Het ontwerpen van een data-logger
digitale deel

Geurts, A.G.M.

Award date:
1963

HET ONTWERP VAN EEN DATA-LOGGER

DIGITALE DEEL

Afstudeerverslag van de Heer A.G.M. Geurts.

HET ONTWERP VAN EEN DATA-LOGGER VOOR INDUSTRIELE TOEPASSINGEN

DIGITALE DEEL

Het ontwerp is tot stand gekomen in de afdeling Electrotechniek, sectie Telecommunicatie EC-B van de Technische Hogeschool te Eindhoven onder leiding van Prof.ir.A. Heetman gedurende de periode november 1962 - maart 1963.

SAMENVATTING.

In dit verslag wordt het ontwerp van het digitale deel van een data-logger gepresenteerd. De opdracht was te komen tot een ontwerp van universele opzet met een zeer grote mate van flexibiliteit. Aan de meetzijde kan door een groot aantal, door het digitale deel te besturen eenheden elk gewenst type gelijkstroom- of gelijkspanningsmeetzender aangesloten worden. In het digitale deel kan de van de analoge schakeling verkregen informatie op velerlei wijze verwerkt worden. Hiertoe is het digitale deel van de data-logger uitgerust met een groot kerngeheugen. In dit geheugen worden de gegevens van elk meetpunt vastgelegd, alsmede de bewerkingen die de informatie van een meetpunt moet ondergaan. Op eenvoudige wijze is het daarbij mogelijk de gegevens der meetpunten te wijzigen.

INHOUDSOPGAVE.

Bijvoegsels:

1. 10 overzichtschema's.
2. "Bidec, A Binary-to-Decimal or Decimal-to-Binary Converter", J. F. Couleur, IRE Trans. on Electronic Computers, 1958, pp. 313 + 316.

LITTERATUUR.

A. v. Balen: Het ontwerp van een data-logger voor Industriele toepassingen - Analoge deel.

A. A. M. Gall: Een keuzeschakelaar voor ruim 500 meetpunten van een data-logger.

Zie voorts de afzonderlijk verschenen litteratuurlijst.

1. INLEIDING.

Aan de hand van een algemeen overzicht en een meer gedetailleerd schema van het digitale deel der data-logger zal de werking van de digitale eenheid nader onderzocht worden (figuur 1.1.).

De digitale eenheid zendt bij het begin van de meting gegevens door naar het analoge deel. Het eerste gegeven is het nummer van het te meten kanaal. Deze informatie wordt toegevoegd aan de scannermatrix ter selectie van het meetkanaal. De scanner koppelt het meetkanaal door middel van relais aan het overige deel van het analoge circuit. Het meetsignaal wordt hierna versterkt en in digitale code omgezet. Aanvullende informatie betreffende het geselecteerde meetkanaal gaat naar de analoge aanpassingsapparatuur. Dit deel van het circuit draagt zorg voor een juiste interpretatie van het aangesloten meetsignaal (mA-stromen, mV-spanningen, weerstandswaarden en contactstanden; bij thermokoppels kan voorzien worden in een compensatie van de koude las). Aan de versterker en de analogue-to-digital converter (A.D.C.) worden gegevens verstrekt omtrent de juiste versterkings- en schaalfactoren. De voor digitale verwerking vereiste informatie komt aan het eind van de meetcyclus vrij aan de uitgangsklemmen van de A.D.C.

Naast de invoer van informatie via de A.D.C. bestaat de mogelijkheid gegevens via de ponsband of met behulp van keuzeschakelaars in de digitale eenheid te brengen. De digitale eenheid verwerkt de binnengekomen gegevens, bewaart deze gegevens en de resultaten voor zover nodig in zijn geheugen en neemt op grond van deze resultaten beslissingen.

De output van de digitale eenheid wordt gevormd door de eerder genoemde registers in het analoge deel der datalogger. Voorts kunnen gegevens en resultaten getypt worden op een typemachine of stripprinter. Voor bepaalde doeleinden kan ook een ponseenheid aangesloten worden.

Het aansluiten der meetkanalen geschiedt volgens een tevoren in het geheugen van de digitale eenheid vastgelegd programma. De volgorde der meetpunten in dit programma is willekeurig te kiezen.

De opbouw van het digitale deel is zeer schematisch weergegeven in figuur 1.2. Het digitale deel bestaat uit de volgende, functioneel met elkaar verbonden onderdelen:
1. de input- en outputorganen;
2. het kerngeheugen;
3. de controle-eenheid;
4. het arithmetische deel.
Deze onderdelen zullen aan een nadere beschouwing onderworpen worden.

1.1. De input- en outputorganen.

De inputorganen zijn de eerder genoemde en getekende (figuur 1.1.) A.D.C., de ponsbandlezer en de keuzeschakelaars. De output wordt gevormd door registers in het analoge deel der data-logger voor het bepalen en aansluiten der meetpunten; voorts kan een typemachine, een stripprinter en indien nodig, een ponseenheid aan het digitale deel der data-logger aangesloten worden. De gegevens der metingen worden getypt in de eenheden der grootheden die zij voorstellen. (Bij thermokoppels b.v. wordt de waarde van de temperatuur in graden C. geprint in plaats van de bijbehorende gemeten waarde der spanning.)

1.2. Het Kerngeheugen.

Het kerngeheugen (figuur 1.3.) bestaat uit de eigenlijke kernenmatrix, de X- en Y-adresselectoren en een bufferregister. Het kerngeheugen is woordgeorganiseerd. Dit wil zeggen, dat gedurende het lezen of schrijven in het geheugen steeds een rij kernen in de opeenvolgende vlakken via de X- en Y-keuzedraden geactiveerd wordt. De informatie wordt bij uitlezen uit het geheugen parallel via de leesdraden aangeboden aan de buffer, ofwel bij schrijven via de buffer en de verbodsdraden parallel in het geheugen ingevoerd.

In het geheugen zijn ondergebracht de gegevens die bij een meetpunt behoren. In de eerste plaats zijn dit de gegevens die aan het analoge deel gepresenteerd worden. Voorts zijn gegevens vastgelegd die de aard der bewerkingen vaststellen alsmede de constanten die bij deze bewerkingen nodig zijn (b.v. alarmniveau's en differenties). Tenslotte wordt nog aangegeven hoe de digitale informatie aan de output-organen aangeboden wordt.

In het geheugen is een lijst van opdrachten opgenomen, een programma, dat zorg draagt voor het successief aansluiten der meetkanalen en verwerken van de meetresultaten. De volgorde der te meten punten is tevens tevoren in het geheugen ingebracht. De opdrachten of instructies van het programma zijn in gecodeerde vorm vastgelegd in een woord van het geheugen. De instructie bestaat uit twee gedeelten, een operatie, welke de bewerking specificeert, en een adres waar de bij deze operatie benodigde informatie is opgeslagen. Dit adres kan betrekking hebben op een geheugenplaats, maar kan ook het adres zijn van een register in het analoge of digitale deel van de datalogger. De woorden van een programma staan na elkaar in het geheugen, d.w.z. de eerste instructie van een programma staat b.v. op geheugenplaats 83, de volgende instructie op geheugenplaats 84 etc. Deze woorden worden na elkaar gelezen en geïnterpreteerd volgens de erbij gegeven, in gecodeerde vorm gebrachte opdrachten. Het programma wordt nader beschouwd in hoofdstuk 4, enkele programma-onderdelen komen ter sprake in hoofdstuk 3.

Linearisatietabellen zijn in het geheugen opgenomen voor die meetpunten waarbij tussen de gemeten waarde van het door de meetzender afgegeven signaal en de gemeten grootheid geen lineair verband bestaat (bij thermokoppels b.v. is er geen evenredig verband tussen de gemeten temperatuur en de daarbij afgegeven spanning). Een linearisatieprogramma wordt besproken in hoofdstuk 3.3.

Het geheugen bezit een capaciteit van 4000 woorden, elk bestaande uit maximaal 25 bits.

1.3. De controle-eenheid.

De controle-eenheid vormt de verbinding tussen het geheugen en het overige deel der data-logger. Deze controle-eenheid kan als een schakelaar gedacht worden, die de verschillende registers en eenheden op het juiste moment met elkaar doorverbindt voor het overgeven van informatie (figuur 1.2.). Deze schakelaar wordt bestuurd door de instructies, zoals deze zijn opgenomen in het programma. In de instructies is slechts een adres gespecificeerd. Dit betekent voor het zeer globale overzicht van figuur 1.2., dat per instructie slechts één van de beide schakelaars een bepaalde positie kan innemen.

De tijd, waarin een instructie van het programma en de interpretatie van deze stap voltooid worden (de operatiecyclus), is verdeeld in twee intervallen, tijdens welke achtereenvolgend de instructie-cyclus en de executiecyclus doorlopen worden. Tijdens de instructie-cyclus wordt een opdracht van het programma gelezen. De in deze instructie, in gecodeerde vorm vastgelegde opdracht wordt via een vertaler in bruikbare vorm omgezet. Tijdens de executie-cyclus wordt de opdracht uitgevoerd met de informatie welke opgeslagen is in het door de instructie gegeven adres.

1.4. De arithmetische eenheid.

Wordt in het programma een rekenkundige bewerking voorgeschreven, dan wordt deze bewerking uitgevoerd in de arithmetische eenheid. Deze eenheid bestaat uit drie registers, een optelinrichting met indicatoren en een binaire teller (zie figuur 1.4.). De rekenkundige bewerkingen zijn: optellen, aftrekken, vermenigvuldigen en delen. De gegevens staan in de registers; na de bewerking is het resultaat wederom in een der registers opgeslagen.

Om moeilijkheden bij het transport van informatie bij een rekenkundige handeling te ontlopen, wordt gebruik gemaakt van het buffer-register van het geheugen. Bij optellen b.v. zal de som in een der registers opgeslagen worden, waar tevens het opteltal geregistreerd is. Om het overlappen van gegevens te voorkomen wordt de som van de optelling tijdelijk in het bufferregister vastgelegd en na afloop van de optelling naar een der registers in het arithmetische deel teruggebracht. In hoofdstuk 3.1. zal nader worden ingegaan op de nauwkeurigheid der berekeningen en het afronden van de resultaten.

