Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 76 (1985)

Heft: 15

Artikel: Automate de contrôle de production

Autor: [s.n.]

DOI: https://doi.org/10.5169/seals-904653

Automate de contrôle de production

La programmation en temps réel peut aussi se faire en langage évolué, à condition que le hardware traite les tâches critiques. La réalisation décrite dans cet article concrétise ce fait.

Die Programmierung in Echtzeit kann auch mit einer höheren Computersprache erfolgen, wenn die zeitkritischen Aufgaben mittels Hardware bearbeitet werden. Dies wird am Beispiel eines Prüfautomaten für elektronische Module gezeigt.

1. Introduction

Dans tous les appareils intégrant un système informatique, le choix entre matériel et logiciel dépend fortement des buts à atteindre. Les solutions possibles sont en corrélation directe avec les critères choisis (coût, vitesse, fiabilité, etc.). Des performances temporelles élevées entraînent obligatoirement un matériel important et un logiciel relativement «léger», tandis qu'un prix minimum impose une programmation très importante.

Le choix entre matériel et logiciel se pose aussi lorsque le temps de développement d'un prototype doit être court. La réalisation décrite ci-dessous a été construite à un seul exemplaire. Cela implique une série de compromis qui seront discutés.

2. Cahier des charges

L'appareil à réaliser contrôle automatiquement des modules électroniques sortant d'une chaîne de fabrication. Il doit vérifier un certain nombre de paramètre électriques et temporels, tenir une statistique des défauts constatés ainsi qu'une fiche signalétique. Le schéma bloc de cette installation est décrit à la figure 1. On dispose d'un ordinateur pesonnel que le client possède en stock, une interface spécialisée permettant au programme d'application de mesurer les paramètres du module au travers de la tête de mesure.

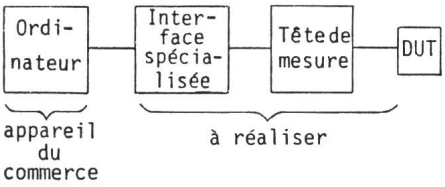

Fig. 1 Schéma bloc de l'automate de test
DUT = *D*evice *U*nder *T*est (module à essayer)

2.1 Description du module à contrôler

Le module à vérifier est un contrôleur d'affichage «intelligent» destiné aux PTT (figure 2). Il permet la visualisation sur un LCD[1] du numéro de téléphone que l'on est en train de taper. Dès que la communication est établie, le temps de celle-ci s'affiche en minutes et secondes, ainsi que son coût. Ce module se présente sous la forme d'un parallélépipède de 78×20 mm et de 15 mm de profondeur. Il comprend un circuit intégré, un circuit imprimé et un connecteur. Sa fabrication demande les étapes suivantes:

1) Coller le circuit intégré livré sous forme de puce sur le circuit imprimé.
2) Relier cette puce sur le circuit imprimé avec la technique du «bonding», technique consistant à souder par ultra-sons un fil microscopique partant de la puce et aboutissant sur une piste du circuit imprimé.
3) Recouvrir cette puce d'une résine que l'on polymérise à haute température.
4) Sertir les contacts du connecteur.
5) Tester fonctionnellement l'ensemble.

