ……; code de ‘else’ précède le code de ‘if’
RET

LABEL_IF: ….
RET

_____________________
CALL COMPARER ; pour appeler la fonction
```

Bon courage !

Au supplément du compte-rendu vous devez m'envoyer les projets entiers Keil µVision :

le dossier entier compressé incluant

les fichiers *.uvproj et *.a