Het is voorts mogelijk de inhoud van twee registers met elkaar te vergelijken. Deze vergelijking wordt uitgevoerd door middel van aftrekken waarbij het verschil verloren gaat. Het resultaat van de vergelijking wordt aangegeven door de positief- en nul-indicatoren.

De mogelijkheid bestaat om bij de inhoud van een der registers 1 (eenheid) op te tellen of af te trekken; de inhoud van dit register kan met 0 vergeleken worden zonder dat de inhoud der overige registers aangetast wordt. In hoofdstuk 2 zijn de instructies in detail uitgewerkt.

Voor het digitale deel is de voorkeur uitgegaan de gegevens te verwerken volgens de zuiver binaire code. Het voordeel is dat de rekenkundige bewerkingen een eenvoudige gedaante krijgen. Hiertegenover staat echter het nadeel van de omzetting van de gegevens in zuiver binaire vorm naar binair-decimale vorm. Een conversie-schakeling wordt gegeven in hoofdstuk 3.4.

De meetresultaten worden via de A.D.C. in de gewenste code gepresenteerd; de woordlengte bedraagt 10 bits, zodat een nauwkeurigheid van verwerking der meetgegevens van 1 : 1024 of van 0,1% bereikbaar is.
De adresselectie van het kerngeheugen (4000 woorden)

vergt 12 bits. Ter onderscheiding van geheugen-adressen en adressen van de output-registers in het overige deel der data-logger is aan het adresveld van de instructies een extra adresbit.

In het programma wordt naast de adresverwijzing in een instructie nog de operatie-code gegeven. Momenteel zijn er ongeveer 20 verschillende instructies. De binaire code van deze instructies vraagt dus minstens 5 bits. De totale woordlengte van een instructiestap in het programma wordt dan 18 bits. Voordelen zijn verbonden aan de keuze van een woordlengte van 20 bits, zodat twee voor de bewerking der meetresultaten benodigde gegevens (elk een lengte van 10 bits) of informatie van meer dan 10 bits lengte in een geheugenwoord ondergebracht kunnen worden.

2. INSTRUCTIES.

2.1. Inleiding.

Voor het samenstellen van een programma zijn de volgende groepen van instructies nodig:

1. arithmetische instructies, voor het verrichten van rekenkundige bewerkingen;
2. interne verplaatsingsinstructies. Door middel van deze instructies wordt informatie getransporteerd naar en van de gewenste adressen;
3. input- en outputinstructies, met het doel informatie in te lezen of te presenteren aan de output-apparatuur;
4. controle-instructies. Deze instructies geven de juiste regulering aan het programma.

In deze onderverdeling is de groep logica-instructies (vergelijk- en springinstructies) niet ondergebracht. Vergelijken geschiedt via een rekenkundige bewerking. De vergelijkingsinstructies zijn daarom ondergebracht in de groep arithmetische instructies.
De springinstructies zijn ondergebracht in de groep contrôle-instructies.

De operatie-cyclus, dit is het geheel van handelingen benodigd voor het volvoeren van een opdracht, bestaat uit twee gedeelten, de instructie-cyclus en de executie-cyclus. Gedurende de instructie-cyclus wordt een instructie uit het kerngeheugen gelezen en wordt de uit te voeren operatie voorbereid. De opdracht die in de instructie besloten is, wordt gedurende de executie-cyclus uitgevoerd met de informatie op de in het adresveld van de instructie aangegeven geheugenplaats.

De werking van het digitale deel zal met verwijzing naar figuur 2.1. tijdens de bespreking der instructies tot in detail beschouwd worden. Uitgangspunt is een woordlengte in het kerngeheugen van 20 bits, 13 voor de

adreskeuze en 7 voor de operatie-code. Daarnaast kan het geheugenwoord in zijn geheel of in twee gedeelten van 10 bits gelezen worden.

De onderdelen en registers van figuur 2.1. zullen hieronder besproken worden. De gebruikte afkortingen betekenen: FF = flip-flop; MSB('s) = meest significante bit(s); LSB('s) = minst significante bit(s).

De geheugenbuffer (buffer) is een 20 FF's lang register. Het register kan in zijn geheel op nul gesteld worden. Input: van kerngeheugen, van reg.Aup (10 MSB's), van reg.Alow (10 LSB's) en van Full Adder (10 MSB's). Output: naar reg.Aup (10 MSB's), naar reg.Alow (10 LSB's) en naar instructieregister (gehele woordlengte).

Register A upper part (reg.Aup) is 10 FF's lang en kan gereset worden. Tesamen met reg.Alow kan de inhoud der registers gecomplementeerd en naar links of rechts verschoven worden. Input: van buffer, party line en van pos-ind. op MSB (vermenigvuldigen). Output: naar buffer, party line, Full Adderen van MSB naar pos-ind. (delen).

Register A lower part (reg.Alow) is 10 FF's lang en kan gereset worden. Tesamen met reg.Aup kan de inhoud van beide registers gecomplementeerd en naar links of rechts verschoven worden. Input: van buffer (apart adresseerbaar), party line en van pos-ind. op LSB (bij delingen). Output: naar buffer, party line en van LSB naar Full Adder (vermenigvuldigen). Een indicator is aangesloten op de MSB van reg.Alow.

Register B (reg.B) is 10 FF's lang. Het register kan gereset worden. Input: van party line. Output: naar party line en Full Adder.

Full Adder. De optelinrichting is 10 posities lang. Input: van reg.Aup, van reg.B en op LSB van de pos-ind. (via het holdcircuit; de carry-bit C_0 wordt hierbij ingevoerd). Output: naar buffer (10 MSB's)en van MSB (overflow) naar indicatoren. Aangenomen wordt, dat in het ongunstigste geval de carry-propagatietijd binnen twee opvolgende klokpulsen ligt.

Counter. Dit is een 4 FF's lange binaire teller. De mogelijkheid tot resetten en presetten (in de positie 0101) bestaat. Input: van party line naar complementaire zijde; voorts kan 1 eenheid opgeteld worden. Output: naar de party line. Een indicator en een programma-alarm zijn aangesloten ter detectie van de stand 1111 (zie ook hoofdstuk 5.).

Indicatoren en Holdcircuit. De positief indicator (pos-ind.) en de nul-indicator (zero-ind.) zijn twee als binaire teller gekoppelde FF's. Het hold-circuit (1 FF) is gekoppeld aan de pos-ind. Mogelijkheid tot resetten en setten bestaat. Input: van Full Adder en van MSB van reg.Aup. Output: van hold-circuit naar Full Adder (zie hoofdstuk 2.2. aftrekinstructie, voor het zogenoemde "Carry-end-around"-mechanisme) en van pos-ind naar MSB van reg.Aup en naar LSB van reg.Alow. Een programma-alarm treedt op als beide indicatoren aan zijn (hoofdstuk 5.).

Instructieregister is 20 FF's lang, 7 bits voor de operatie-code en 13 voor de adreskeuze. Het register kan gereset worden. Input: van buffer. Output van het operatiedeel: naar de operatietranslator. Output van het adresdeel: naar de adresselector van het kerngeheugen, naar de programmateller en naar de adrestranslator voor de overige registers. De 12 LSB's van het adresdeel zijn als binaire teller gekoppeld; bij

dit adresdeel kan 1 opgeteld worden. De 13e adresbit is verbonden aan een programma-alarm (hoofdstuk 2.4. en 5.).

Programmateller. Dit is een 12 FF's lang, als binaire teller gekoppeld register. De mogelijkheid bestaat om 1 bij de inhoud van de programmateller op te tellen en om de teller te resetten. Input: van instructieregister (adresdeel). Output: naar adres-selector van het kerngeheugen.

Analogue-to-digital Converter (A.D.C.), 10 bits lang. Input: uit analoge deel. Output: naar party line. Mogelijkheid tot starten der conversie bestaat.

Registers in het analoge deel. Deze registers kunnen gereset worden. Input: van party line naar scannermatrix (9 FF's), analoge aanpassingsapparatuur (2 FF's), versterker (2 FF's) en ingangsweerstand der A.D.C. (3 FF's).

Print- en typeregisters. Deze drie registers zijn 10 bits lang. De mogelijkheid tot resetten der registers en starten der output-apparatuur bestaat.

Inleesregisters. Deze twee registers zijn 10 FF's lang.De mogelijkheid tot starten van de lees-apparatuur bestaat. Een indicatie is aanwezig voor het "end-of-line"-teken dat bij gegevensinvoer via de ponsband het einde van de boodschap kenmerkt.

Al de genoemde onderdelen zijn via poorten met elkaar verbonden. Het bedienen van een poort gebeurt bij een combinatie van tijd (klokpuls) en operatie-code (en indien nodig adreskeuze). Het transport van informatie tussen registers onderling, in het kern-geheugen en tussen registers en het kerngeheugen gaat steeds via de party line en reg.Aup en/of reg.Alow. In de volgende paragrafen wordt een analyse van de poorten-controle gegeven. Aangenomen is, dat de klok-frequentie 20 KHz bedraagt (pulsduur dus 50 /uSec.).

2.2. Instructie-cyclus.

Elke operatie wordt voorafgegaan door een instructie-cyclus. Deze instructie-cyclus is voor alle operaties dezelfde. De volgende bewerkingen moeten gedurende de drie opvolgende klokpulsen voltooid worden:

Instructie-cyclus. Klokpuls:

1. breng de inhoud van de programmateller naar de adresselector van het kerngeheugen; reset het instructieregister; reset de buffer;
2. lees programmawoord in de buffer;
3. breng de inhoud van de buffer naar het instructieregister; tel 1 op in de programmateller.