[1] LCD = Liquid-Crystal-Display

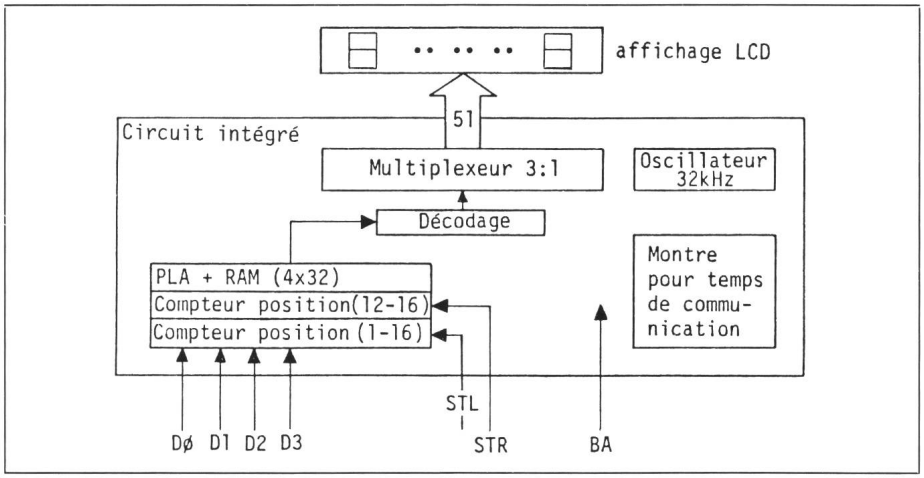

Fig. 2 Schéma bloc du circuit intégré du module à tester

Adresse des auteurs

Claude Balmer, ing. él. dipl. EPFL, Balmer Informatique Industrielle, 22, ch. des Baumettes, 1008 Prilly.
Salomon Cohen, ing. él. dipl. EPFL, CFG S.A., 2, av. de Lonay, 1110 Morges.

Le circuit intégré utilisé peut directement piloter un afficheur LCD de 16 digits de 7 segments, soit 112 segments. Pour diminuer le nombre de connections, un multiplexage de rapport 3:1 est utilisé (voir les figures 3 et 4 et le texte encadré). Il ne reste que 48 fils plus 3 fils communs appelés common 1 (COM1), common 2 (COM2) et common 3 (COM3).

L'interface entre le circuit et le monde extérieur est simple. Un bus de data de 4 bits est disponible avec 3 signaux de contrôle (fig. 2) STL, STR et BA. STL échantillonne et mémorise l'état du data bus D0 à D3 et le visualise à gauche dans l'affichage, STR le fait à droite. BA fait le reset du circuit.

2.2 Fonctions de l'automate de test

L'automate a trois modes de fonctionnement: le mode production, le mode simulation manuelle et le mode autodiagnostique.

Mode production

Pour garantir une cadence de test raisonnable et une haute fiabilité des contrôles, aucun test visuel n'est fait. Tout est contrôlé par l'automate, plus particulièrment:

- la consommation maximum et la plage de tension de fonctionnement du module,
- le fonctionnement de la logique d'interface (signaux STL, STR, BA, D0 à D3),
- le bon fonctionnement des signaux pour l'afficheur LCD,
- le bon fonctionnement de l'horloge.

Dans ce mode, le programme tient à jour une statistique du lot. Le numéro de celui-ci et sa date sont mémorisés, le nombre de pièces est comptabilisé, ainsi que le nombre de pièces défectueuses et leur type de panne.

Le contrôle d'un module se fait ainsi: une opératrice prend une pièce, la pose sur la «planche à clou» de la tête de mesure, puis ordonne le démarrage du test. Les contacts électriques se font grâce aux petites aiguilles retractiles de la «planche à clou» dans la tête de mesure. L'ordinateur affiche le résultat sous forme Bon–Mauvais. Si la pièce est bonne, l'opératrice en prend une autre et démarre un nouveau test. Dans le cas contraire, elle a le choix de recommencer le test ou de jeter le module défectueux dans une boîte correspondant au type de panne détectée.

Mode simulation

L'opératrice peut vérifier une pièce de façon «manuelle». Sur le moniteur TV sont affichés en temps réel l'état de l'affichage, le data qui se trouve sur D0–D3, et l'état de BA, STR et STL. Elle peut forcer un reset du module avec BA, un décalage à gauche avec STL, ou à droite avec STR, comme elle peut changer le data à envoyer. Les routines utilisées dans ce mode de fonctionnement sont les mêmes que pour le mode production. Cela permet de garantir le bon fonctionnement de celles-ci.

Ce mode est très utile pour affiner les routines de test et pour vérifier le comportement réel d'un module défectueux.

L'affichage LCD

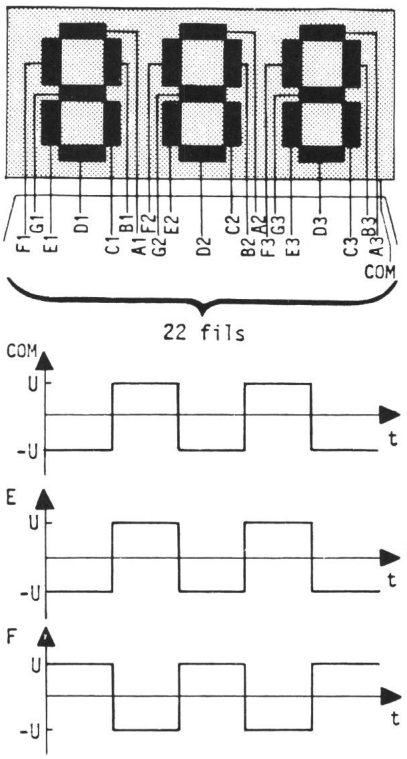

Fig. 3 Multiplexage 1:1

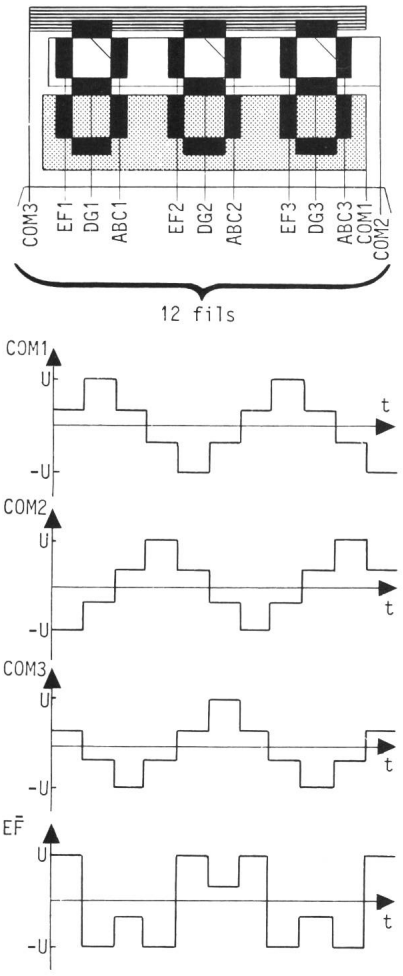

Fig. 4 Multiplexage 3:1

Un affichage à cristaux liquides est équivalent à une matrice dont les intersections des lignes et des colonnes forment les points qui seront visibles. Soit l'affichage de 3 digits à 7 segments de la figure 3. Le fil appelé «COM» est commun à tous les segments et correspond aux lignes de la matrice, les colonnes sont les fils allant vers les segments. Les tensions appliquées sur chaque fil sont périodiques et à valeur moyenne nulle, cela pour éviter une électrolyse du cristal liquide. Cette période est appelée temps de rafraîchissement. Pour qu'un segment soit visible, il faut que la différence de tensions entre COM et le segment choisi soit plus grande que zéro. La figure 3 montre le segment E visible et F invisible, car «COM-E» produit un signal rectangulaire d'amplitude ±2U, tandis que «COM-F» donne un signal nul.

Lorsque le nombre de points à afficher augmente, il devient intéressant de faire diminuer le nombre de fils de connection au moyen d'un multiplexage du signal. Dans le cas d'un multiplexage 3:1, on code trois informations de segment sur un fil (fig. 4). Il ne reste plus que 12 fils de connection, contre 22 précédemment. L'état des segments E et F est transmis sur le fil appelé EF. Physiquement sur le verre, E se trouvera à l'intersection entre l'électrode COM1 et EF, tandis que le segment F se trouvera entre COM2 et EF. Si on fait la différence des tensions COM1 avec EF et COM2 avec EF, un segment est allumé lorsque celle-ci dépasse un certain seuil (sur l'exemple 4/3 de U). L'exemple montre la tension, appliquée à EF pour que E soit visible et F invisible.

Cette méthode a l'avantage de réduire considérablement le nombre de fils de connection, par là d'abaisser le prix de revient du produit. Malheureusement, plus le taux de multiplexage augmente, plus le contraste de l'affichage diminue (il est inversement proportionnel au seuil de tension pour discriminer l'état visible de l'état invisible). Enfin, pour corser le tout, plus le taux de multiplexage est élevé, plus l'affichage deviendra sensible à la température. Comme dans toute application technique, un compromis doit être trouvé.