De instructie-cyclus duurt 3 klokpulsen (= 0,15mSec.). In de op deze instructie-cyclus volgende klokpulsen wordt de operatie uitgevoerd.

2.3. Arithmetische Instructies.

In de in deze paragraaf gegeven overzichten wordt de inhoud van reg.Aup aangegeven met Au, de inhoud van reg.Alow met Al en de inhoud van reg.B met B; reg.Aup en reg.Alow worden samengevat onder de naam reg.A, evenzo is Au + Al = A; aangenomen is dat Au, Al en B positieve, gehele getallen zijn van maximaal 10 bits lengte. Het product bij vermenigvuldigen en het deeltal bij delen mogen 20 bits lang zijn.
Vooreerst wordt een overzicht van arithmetische instructies gegeven; het volgende overzicht geeft aan welke de inhoud der registers voor en na de bewerking is, tevens is de stand der indicatoren aangegeven; het derde overzicht geeft de verdeling der klokpulsen weer terwijl het laatste schema de voor elke instructie maximaal benodigde tijd geeft.

arithmetische instructies:

instructie	code	bewerking
1. optellen	A	tel Au op bij B; de som wordt vastgelegd in reg.Aup.
2. optellen + 1	AM1	tel 1 (binaire eenheid) op bij Au; de som staat in reg.Aup.
3. aftrekken	S	trek Au van B af; het verschil wordt in reg.Aup vastgelegd. Het teken van het verschil wordt aangegeven door de positief-indicator. Is het verschil positief, dan is de indicator aan; is het verschil negatief of nul, dan is de indicator uit.
4. aftrekken - 1	SM1	trek 1 (binaire eenheid) van Au af; het verschil staat in reg.Aup. De pos-ind. is aan als het verschil positief is. Was Au = 0, dan gaat de zero-ind. aan. Het in reg.Aup vastgelegde resultaat is dan fout.
5. vergelijken	C	vergelijk B met Au. Is B $>$ Au, dan is de pos-ind. aan; is B = Au, dan is de zero-ind. aan; voor B $<$ Au zijn beide indicatoren uit.
6. vergelijk met 0	CMO	vergelijk Au met 0. De beide indicatoren zijn uit als Au $\neq$ 0; de zero-ind. is aan als Au = 0.
7. vermenigvuldigen	M	vermenigvuldig B met A1; het product gaat naar reg.A (en kan dus 20 bits lang zijn).
8. delen	D	deel A (maximaal 20 bits) door B; het resultaat gaat naar reg. Alow.

Bij het volgend overzicht wordt nagegaan welke de inhoud der registers vóór en na de bewerking is; tevens wordt de stand der indicatoren na de bewerking aangegeven. Vóór iedere rekenkundige bewerking worden de indicatoren gereset.

De volgende symbolen worden ingevoerd:
Au + B = S; Au + 1 = S; B - Au = D; Au - 1 = D; D' = |D|;
Al x B = P; A/B = Q(uotient) + R(est).
Met een aangestreept symbool (b.v. $\overline{Au}$) wordt aangegeven dat de inhoud van het betreffende register (reg.Aup) in complementaire vorm gegeven is. De gegevens X, Y en Z staan in die registers, die niet bij een rekenkundige behandeling betrokken zijn

code		reg.Aup	reg.Alow	reg.B[1]	Counter	indicatoren[2] 00	10	01	11
1. A	voor	Au	X	B	Y		S te		
	na	S	X	B	Y	-	groot	fout	fout
2. AM1	voor	Au	X	Y	Z				
	na	S	X	Y	Z	fout	-	S=0	fout
3. S	voor	Au	X	B	Y				
	na	D'	X of $\overline{X}$	B	Y	$D \leqslant 0$	$D > 0$	fout	fout
4. SM1	voor	Au	X	Y	Z				
	na	D	X	Y	Z	fout	$D \geqslant 0$	A=0[3]	fout
5. C	voor	Au	X	B	Y				
	na	$\overline{Au}$	$\overline{X}$	B	Y	$B < Au$	$B > Au$	B=Au	fout
6. CM0	voor	Au	X	Y	Z				
	na	Au	X	Y	Z	$Au \neq 0$	fout	Au=0	fout
7. M	voor	X	Al	B	Y				
	na	P	P	B	1	-	fout	fout	fout
8. D	voor	$\overline{A}$	$\overline{A}$	B	Y				
	na	$\overline{T}$	Q	B	1	fout	-	fout	fout

[1]) De inhoud van reg.B blijft steeds bewaard.
[2]) betekenis 00: pos-ind. uit; zero-ind. uit;
10: pos-ind. aan; zero-ind. uit;
01: pos-ind. uit; zero-ind. aan;
11: pos-ind. aan; zero-ind. aan. Deze laatste stand geeft steeds een programma-alarm aan ("fout").
[3]) het resultaat D in reg.Aup is in dit geval fout (het werkelijk vastgelegd resultaat is $\overline{D} + 1$).

In de volgende overzichten wordt aangegeven hoe het microprogramma verloopt tijdens de executie-cyclus der arithmetische instructies. Enige uitleg en voorbeelden zijn toegevoegd (de capaciteit der registers in de voorbeelden is 3 bits). De positief-indicator wordt bij sommige instructies gebruikt ter detectie van een overflow van de Full Adder.

1. optellen (A). klokpuls:
   1. reset indicatoren; reset buffer;
   2. tel op B + Au;
   3. tel op B + Au; transporteer resultaat Full Adder naar buffer;
   4. reset reg.Aup;
   5. transporteer inhoud buffer naar reg.Aup;
   6. einde der operatie.

De stap 2 is toegevoegd om voor het ongunstigste geval te garanderen, dat de carry-propagatie binnen de vastgestelde tijd verloopt.

voorbeeld a, correcte optelling.

tel op B (010) + Au (011) = S (101).

B = 0 1 0 (2)
Au= 0 1 1 (3) +

het resultaat is S = 1 0 1 (5)

De capaciteit van de Full Adder wordt niet overschreden (stand der indicatoren 00).

voorbeeld b, foutieve optelling.

tel op B (011) + Au (110) = S (1001).

B = 0 1 1 (3)
Au = 1 1 0 (6) +

de som is S = 1 0 0 1 (9)

Een overflow treedt op. Hiermee wordt aangegeven, dat de capaciteit van de Full Adder overschreden wordt. De stand der indicatoren is 10. Het in reg.Aup geregistreerde resultaat is nu 0 0 1 (1).

De pos-ind. is hier als overflow-indicator gebruikt.

2. Optellen + 1 (AM1). klokpuls:

1. reset indicatoren; reset buffer;
2. set pos-ind. en holdcircuit;
3. tel op Au + C_o;
4. tel op Au + C_o; transporteer resultaat Full Adder naar de buffer;
5. reset reg.Aup;
6. transporteer de inhoud van de buffer naar reg.Aup;
7. einde der operatie.

C_o is een carry op de minst significante positie van de Full Adder. Het optellen van deze carry gebeurt via het holdcircuit.

Voorbeeld a, correcte optelling.

tel op Au (010) + C_o (1) = S (011).

Au	=	0	1	0	(2)
C_o	=			1	(1) +
het resultaat is S	=	0	1	1	(3)

Dit resultaat is goed. De stand der indicatoren is 10.

Voorbeeld b, foutieve optelling.

tel op Au (111) + C_o (1) = S (1000).

Au	=	1	1	1	(7)
C_o	=			1	(1) +
de som is S	=	1	0	0	0 (8)

De overflow verandert de stand der indicatoren is 01, de zero-ind. is aan. Het in reg.Aup geregistreerde resultaat is 0 0 0 (0).

3. Aftrekken (S). klokpuls:

1. reset indicatoren; reset buffer; complementeer de inhoud van reg.A = $\overline{A}$;
2. tel op $\overline{A}$u + B;
3. tel op $\overline{A}$u + B + C_o;
4. tel op $\overline{A}$u + B + C_o; transporteer resultaat Full Adder naar de buffer;
5. reset reg.Aup;
6. transporteer de inhoud van de buffer naar reg.Aup;

7. is pos-ind. aan? - einde der operatie;
8. neem het complement van reg.A;
9. einde der operatie.

Aftrekken geschiedt door complementair optellen; in alle gevallen wordt het complement ten opzichte van $2^n - 1$ genomen (n is het aantal bits in een woord).
Voorbeeld a, verschil positief.
trek af B (011) - Au (001) = D'(010).

	B =	0 1 1	(3)
	$\overline{Au}$ =	1 1 0	($\overline{2}$) +
het verschil is	D = 1|0 0 1		(1)

De capaciteit der Full Adder is overschreden; de overflow doet de pos-ind. aan gaan. Het resultaat van de aftrekking is correct als aan het verschil nog 1 binaire eenheid toegevoegd wordt.

	D =	0 0 1	(1)
	C_o =	1	(1) +
resultaat:	D' =	0 1 0	(2)

Dit verschil is het positieve getal (pos-ind. aan!) in reg.Aup.
Voorbeeld b, verschil negatief.
trek af B (001) - Au (011) = - D' (010).

	B =	0 0 1	(1)
	$\overline{Au}$ =	1 0 0	($\overline{3}$) +
het resultaat is		1 0 1	($\overline{2}$)

De pos-ind. blijft uit, er is immers geen overflow. Het resultaat is op complementeren na correct. Het verschil 0 1 0 (2) is nu in absolute waarde gegeven in reg.Aup. Het teken van het verschil is negatief (pos-ind. is uit).

In het micro-programma zijn de twee mogelijkheden a en b samengebracht, doordat steeds de (al of niet aanwezige) overflowbit via het holdcircuit bij het resultaat van de Full Adder opgeteld wordt. Dit proces heet het "carry-end-around"-mechanisme. Aan de hand van de stand van de pos-ind. wordt het verschil in reg.A vervolgens gecomplementeerd.