Mode autodiagnostique

Le personnel chargé de la maintenance peut visualiser sur un oscilloscope tous les signaux que génère l'automate de test. L'opérateur choisit la fonction à contrôler dans un menu, et le programme lui indique les signaux qu'il doit vérifier.

3. Réalisation

Dans un projet comme celui-ci, la facilité d'utilisation et la maintenance du système informatique sont recherchées. Le temps mis pour développer ce prototype doit être aussi court que possible. Cela conduit naturellement à utiliser un langage de haut niveau. Une version améliorée de Pascal a été utilisée. Elle comprend entre autres deux nouveaux types de données (byte et word), des instructions pour la manipulation des entrées-sorties (inp et out), de bit et de byte (hi, lo, clrbit, setbit, testbit, shl, shr, swap). Ces facilités simplifieront la commande de l'interface spécialisée. Il n'y aura plus besoin d'avoir recours à l'assembleur, toujours long à mettre au point. Tout peut être écrit en Pascal.

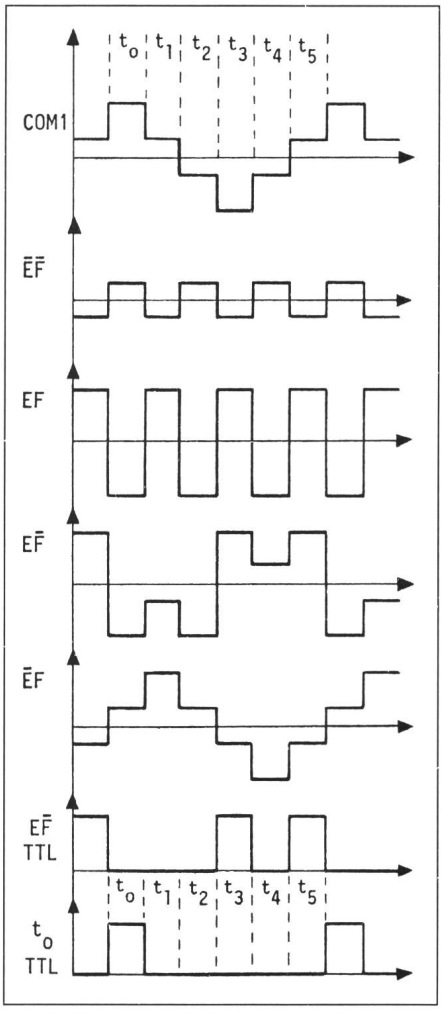

Fig. 5 Signaux du DUT - signaux TTL

Le temps que prend une instruction Pascal n'est pas connu. C'est le défaut que l'on trouve dans (presque) tous les compilateurs. Cela impose une condition supplémentaire à l'interface à construire: toutes les contraintes temps réel doivent être résolues au niveau de l'interface uniquement.

Pour cette application, le véritable problème temps réel est le décodage de l'affichage, qui est raffraîchi toutes les 11,6 ms (voir encadré). En une période, tous les segments doivent être décodés et mémorisés. Ce problème est difficile à résoudre avec un programme écrit en assembleur, et pratiquement impossible pour un langage de haut niveau. Le hardware doit réaliser ce décodage.

3.1 Décodage de l'affichage

La figure 5 donne les caractéristiques de l'affichage pour les segments EF en fonction de COM1. On a représenté, par rapport au signal COM1, les quatres signaux que peut générer le fil EF; $\overline{EF}$ représente les segments E et F éteints (invisibles), EF les deux allumés (visibles), $E\overline{F}$ et $\overline{E}F$ l'un ou l'autre allumé. En mettant en forme les signaux analogiques et en les seuillant avec des amplificateurs opérationnels, on obtient les signaux logiques $E\overline{F}$TTL et t_0 TTL de la figure 5. Pendant le temps t_1 on peut prédire si le segment F est allumé ou éteint et pendant t_5 le segment E. En généralisant ce principe aux 51 fils du module, on a:

- pendant t_1, les segments B, F et G sont lus (et décodés)
- pendant t_3, le segment C est lu (et décodé)
- pendant t_5, les segments A, E et D sont lus (et décodés).

Cette méthode a l'avantage de n'utiliser qu'un seul type d'amplificateur opérationnel avec un seuil de commutation commun pour tous les segments. La figure 6 représente le schéma bloc de la tête de mesure. On a, à gauche, les signaux analogiques provenant du module sous test, à droite, les signaux logiques allant vers l'interface spécialisé.

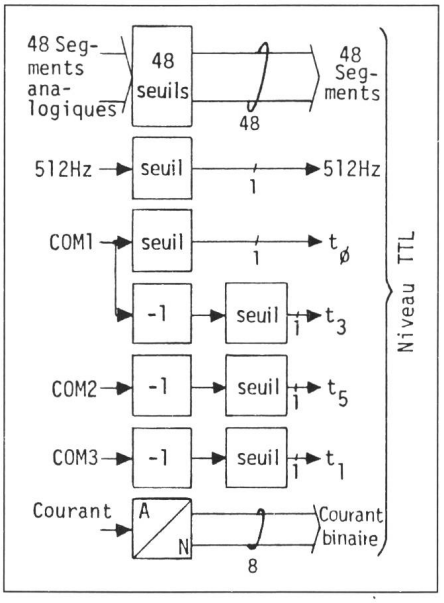

Fig. 6 Schéma bloc de la tête de mesure

3.2 Unité de traitement

Pour faire face aux modifications qui viendront suite à l'expérience acquise au cours des tests, une solution en logique programmée est choisie [1]. L'automate se décompose en deux machines distinctes: l'unité de traitement (fig. 7) et l'unité de commande (fig. 8) [2]. Cette solution permet au programme Pascal de s'affranchir des contraintes «temps réel» du décodage de l'affichage.

L'unité de traitement décode les digits de l'affichage. La valeur des segments choisis par les lignes A et B (fig. 7) est mémorisée par CP dans le registre à décalage RD. L'ordinateur peut y lire une valeur correctement codée grâce à l'EPROM de transcodage. Le registre R1 choisit le transcodage voulu, tandis que R2 choisit le digit à lire. Pour décoder l'affichage, la séquence suivante doit être programmée dans l'unité de commande de la figure 8:

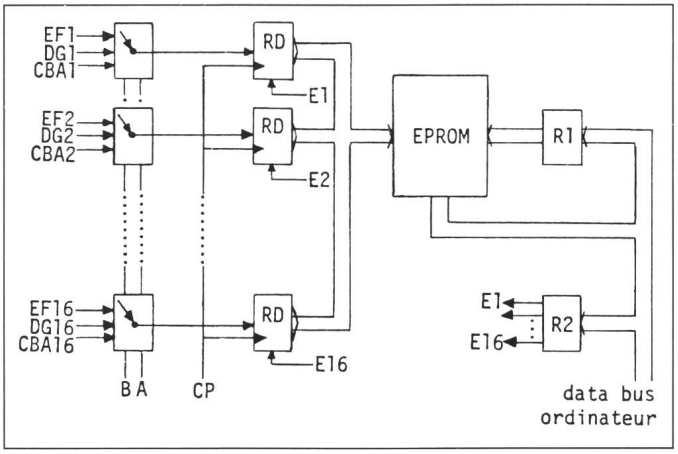

Fig. 7
Décodage des digits par l'unité de traitement
RD registre à décalage
Ri registre (i)

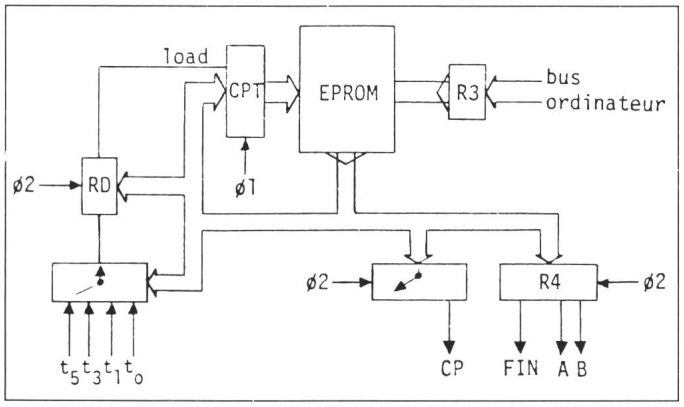

Fig. 8
Micro-séquenceur
RD registre à décalage
CPT compteur
Ri registre
øi horloge système