4. Aftrekken - 1 (SM1). klokpuls:

1. reset indicatoren; reset buffer; neem het complement van reg.A = $\overline{A}$;
2. set pos-ind. en holdcircuit;
3. tel op $\overline{A}u + C_o$;
4. tel op $\overline{A}u + C_o$; transporteer resultaat Full Adder naar de buffer;
5. reset reg.Aup;
6. transporteer inhoud buffer naar reg.Aup;
7. complementeer reg.A;
8. einde der operatie.

Voorbeeld a, correcte aftrekking.
trek af Au (010) - C_o (1) = D (001).
Ook hier wordt de inhoud van reg.A gecomplementeerd.

$\overline{A}u$ = 1 0 1 $(\overline{2})$
C_o = 1 $(\underline{1})$ +
het resultaat is D = 1 1 0 (1)

Dit resultaat moet nog gecomplementeerd worden: 0 0 1 (1).
Het teken van het verschil is positief (pos-ind. aan).
Voorbeeld b, foutieve aftrekking.
trek af Au (000) - C_o (1) = - D (001).

$\overline{A}u$ = 1 1 1 $(\overline{0})$
C_o = 1 $(\underline{1})$ +
het verschil is 1 | 0 0 0 (1)

De overflow doet de zero-ind. aan gaan. Het in reg.Aup geregistreerde resultaat is na complementeren 1 1 1 (7). Dit resultaat is foutief. De zero-ind. geeft dus aan, dat de inhoud van reg.Aup vóór de aftrekking nul was.
Er is evenwel van uitgegaan, dat bij de aftrekinstructies uitsluitend positieve getallen gebruikt mogen worden.

5. Vergelijken (C). klokpuls:

1. reset indicatoren; complementeer A = $\overline{A}$;
2. tel op $\overline{A}u + B$;
3. is pos-ind. aan? - einde der operatie;
4. set pos-ind. en holdcircuit;
5. tel op $\overline{A}u + B + C_o$;

6. is zero-ind. aan? - einde der operatie;
7. reset indicatoren;
8. einde der operatie.

Voorbeeld a, vergelijking met resultaat B $>$ Au.
Neem aan B = 011 en Au = 010. De vergelijking wordt uitgevoerd door aftrekken d.w.z. complementair optellen.

$$\begin{array}{rcl} B & = & 0\ 1\ 1\ \ (3) \\ \overline{Au} & = & \underline{1\ 0\ 1\ \ (\overline{2})}\ + \\ & & 1\,|\,0\ 0\ 0\ \ (+) \end{array}$$

Het resultaat van de Full Adder (0 0 0) gaat verloren. De overflow doet de pos-ind. aan gaan. Dit betekent het einde der operatie.

Voorbeeld b, vergelijking met resultaat B = Au.
Stel B = 0 1 1 en Au = 0 1 1. De volgende bewerking wordt uitgevoerd:

$$\begin{array}{rcl} B & = & 0\ 1\ 1\ (3) \\ \overline{Au} & = & \underline{1\ 0\ 0\ (\overline{3})}\ + \\ & & 1\ 1\ 1\ (0) \end{array}$$

Er treedt geen overflow op (de situatie van voorbeeld a treedt niet op). Wordt aan het resultaat van de Full Adder een bit op de minst significante positie toegevoegd (C_o), dan wordt het resultaat:

$$\begin{array}{rcl} & & 1\ 1\ 1 \\ C_o & = & \underline{\qquad 1}\ + \\ & & 1\,|\,0\ 0\ 0 \end{array}$$

De overflow doet nu de zero-ind. aan gaan. Dit betekent het einde der operatie. Het resultaat van de Full Adder (0 0 0) gaat verloren.
Voorbeeld c, vergelijking met resultaat B $<$ Au.
Neem aan B = 0 1 1 en Au = 1 0 0.

$$\begin{array}{rcl} B & = & 0\ 1\ 1\ \ (3) \\ \overline{Au} & = & \underline{0\ 1\ 1\ \ (\overline{4})}\ + \\ & & 1\ 1\ 0\ \ (-) \end{array}$$

De situatie van voorbeeld a treedt niet op.
Aan het resultaat wordt de carry C_o toegevoegd:

$$\begin{array}{rcl} & & 1\ 1\ 0 \\ C_o & = & \underline{\qquad 1} \\ & & 1\ 1\ 1 \end{array}$$

In dit geval ontstaat ook de situatie van voorbeeld b niet. Hieruit wordt het resultaat B $<$ Au vastgesteld (beide indicatoren uit).

6. Vergelijken met 0 (CMO). klokpuls:

1. reset indicatoren; complementeer A = $\bar{A}$;
2. set pos-ind. en holdcircuit;
3. tel op $\bar{A}u + C_o$;
4. complementeer $\bar{A}$ = A;
5. is zero-ind. aan? - einde der operatie;
6. reset indicatoren;
7. einde der operatie.

Eenzelfde situatie ontstaat als bij vergelijken (C) en aftrekken (SM1). Gerefereerd wordt aan voorbeeld b der vergelijkingsinstructie (C), waarbij de situatie van voorbeeld a overgeslagen wordt.

7. Vermenigvuldigen (M). klokpuls:

1. reset reg. Aup; preset Counter (stand 0101);
2. reset indicatoren; reset buffer;
3. is Counterstand 1111? - einde der operatie;
4. tel op B + Au op conditie van de minst significante bit van reg.Alow;
5. tel op B + Au op conditie van de minst significante bit van reg.Alow; transporteer resultaat Full Adder naar buffer;
6. reset reg.Aup; tel 1 op bij de Counter;
7. transporteer inhoud buffer naar reg.Aup;
8. schuif inhoud reg.A een plaats naar rechts;
9. breng pos-ind. naar meest significante bit van reg.Aup;
10. spring terug naar klokpuls 2.

Vermenigvuldigen geschiedt door herhaald (10 keer) optellen en schuiven. Een optelling geschiedt dan, als in de LSB van reg.Alow een 1 staat.

Voorbeeld: vermenigvuldig B (1 1 1) met A1 (1 1 0); het resultaat P (1 0 1 0 1 0) staat in reg.A. (7 x 6 = 42) (N.B. bij een registergrootte van 3 bits behoort een 2 FF Counter).

	operatie	stand Counter
A = 0 0 0 1 1 0 B = 1 1 1	optellen (kan niet, LSB = 0)	0 0
A = 0 0 0 1 1 0 B = 1 1 1	schuiven pos-ind. (0) naar MSB van reg.Aup.	0 1
A = 0 0 0 0 1 1 B = 1 1 1	optellen (kan, LSB = 1)	
A = 1 1 1 0 1 1 B = 1 1 1	schuiven pos-ind. (0) naar MSB van reg.Aup.	1 0
A = 0 1 1 1 0 1 B = 1 1 1	optellen (kan, LSB = 1)	
A = 0 1 0 1 0 1 B = 1 1 1	overflow! schuiven pos-ind. (1) naar MSB van reg.Aup.	1 1
A = 1 0 1 0 1 0 B = 1 1 1		

De Counterstand 1 1 geeft het einde der operatie aan. Het resultaat staat in reg.A. De indicatoren zijn beide uit (stand 00).

8. Delen (D). Klokpuls:

1. preset Counter op 0 1 0 1;
2. reset indicatoren; reset buffer;
3. is Counterstand 1111? - einde der operatie;
4. schuif de inhoud van reg.A een plaats naar links; breng MSB van reg.Aup naar pos-ind.;
5. tel op B + $\overline{A}$u;
6. tel op B + $\overline{A}$u op conditie van de pos-ind.;
7. tel op B + $\overline{A}$u op conditie van de pos-ind.; transporteer resultaat Full Adder naar buffer;
8. reset reg.Aup; tel 1 op bij de Counter;
9. breng inhoud buffer naar reg.Aup; breng pos-ind. naar LSB van reg.Alow;
10. spring terug naar klokpuls 2.

Delen geschiedt door herhaald (10 keer) aftrekken en schuiven.
Een aftrekking is dan mogelijk als de pos-ind. aan is.
Het getal in reg.A moet in complementaire vorm aanwezig zijn.
Voorbeeld: deel 35 door 6; 35 (100 011) staat in reg.A in complementaire vorm; 6 staat in reg.B (110); het quotient 5 (101) verschijnt in reg.Alow; de rest 5 (101) staat in complementaire vorm (010) in reg.Aup.

bewerking		pos-ind.	registers	Counterstand
	A		0 1 1 1 0 0	0 0
	B	0	1 1 0	
schuif links, breng MSB naar pos-ind.	A		1 1 1 0 0 X	
	B	0	1 1 0 +	
tel op (kan, pos-ind.gaat aan)		1	← 1 1 0 1	
resultaat	A	0	1 0 1 0 0 1	0 1
schuif links, breng MSB naar pos-ind.	A		0 1 0 0 1 Y	
	B	1	1 1 0 +	
tel op (kan niet, want pos-ind. gaat uit)		0	← 1 0 0 0	
resultaat	A	0	0 1 0 0 1 0	1 0
schuif links, breng MSB naar pos-ind.	A		1 0 0 1 0 Z	
	B	0	1 1 0 +	
tel op (kan, pos-ind. gaat aan)		1	← 1 0 1 0	
resultaat	A	0	0 1 0 1 0 1	1 1

De Counterstand 1 1 geeft aan dat de operatie voltooid is.
De deelinstructie wordt nader besproken in hoofdstuk 3.1.
Daar zal aangegeven worden aan welke beperkingen de deelinstructie onderhevig is.

In het laatste overzicht der arithmetische instructies wordt de tijd gegeven die per operatie vereist is (inclusief de instructie-cyclus). De klokfrequentie is 20 KHz.