1) attendre t_0 pour se synchroniser,
2) sélectionner l'entrée EFi du MUX et attendre t_1,
3) faire une impulsion CP; le segment E est mémorisé,
4) sélectionner l'entrée DGi du MUX,
5) faire une impulsion CP; le segment D est mémorisé,
6) sélectionner l'entrée CBAi du MUX,
7) faire une impulsion CP; le segment C est mémorisé,
8) sélectionner l'entrée CBAi du MUX et attendre t_3,
9) faire une impulsion CP; le segment A est mémorisé,
10) sélectionner l'entrée EFi du MUX et attendre t_5,
11) faire une impulsion CP; le segment F est mémorisé,
12) sélectionner l'entrée DGi du MUX,
13) faire une impulsion CP; le segment D est mémorisé,
14) sélectionner l'entrée CBAi du MUX,
15) faire une impulsion CP; le segment B est mémorisé,
16) sélectionner l'entrée mise à zéro du MUX,
17) faire une impulsion CP; 0 est mémorisé en bit 7,
18) signaler au processeur avec la ligne FIN que la conversion est terminée.

Cet algorithme se microprogramme facilement. A l'entrée de l'EPROM de transcodage, on se retrouve avec l'information suivante:

D7	D6	D5	D4	D3	D2	D1	D0
O	B	G	F	A	C	D	E

De cette manière, un affichage complet se décode dans un temps inférieur à 22 ms!

3.3 Unité de commande

La figure 8 montre l'unité de commande générant les signaux CP, A et B de la figure 7. Le matériel d'aide au développement existant sur le marché travaille généralement sur des mots de 8 bits. Cela explique le format 8 bits pour les instructions. L'unité de la figure 8 à l'avantage d'utiliser relativement peu de matériel au prix d'un multiplexage des instructions dans le temps [3]. Le compteur CPT sert à balayer les adresses de la mémoire l'une après l'autre. Une nouvelle adresse peut être chargée en fonction de la condition (t_0, t_1, t_3 ou t_5) choisie sur le multiplexeur. Les instructions ont un temps de cycle de 2 µs, qui est amplement suffisant pour cette application.

Avec ce type de séquenceur, les instructions suivantes sont disponibles:

D7	D6	D5	D4	D3	D2	D1	D0

Avec D7 = 1 on a

D6	05	description
0	0	impulsion CP de durée égale à «phi» contrôlée par D0–D2
0	1	chargement du registre R4 avec D0–D3
1	0	test du bit t_0 à t_3 choisi par D0–D2
1	1	pas d'opération (NOP)

Avec D7 = 0, on a un saut conditionnel. L'adresse est codée de D6 à D0. Pour avoir une instruction de saut selon l'état d'une des variables t_0 à t_5, il faut programmer les deux instructions suivantes:

1100 0xxx teste entrée t_i (i = en binaire, de 0 à 7)
0*nnn nnnn* saut si t_i = 1 à l'adresse n (0 à 127)

4. Conclusion

Cette application décrit comment la programmation peut être déchargée de la tâche de décodage en temps réel par une réalisation matérielle adéquate (interface spécialisée). Si le nombre d'appareils à réaliser est élevé, cette solution est trop coûteuse. Dans ce cas, un microprocesseur programmé en assembleur se justifierait.

Les problèmes relatifs au temps réel avec des langages évolués tels que Pascal peuvent facilement se résoudre dans le matériel, à condition de bien étudier le problème.

Bibliographie

[1] *J. Florine:* Logique transitionnelle microprogrammée. Paris, Inter Editions 1983.
[2] *D. Mange, E. Sanchez* et *A. Stauffer:* Systèmes logiques programmés. Lausanne, Presses polytechniques romandes 1982.
[3] Un système microprogammable de gestion de processus. Electronique Applications -(1981)20, p. 55...59.