1. A	3 + 6 klokpulsen		9 klokp.	= 0,45mSec.
2. AM1	3 + 7 ,,		10 ,,	= 0,50mSec.
3. S	3 + 7 of 9 ,,	max.	12 ,,	= 0,60mSec.
4. SM1	3 + 8 ,,		11 ,,	= 0,55mSec.
5. C	3 + 3 of 6 of 8 klokp.	max.	11 ,,	= 0,55mSec.
6. CMO	3 + 5 of 7 klokp.	max.	10 ,,	= 0,50mSec.
7. M	3 + 1 + 10 x 9 + 2 klokp.		96 ,,	= 4,80mSec.
8. D	3 + 1 + 10 x 9 + 2 ,,		96 ,,	= 4,80mSec.

2.4. Interne Verplaatsingsinstructies.

De verplaatsingsinstructies verzorgen het transport van informatie van reg.Aup naar de overige registers en terug. Via reg.Aup + reg.Alow kan informatie in het kerngeheugen geschreven en weer gelezen worden in reg. Aup of reg.Aup + reg.Alow. Ook het verschuiven van informatie in reg.A behoort tot deze instructies.

Overzicht

instructie	code	adres	bewerking
9. transfer	TF	1X	breng de inhoud van reg.Aup naar register X;
10. transfer back	TFB	1X	breng de inhoud van register X naar reg.Aup;
11. read	RD	0X	lees de inhoud van geheugenplaats X in reg.Aup + reg.Alow;
12. read part	RDP	0X	lees de inhoud van geheugenplaats X in reg.Aup; alleen de linkerhelft van het geheugenwoord wordt gebruikt;
13. write	WR	0X	schrijf de inhoud van reg.Aup+ reg.Alow op geheugenplaats X;
14. schuif rechts	SHR	-	schuif de inhoud van reg.A naar rechts zoveel plaatsen als aangegeven is in de Counter;
15. schuif links	SHL	-	schuif de inhoud van reg.A naar links totdat op de meest significante plaats van reg.Alow een 1 staat; het aantal schuiven wordt in de Counter geregistreerd; het maximaal aantal schuiven is 15.

Het adres in de instructies 9 + 13 is voorzien van een 13e controlebit. Voor de instructies die betrekking hebben op in- of uitvoer van informatie in het geheugen (11 + 13) is deze controlebit een 0; voor de instructies die

betrekking hebben op informatietransport tussen de registers onderling (9 en 10) is de controle bit een 1. Wordt bij een instructie een foutieve adressering gebruikt, dan kan een programma-alarm ingeschakeld worden (hoofdstuk 5).

Bij de schuifinstructies (14) wordt het adresveld niet gebruikt. Door middel van een transferinstructie (9) kan vooraf het aantal schuiven in de Counter vastgelegd worden.

De SHL-instructie wordt bij de deling gebruikt. Het aantal schuiven is maximaal 15. Hiermee wordt voorkomen, dat men door nul deelt. Een programma-alarm kan in dat geval aangesloten worden (zie ook hoofdstuk 3.1. en 5.).

Het adresveld van een instructie moet twee soorten adressen kunnen aangeven: geheugenadressen en adressen van registers in het overige digitale deel. De zuiver binaire code staat bij een adresveld van 12 bits een selectie toe van 1 uit 4096 adressen. Dit is het aantal adressen in het geheugen. Anderzijds is dit adresveld te lang voor de (momenteel) 13 te adresseren registers. Teneinde de adrestranslator bij de instructies 9 en 10 zo eenvoudig mogelijk te houden, correspondeert de positie van een 1 in het adresveld van de instructie met het adres van een register. Er zijn zo doende 12 adressen voor registers beschikbaar. Door een combinatie van de 13 te adresseren registers zijn momenteel 11 van de 12 mogelijke posities bezet: reg.Alow, reg.B, Counter, scannermatrix, één adres voor de overige drie registers in het analoge deel, de A.D.C., drie adressen voor de printers en 2 voor de inputapparatuur. Het is dus mogelijk om b.v. twee outputregisters tegelijkertijd dezelfde informatie te laten verwerken (b.v. typemachine en ponsband).

Ook bij de verplaatsingsinstructies is het verloop gedurende de diverse klokpulsen gegeven.

9. Transfer (TF). klokpuls:
   1. reset reg.X; check 13e adresbit ≠ 1 - programma-alarm;
   2. transporteer inhoud reg.Aup naar reg.X;
   3. einde der operatie.

De inhoud van reg.Aup blijft bewaard.

10. Transfer Back (TFB). klokpuls:
   1. reset reg.Aup; check 13e adresbit ≠ 1 - programma-alarm;
   2. transporteer inhoud reg.X naar reg.Aup;
   3. einde der operatie.

De inhoud van reg.X blijft bewaard.

11. Read (RD). klokpuls:
   1. reset reg.A; reset buffer; transporteer adresveld instructiereg. naar adresselector; check 13e adresbit ≠ 0 - programma-alarm;
   2. lees inhoud geheugenplaats X in buffer;
   3. breng inhoud buffer naar reg.A;
   4. einde der operatie.

12. Read Part (RDP). klokpuls:
   1. reset reg.Aup; reset buffer; transporteer adresveld instructiereg. naar adresselector; check 13e adresbit ≠ 0 - programma-alarm;
   2. lees inhoud geheugenplaats X in buffer;
   3. breng inhoud buffer (10 MSB) naar reg.Aup;
   4. einde der operatie.

De inhoud van reg.Alow wordt niet aangetast.

13. Write (WR). klokpuls:
   1. reset buffer; transporteer adresveld instructiereg. naar adresselector; check 13e adresbit ≠ 0 - programma-alarm;
   2. breng inhoud reg.A naar de buffer;
   3. lees buffer in geheugenplaats X;
   4. einde der operatie.

De inhoud van reg.A blijft bewaard.

14. Schuif rechts (SHR). klokpuls:
    1. is stand Counter 1111? - einde der operatie;
    2. schuif de inhoud van reg.A een plaats naar rechts; tel 1 op bij de Counter;
    3. spring terug naar klokpuls 1.

15. Schuif links (SHL). klokpuls:
    1. reset Counter;
    2. staat op MSB van reg.Alow een 1? - einde der operatie;
    3. schuif de inhoud van reg.A een plaats naar links; tel 1 op bij de Counter;
    4. is de stand van de Counter 1111? - programma - alarm;
    5. spring terug naar klokpuls 2.

In het laatste overzicht wordt wederom de voor elke instructie benodigde tijd (inclusief instructiecyclus) berekend:

9. TF	3 + 3 klokpulsen		6 klokp.	= 0,30mSec.
10. TFB	3 + 3 ,,		6 ,,	= 0,30mSec.
11. RD	3 + 4 ,,		7 ,,	= 0,35mSec.
12. RDP	3 + 4 ,,		7 ,,	= 0,35mSec.
13. WR	3 + 4 ,,		7 ,,	= 0,35mSec.
14. SHR	3 + N' x 3 + 1 klokp.	max.31	,,	= 1,55mSec.
	(N' = aantal schuiven, max. 9)			
		tijd = (0,20 + 0,15N')mSec.		
15. SHL	3 + 1 + N x 4 + 1 klokp.	max.41 klokp.		= 2,05mSec.
	(N = aantal schuiven, max 9)	tijd = (0,25 + 0,20N)mSec.		

(Is het aantal schuiven N + N' = 10, delen,hoofdstuk 3.1., dan is de tijd voor de instructies 14 en 15 samen (1,95 + 0,05N)mSec. Deze tijd is dan maximaal voor N = 9, tijd 2,40mSec.)

2.5. Input-/Output-instructies.

Bij het schrijven van dit rapport was nog te weinig bekend omtrent de output-apparatuur. Met voorbehoud worden hier de instructies gegeven. Aanvullende gegevens zijn te vinden in hoofdstuk 3.4.: de conversie van binaire naar binair-decimale code en omgekeerd.

De volgende instructies zijn in deze groep samengevat:

16. lees	LS	0X	Lees de inhoud van de ponsband-lezer op geheugenplaats X en in successief hogere adressen van het kerngeheugen; ga zo door totdat een "end-of-line"-teken optreedt;
17. print	PT	1X	Print de gegevens van output-register X op de bijbehorende printer; verzorg de conversie van binair naar binair-decimale code.

De verdeling der handelingen over de diverse klokpulsen is als volgt.

16. Lees (LS). klokpuls:

1. breng inhoud adresveld van het instructieregister naar de a dresselector van het kerngeheugen;
2. reset de buffer; reset reg.Aup; reset reg.Alow; start ponsbandinvoer;
3. is er een "end-of-line"-teken? - einde der operatie;
4. breng de inhoud van het input-register naar reg.Aup; verhoog de inhoud van het adresveld van het instructieregister met 1;
5. transporteer de inhoud van reg.Aup naar reg.Alow;
6. reset reg.Aup;
7. breng tweede gegeven van inputregister naar reg.Aup;
8. breng de inhoud van reg.A naar de buffer;

9. lees de inhoud van de buffer op geheugenplaats X;
10. spring terug naar klokpuls 1.

Aangenomen wordt, dat de ponsbandlezer 10 registratiesporen heeft (exclusief het "end-of-line"-teken). Stel dat ingelezen wordt in het geheugen vanaf geheugenplaats 2830. Het eerst gelezen woord (van 10 bits) gaat naar de rechterhelft van geheugenwoord 2830. Het tweede woord van de ponsbandlezer gaat naar de linkerhelft van geheugenwoord 2830. Het derde woord gaat naar de rechterhelft van geheugenwoord 2831 etc. Op de ponsband moeten dus steeds bijelkaar horende woorden aanwezig zijn.

17. Print (PT). klokpuls:
1. start outputprinter X;
2. einde der operatie.

Deze instructie verzorgt ook de omzetting van code zoals deze beschreven wordt in hoofdstuk 3.4. Deze omzetting geschiedt automatisch.

Ook bij de input/output-instructies wordt aan het adres van de instructie een controlebit toegevoegd. Een programmaalarm kan ingeschakeld worden als een foutieve adressering plaatsvindt.

De tijd voor instructie 16(LS) wordt bepaald door de snelheid der invoerorganen (de machinetijd voor het inlezen van twee woorden van 10 bits is 0,50mSec.). Een invoersnelheid van 1000 karakters van elk 10 bits is bereikbaar met een snelle ponsbandlezer (d.w.z. 2mSec. zijn nodig om twee woorden van 10 bits in te lezen.). Het geheugen kan dan in ongeveer 8 seconden volgelezen worden (4000 woorden van 20 bits).

2.6. Controle-instructies.

De controle-instructies maken het mogelijk in het programma sprongen te maken of het programma te stoppen. De sprongen kunnen onvoorwaardelijk gemaakt worden. Anderzijds is het mogelijk sprongen te maken aan de hand van de stand van bepaalde indicatoren. Momenteel is nog niet bekend via welke indicatoren een spring-instructie geactiveerd kan worden. Het stoppen van de machine kan eveneens onvoorwaardelijk, dan wel aan de hand van de stand van bepaalde indicatoren veroorzaakt worden.

De volgende instructies zijn onderzocht:

18.	spring op pos-ind.	BP	OX	spring naar aangegeven adres als de positief indicater aan is;
19.	spring op zero-ind.	BE	OX	spring naar aangegeven adres X als de zero(nul-)-indicator aan is;
20.	spring	B	OX	spring naar adres X;
21.	spring op MSB-ind.	BNI	OX	spring naar aangegeven adres X als de meest significante bit van reg. Alow een 0 is (ind. is uit);
22.	stop	ST	-	stop de machine; vervolg het programma met de volgende instructie als startknop ingedrukt wordt;
23.	halt	H	-	stop de machine; begin het programma met de instructie op geheugenplaats 0 als de startknop ingedrukt wordt.

De volgende klokverdeling geeft de volgorde van handelen bij de onderscheiden instructies:

18. Spring op pos-ind.(BP). klokpuls:
    1. is pos-ind. uit? - einde der operatie;
    2. reset de programmateller;
    3. breng de inhoud van het adresveld van het instructieregister naar de programmateller;
    4. einde der operatie;

19. Spring op zero-ind. (BE). klokpuls:
    1. is de zero-ind. uit? - einde der operatie;
    2. reset de programmateller;
    3. breng de inhoud van het adresveld van het instructieregister naar de programmateller;
    4. einde der operatie;
20. Spring (B). klokpuls:
    1. reset de programma-teller;
    2. breng de inhoud van het adresveld van het instructieregister naar de programmateller;
    3. einde der operatie;
21. Spring op MSB-ind. (BNI). klokpuls:
    1. is de meest significante bit van reg.Alow een 1? - einde der operatie;
    2. reset de programmateller;
    3. breng de inhoud van het adresveld van het instructieregister naar de programmateller;
    4. einde der operatie;
22. Stop (ST). klokpuls:
    1. onderbreek de klokpulsgenerator;
    2. einde der operatie;
23. Halt (H). klokpuls:
    1. reset de programmateller;
    2. onderbreek de klokpulsgenerator;
    3. einde der operatie.

Bij de instructies 22 en 23 geschiedt het stopzetten van de machine door een onderbreking aan te brengen in de klokpulsgenerator.

De voor deze instructies benodigde tijd en het totaal aantal klokpulsen is in het ondervolgend staatje aangegeven.

18. BP	3 + 1 of 4	klokpulsen	max. 7 klokp.	= 0,35mSec.
19. BE	3 + 1 of 4	,,	max. 7 ,,	= 0,35mSec.
20. B	3 + 3	,,	6 ,,	= 0,30mSec.
21. BNI	3 + 1 of 4	,,	max. 7 ,,	= 0,35mSec.
22. ST	3 + 2 klokp.	afhankelijk van bediening	min. 5 ,,	= 0,25mSec.
23. H	3 + 3 ,,	afhankelijk van bediening	min. 6 ,,	= 0,30mSec.

2.7. Samenvatting.

De in de voorgaande paragrafen 2.2. + 2.6. ontwikkelde en geanalyseerde instructies zullen, met verwijzing naar figuur 2.2., worden samengevat. In figuur 2.2. zijn dezelfde onderdelen en registers geschetst als in figuur 2.1. Daarnaast zijn in figuur 2.2. alle poorten getekend die de verschillende componenten met elkaar verbinden. Een poort kan geactiveerd worden door een bepaalde combinatie van tijd (klokpuls), van de uit te voeren opdracht (operatie) en in enkele gevallen van het gekozen adres.

Figuur 2.3. geeft aan welke poorten bij een bepaalde instructie op ieder tijdstip geactiveerd worden. De nummers corresponderen met de poortnummers van figuur 2.2. Opgemerkt wordt nog dat een poort bij meerdere combinaties van tijd en operatie open kan gaan. Geen poging is nog gedaan om het mogelijk aantal combinaties te reduceren.

In figuur 2.1. is aangegeven welke indicatoren en welke programma-alarm-punten aanwezig zijn.

3. PROGRAMMA-ONDERDELEN.

In dit hoofdstuk zullen, voorafgaande aan de bespreking van het hoofdprogramma, enkele programma-onderdelen nader beschouwd worden. De bedoeling is enige wendbaarheid te ontplooien met de in het voorgaande hoofdstuk gegeven instructies.

In een gecompliceerd hoofdprogramma is het mogelijk dat bepaalde bewerkingen veelvuldig voorkomen. Deze, wat bewerking betreft identieke programma-onderdelen kunnen dan in het hoofdprogramma een te grote ruimte innemen. Het is dan mogelijk in het hoofdprogramma te verwijzen naar een hulpprogramma zo vaak als dit in het hoofdprogramma wenselijk is. In het hulpprogramma zijn dan slechts eenmaal de noodzakelijke bewerkingen aangegeven. Het hoofdprogramma specificeert dan welk hulpprogramma uitgevoerd moet worden. De gegevens voor dit hulpprogramma staan dan op vaste geheugenadressen. Ook de resultaten van het hulpprogramma komen op vaste geheugenadressen. Dergelijke hulpprogramma's worden wel subroutines genoemd.

Vier voorbeelden van programma-onderdelen zullen worden besproken.

1. programma-onderdelen, waarbij een arithmetische bewerking betrokken is;
2. een voorbeeld zal gegeven worden op welke wijze in het hoofdprogramma een verwijzing naar een subroutine gemaakt kan worden. Na voltooien van de bewerkingen der subroutine wordt naar het hoofdprogramma teruggesprongen;
3. een subroutine voor het zoeken van gegevens in tabellen;
4. tenslotte zal aangegeven worden hoe de omzetting van binaire naar binair-decimale code verloopt.

3.1. Arithmetische Programma-onderdelen.

Het feit dat er uitsluitend positieve, gehele getallen van 10 bits lengte mogen voorkomen, is uitgangspunt geweest voor de arithmetische instructies. Daarbij mocht evenwel het product van een vermenigvuldiging of het deeltal van een deling een woordlengte hebben van 20 bits

Vooreerst zal een voorbeeld van een programma gegeven worden, waarbij getallen met een grotere dan de nominale woordlengte vermenigvuldigd worden. Een komma is ingevoerd. Vervolgens zullen in het algemeen geldende regels gegeven worden om arithmetische bewerkingen met grotere woordlengten (dus grotere precisie) uit te voeren. Daarna zal nader op het mechanisme van de deling ingegaan worden.

Gevraagd wordt (bij een registerlengte van 3 digits, decimale eenheden) de vermenigvuldiging 14 * 32,54 uit te voeren. Het geheugen heeft hierbij een woordlengte van 6 digits. Het voorbeeld met decimale cijfers betekent voor de (zuiver binair bedoelde) instructies geen complicatie; wel wordt afgesproken dat de MSB-ind. aan is als het getal op de meest significante positie van reg.Alow 5 of groter is. Van het probleem wordt verder gevraagd het product in afgeronde vorm (afronding na de komma) wederom in het geheugen terug te brengen. Gegeven is dat het getal 14 in het geheugen is vastgelegd op positie 0300 in de vorm 000.014; het getal 32,54 staat op geheugenpositie 0301 in de vorm 032.540. Het resultaat wordt gewenst op geheugenplaats 0302 in de vorm 000.RRR. Het programma begint op geheugenplaats 0200. Een hulpadres 0303 wordt gebruikt voor het opslaan van tussenresultaten.

De vermenigvuldiging wordt nu in twee delen gesplitst: 14 * 32 + 14 * 0,54. Het volgend programma kan nu opgesteld worden.

Naast het programma is in de laatste drie kolommen de inhoud der registers en de voor elke operatie benodigde tijd aangegeven.

geheugenpl.	operatie	adres	reg.Aup.reg.Alow.	reg.B	tijd
0200	RD	0300	000.014	---	0,35mSec.
0201	TFB	reg.Alow	014.014	---	0,30
0202	TF	reg.B	014.014	014	0,30
0203	RD	0301	032.540	014	0,35
0204	M	-	007.560	014	4,80
0205	WR	0303	007.560	014	0,35
0206	RD	0301	032.540	014	0,35
0207	TF	reg.Alow	032.032	014	0,30
0208	M	-	000.448	014	4,80
0209	TFB	reg.Alow	448.448	014	0,30
0210	TF	reg.B	448.448	448	0,30
0211	RD	0303	007.560	448	0,35
0212	A	-	455.560	448	0,45
0213	BNI	0215	455.560	448	0,35
0214	AM1	-	456.560	448	0,50
0215	TF	reg.Alow	456.456	448	0,30
0216	RDP	0300	000.456	448	0,35
0217	WR	0302	000.456	448	0,35
					15,15mSec.

In het algemeen kan bij vermenigvuldigen, delen, optellen en aftrekken op betrekkelijk eenvoudige wijze een dubbele nauwkeurigheid bereikt worden.

Neem aan de getallen A en B (20 bits lengte) verdeeld in A_1 resp. B_1 (10 meest significante bits) en A_2 resp. B_2 (10 minst significante bits).

Bij optellen geldt:

$$A + B = (A_1+A_2) + (B_1+B_2) = (A_1 + B_1) + (A_2 + B_2).$$

De optelling is nu gesplitst in delen, welke door de machine verwerkt kunnen worden. Indien bij de optelling van het minst significante deel een overflow optreedt, moet een eenheid bij het resultaat van de optelling van het meest significante deel opgeteld worden.

Aftrekken:

$$(A - B) = (A_1+A_2) - (B_1+B_2) = (A_1 - B_1) + (A_2 - B_2).$$

De tekenbepaling van het verschil levert enige complicaties op:

a. $A_1 - B_1$ pos.; $A_2 - B_2$ pos.; resultaat correct, teken +

b. $A_1 - B_1$ pos.; $A_2 - B_2$ neg.; van $A_1 - B_1$ een eenheid aftrekken; van $A_2 - B_2$ een eenheid aftrekken en complement nemen, teken +

c. $A_1 - B_1$ neg.; $A_2 - B_2$ pos.; van $A_1 - B_1$ een eenheid aftrekken; van $A_2 - B_2$ een eenheid aftrekken en complement nemen, teken -

d. $A_1 - B_1$ neg.; $A_2 - B_2$ neg.; resultaat correct, teken -

(N.B. met de gegeven instructies is het slechts via een kunstgreep mogelijk bij de stappen b. en c. het tussenresultaat te complementeren; de vergelijkingsinstructie (C) vergelijkt de inhoud van reg.B met reg.Aup; de inhoud van reg.A is na de vergelijkingsopdracht in complementaire vorm aanwezig.)

Vermenigvuldigen:

$$A \times B = (A_1+A_2) \times (B_1+B_2) = A_1 \times B_1 + A_1 \times B_2 + A_2 \times B_1 + A_2 \times B_2.$$

De laatste term kan veelal weggelaten worden; deze term valt in het algemeen binnen de nauwkeurigheid van de waarnemingen A en B. (N.B. de plaats van een komma moet in acht genomen worden.)

Delen:

$$\frac{A}{(B_1 + B_2)} = \frac{A}{B_1}\left(\frac{1}{1 + B_2/B_1}\right) = \frac{A}{B_1}(1 - B_2/B_1).$$

Ook hier zijn, door toepassing van een reeksontwikkeling, minder significante termen weggelaten. Deze deling met dubbele precisie vereist tevens een vermenigvuldiging van dubbele precisie.

Formules kunnen opgesteld worden voor meervoudige nauwkeurigheid.

De deling van een 20 bits lang getal door een 10 bits deler levert een 10 bits lang quotient op, althans als de deler groter is dan het getal gevormd door de 10 meest significante bits van het deeltal. Is dit niet het geval, dan is het antwoord 11 bits lang. Dit resultaat overschrijdt de capaciteit der registers. De delinginstructie (D) moet in de volgende zin opgevat worden. Vooreerst moet de inhoud van register B zodanig zijn, dat de meest significante bit uiterst links geplaatst is. Dit positioneren kan geschieden met de SHL-instructie (naar links schuiven), toegepast op gegevens in reg.Alow. Het aantal schuiven wordt in de Counter geregistreerd. Vervolgens moet de (verschoven) inhoud van reg.B groter zijn dan de inhoud van reg.Aup (10 MSB's van het deeltal). De vergelijkingsinstructie (C) geeft het resultaat van de vergelijking via de indicatoren aan; het getal in reg.A is na de vergelijking in complementaire vorm aanwezig en is direct te gebruiken voor de deel-instructie. Het vergelijken (C-instructie) gaat essentieel vooraf aan de deling (D-instructie). Het quotient van de deling komt na de D-instructie te staan in reg.Alow in de vorm ,QQQ. De werkelijke plaats van de komma wordt nu berekend via het geregistreerde aantal schuiven van de SHL-instructie. Een afronding kan via de BNI-instructie ingevoerd worden.

Voorbeeld: Voltooi de deling 732,720 (geheugenplaats 0300)door 000.086 (geheugenpl. 0301); het resultaat wordt

in afgeronde vorm gewenst op geheugenplaats 1300 in de vorm 000.QQQ. De rest bij de deling is verder buiten beschouwing gelaten. Het programma begint op adres 0200, de constante 003.000 is beschikbaar op adres 0400, 0401 is een hulpadres.

adres	operatie-adres	bewerking	reg.B	reg.Aup.reg.Alow	tijd
0200	RD 0301		---	000.086	0,35mSec.
0201	SHL -	schuiven	---	000.860	
0202	RDP 0400	komma-plaatsen	---	003.860	0,35
0203	TF reg.B	,,	003	003.860	0,30
0204	TFB Counter	,,	003	001.860	0,30
0205	S -	,,	003	002.860	0,60
0206	WR 0401	,,	003	002.860	0,35
0207	TFB reg.Alow	delen	003	860.860	0,30
0208	TF reg.B	,,	860	860.860	0,30
0209	RD 0300	,,	860	732.720	0,35
0210	C -	voorwaarde	860	732.720	0,55
0211	BP 0213	,,	860	732.720	0,35
0212	ST -	,,	860	732.720	-
0213	D -	delen	860	000.852	4,80
0214	RDP 0401	komma-plaatsen	860	002.852	0,35
0215	TF Counter	,,	860	002.852	0,30
0216	TFB reg.Alow	,,	860	852.852	0,30
0217	SHR -	,,	860	008.528	2,40
0218	BNI 0220	afronden	860	008.528	0,35
0219	AM1 -	,,	860	009.528	0,50
0220	TF reg.Alow	schrijven	860	009.009	0,30
0221	RDP 0301	,,	860	000.009	0,35
0222	WR 1300	,,	860	000.009	0,35
					14,05mSec.

Een eenvoudige en snelle deling kan gegeven worden, wanneer de deler een getal is, gelijk aan een macht van twee. De deling kan dan uitgevoerd worden door het deeltal in reg.A evenveel malen naar rechts te verschuiven als de exponent van de 2-macht bedraagt. Op dezelfde manier vermenigvuldigen door naar links te schuiven is niet mogelijk. Bij delen door 2-machten moet de Counter het aantal schuiven bevatten voordat de SHR-instructie wordt toegepast; bij de SHL-instructie wordt de Counter daarentegen gereset.

3.2. Verwijzing naar Subroutines.

In het onderstaande wordt een voorbeeld gegeven van een verwijzing naar een subroutine. In deze subroutine wordt een bewerking uitgevoerd met de gegevens X en Y (verkregen uit het hoofdprogramma). Met het resultaat van de bewerking Z = Z(X,Y) wordt in het hoofdprogramma verder gewerkt. In het hoofdprogramma staan X en Y resp. op de adressen 0500 en 0501; Z wordt op adres 0502 gewenst. Voor de subroutine moet gespecificeerd worden waar de gegevens X, Y en Z staan (neem aan resp. op 0973, 0974 en 0975). Voorts moet aangegeven worden waar de subroutine begint (hier op 1705) en op welk adres van het hoofdprogramma de subroutine na afloop van de berekeningen moet terugspringen (in het voorbeeld 0291).

```
   hoofdprogramma
   --------------
0283 RD  0500  (lees X)
0284 WR  0973  (X naar subr.)
0285 RD  0501  (lees Y)
0286 WR  0974  (Y naar subr.)
0287 RD  0290  (specificatie
0288 WR  1740   einde subr.)
0289 B   1705   ----------------------->  subroutine
                                          1705 RD 0973 (leesX)
                                          ----
0290 B   0291  (dummy-instr.)             1707 RD 0974 (lees Y)
                                          ----
                                          Bereken Z = Z(X,Y)
                                          ----
                                          1739 WR 0975 (schrijf Z)
                 <---------------------   1740 B  0291 (spring terug)
0291 RD  0975  (lees Z)
----
   vervolg programma
```

Gebruik is gemaakt van een zogenoemde dummy-instructie (0290). Deze instructie specificeert naar welk adres in het hoofdprogramma na afloop van de bewerkingen van de subroutine teruggesprongen moet worden. Deze dummy-instructie wordt gebracht naar het laatste adres van de subroutine (1740).

3.3. Linearisatie Subroutine.

Bij bepaalde soorten metingen is er geen evenredig verband tussen de gemeten waarde van spanning of stroom en de daarmee corresponderende physische grootheid. Bij thermokoppels en weerstandsopnemers verloopt de afgegeven spanning niet evenredig met de gemeten temperatuur (zie figuur 3.1.)

De linearisatie verloopt op de volgende wijze. De grafiek wordt verdeeld in 16 intervallen. Elk interval wordt gekenmerkt door zijn startwaarde X; bij iedere startwaarde X behoort een initiele Waarde Y en een increment a. De kromme wordt nu door 16 rechte lijnstukjes zo goed mogelijk benaderd.

De gemeten spanning (digitaal in 10 bits woordlengte aanwezig) geeft zelf aan in welk interval de linearisatie plaats vindt; de eerste vier bits n.l. van de gemeten spanning geven al een verdeling in 16 intervallen. Deze vier bits specificeren de startwaarde. Het verschil tussen de gemeten waarde en de startwaarde (dus de laatste zes bits van de meetwaarde) wordt vermenigvuldigd met het increment a. Bij het product wordt de initiele waarde Y opgeteld. Het resultaat wordt afgerond.

Het volgend voorbeeld zal ter verduidelijking gegeven worden. Stel de gemeten waarde is 731; dit getal is in reg.Aup vastgelegd. De initiele waarde Y voor het 7e interval (in dit voorbeeld met decimale cijfers wordt gemakshalve met 10 intervallen gewerkt) is 692, het increment a is 1,12. De subroutine begint op adres 0500. Vooraf wordt gegeven waar de tabel der initiele waarden en incrementen begint. In het voorbeeld staan de gegevens van het 0e interval op adres 0400; op adres 0401 staan de gegevens van het 1e interval etc. In de 16 geheugenwoorden der tabel staat links de initiele waarde (geheel en positief) en rechts het increment (positief). De initiele waarde behoeft niet op de kromme van figuur 3.1. te liggen. Het formaat van het binaire getal dat het increment voorstelt is 4,6 d.w.z. 4 binaire cijfers staan vóór de komma,

6 cijfers staan achter de komma. Het increment kan dus gegeven worden in getallen tot 16 met een nauwkeurigheid van $\pm \frac{1}{2} \times 2^{-6} =$ = 0,008. Deze bepaling van het increment is nauwkeurig genoeg.

Het volgende programma kan geschreven worden. Aangenomen wordt, dat de tabel beginnende op adres 0400 genomen wordt.

progr.adres	oper.	adres	reg.Aup.reg.Alow	reg.B	bewerking	tijd
0500	TF	reg.B	731.---	731	interval	0,30mSec.
0501	RD	0300	006.000	731	bepaling	0,35
0502	TF	Counter	006.000	731	,,	0,30
0503	TFB	reg.B	731.000	731	,,	0,30
0504	SHR	-	007.310	731	,,	1,10
0505	TF	reg.B	007.310	007	,,	0,30
0506	WR	0301	007.310	007	,,	0,35
0507	RDP	0800	400.310	007	,,	0,35
0508	A	-	407.310	007	Y,a zoeken	0,45
0509	TF	reg.Alow	407.407	007	,,	0,30
0510	RDP	0516	RD0.407	007	dummy 1	0,35
0511	WR	0516	RD0.407	007	,,	0,35
0512	WR	0521	RD0.407	007	dummy 2	0,35
0513	RD	0301	007.310	007	incr.berekenen	0,35
0514	TFB	reg.Alow	310.310	007	,,	0,30
0515	TF	reg.B	310.310	310	,,	0,30
0516	RD	0407	692.112	310	,,	0,35
0517	M	-	034.720	310	,,	4,80
0518	BNI	0520	034.720	310	afronden	0,35
0519	AM1	-	035.720	310	,,	0,50
0520	TF	reg.B	035.720	035	init.waarde	0,30
0521	RD	0407	692.112	035	,,	0,35
0522	A	-	727.112	035	resultaat	0,45
0523	TF	reg.Alow	727.727	035	,,	0,30
0524	RDP	0302	000.727	035	,,	0,35
0525	WR	0801	000.727	035	,,	0,35
						14,20mSec.

geheugenadres 0300 bevat de constante 006.000; 0301 is een hulpadres; 0302 bevat de constante 000.xyz. Op adres 0800 is aangegeven welke tabel gebruikt wordt; op adres 0801 staat het resultaat, deze waarde is geheel en positief.

Gebruik is gemaakt van twee dummy-instructies 0516 en 0521. Het programma telt 26 adressen; per tabel zijn nog 16 geheugenadressen nodig. 5 Hulpadressen zijn gebruikt.

Aan het plaatsen van de tabellen in het geheugen is een beperking opgelegd: de beginpunten van de tabellen mogen niet geplaatst worden op de adressen 1009 + 1023, 2033 + 2047, 3057 + 3071 en 4081 + 4095, indien de tabellen 16 woorden lang zijn.

3.4. Binair naar Binair-decimaal conversie v.v.

Het in het "Bidec"-rapport (zie aanhangsel) beschreven systeem zal gebruikt worden in alle output en enkele input-bufferregisters teneinde de conversie tot stand te brengen tussen de in het digitale deel vereiste binaire code en de voor de in- en outputapparatuur noodzakelijke binair-decimale code. De vereiste nauwkeurigheid bij het uitprinten van gegevens is drie decimale digits; dit impliceert, dat de in- en outputbuffers 12 FF's lange schuifregisters moeten zijn. Het systeem, waarbij het te converteren getal in dat register opgeslagen wordt, waarin tevens de conversie plaats vindt,zal gebruikt worden. Het is dan wel noodzakelijk een conversieverbod op te leggen voor de eenheden-decade (le groep van vier bits) tot na de derde schuifpuls; tevens geldt voor de tweede decade (de tientallen) een conversieverbod tot na de zesde schuifpuls. Er wordt nu reeds op gewezen dat er in de derde groep van vier bits (de meest significante bits)geen optelling of aftrekking kan plaats vinden. Per conversieschakeling zijn dus slechts twee groepen van diodes nodig. De conversie geschiedt in 19 klokpulsen: de oneven klokpulsen schuiven de inhoud van het register een plaats op; de even klokpulsen verzorgen de juiste correctie in iedere decade (tenzij er een conversieverbod geldt voor een bepaalde decade).

Voor de conversie van binair naar binair-decimaal gecodeerde getallen zijn per groep van vier bits 25 vereist. Het schuifregister moet naar links kunnen schuiven. De volgende logische vergelijkingen zijn enigszins vereenvoudigd in vergelijking met het in het "Bidec"-rapport voorgestelde systeem:

$$A' = B.(C + D);$$
$$B' = (A + B).(C + D);$$
$$C' = \overline{D}.(A + B.C);$$
$$D' = A + B.(C + D).$$

Hierin is A de meest significante bit("8") van iedere groep van vier bits, B de "4" bit etc. Voorts A' de logische functie die de complement-poort van flip-flop A stuurt.

Overeenkomstige vergelijkingen kunnen afgeleid worden voor de conversie van binair-decimaal naar zuiver binaire code. Het schuifregister moet naar rechts kunnen schuiven. Per groep van vier bits zijn nu 14 diodes nodig.
De volgende vergelijkingen kunnen afgeleid worden:

$$A'' = A.\overline{B}.(\overline{C} + \overline{D});$$
$$B'' = A.(\overline{B}.(\overline{C} + \overline{D}) + B);$$
$$C'' = A.\overline{B}.D;$$
$$D'' = A.$$

Deze conversiesystemen vormen een deel van het ontwerp der input/outputbuffers, teneinde een automatische conversie te verzekeren van alle getallen die aan de machine aangeboden of door de machine gepresenteerd worden.

4. HET HOOFDPROGRAMMA.

In het hoofdprogramma staan alle bewerkingen vermeld, die een meetpunt kan ondergaan. De meetkanalen worden in een bepaalde volgorde aangesloten. Deze volgorde wordt bepaald door een in het geheugen opgenomen lijst der aansluitpunten. Punt na punt van deze lijst wordt bewerkt. In deze lijst mogen meetpunten gedupliceerd worden, zodat ze gedurende de meetcyclus meerdere keren aangeschakeld kunnen worden.

De bewerkingen kunnen voor elk meetpunt verschillend zijn. Bij de lijst der meetpunten wordt tevens aangegeven welke bewerkingen uitgevoerd moeten worden. Daarnaast zijn de voor deze individuele aanpak der meetpunten noodzakelijke constanten (alarmniveau's, differenties, versterkingsfactoren, schaalfactoren etc.) op een bepaalde plaats in het geheugen ondergebracht.

Het stromingsdiagram van figuur 4.1.a. en 4.1.b. geeft aan hoe de bewerkingen kunnen verlopen.
De lijst der meetpunten wordt vanaf i = 0 doorlopen. Met i correspondeert een willekeurig meetpunt. Bij het meetpunt wordt aangegeven of, en indien ja welke alarmvergelijking getroffen dient te worden.Vervolgens wordt ook aangegeven of een differentie bepaald moet worden en indien zo, welke differentie-vergelijking uitgevoerd wordt.
Het resultaat van een meting wordt als er een alarm-overschrijding plaatsvond, op de stripprinter gedrukt. Hetzelfde geschiedt ook als in het programma een aantekening is opgenomen voor het telkens uitprinten van een bepaald meetpunt. Tijdstip, kanaalnummer en meetwaarde worden geprint; voor enkele meetpunten kan een linearisatie voorafgaande aan het uitprinten verzorgd worden. Van bepaalde meetkanalen kan informatie gewenst worden omtrent de bedrijfsvoering. Deze informatie wordt tijdens de logcyclus getypt op de schrijfmachine; een efficiency-berekening kan aan het einde van de logcyclus gegeven worden.

5. PROGRAMMA-ALARM.

In het voorgaande is reeds aangegeven hoe bij fouten in het programma of tijdens de bewerking van informatie programma-alarmpunten ingeschakeld kunnen worden.

Resumerend zijn deze alarmpunten:

1. foutieve adreskeuze (instructieregister). Het alarm zal hoofdzakelijk te wijten zijn aan een foutieve adressering in het programma zoals dit in de machine gebracht is.

2. beide indicatoren aan. Dit alarm kan pas optreden ná een arithmetische instructie. Het alarm betekent dat de bewerkingen tijdens een of meer klokpulsen van een arithmetische operatie foutief zijn uitgevoerd. Een nauwkeuriger alarm is mogelijk als afgegaan wordt op de specifieke stand der indicatoren na iedere arithmetische bewerking zoals dit is aangegeven in het tweede overzicht van hoofdstuk 2.3.

3. Een programma-alarm is op de Counter geschakeld. Dit alarm treedt in werking als bij de SHL-instructie (links schuiven) meer dan 15 schuiven geregistreerd zijn (een maximum wat betreft de registerlengte,is 9 schuiven). De SHL-instructie wordt bij een deling gebruikt om de deler op de juiste plaats te brengen (meest sign.bit van deler op meest sign. bit van reg.B, zie ook hoofdstuk 3.1.). Worden meer dan 15 schuiven geregistreerd, dan betekent dit dat de deler nul is.

4. Voorts kan een programma-alarm ingeschakeld worden als de (verschoven) deler kleiner is dan de 10 MSB's van het deeltal. Het antwoord is dan 11 bits lang, de 11e bit van het antwoord is niet bekend. Dit programma-alarm wordt gecombineerd met de ST(stop)-instructie.

Naast deze alarmpunten kan een parity-check ingevoerd worden. Aan ieder woord van 10 bits wordt een bit toegevoegd zodat de som der "enen" van het woord en zijn parity-bit een even of oneven aantal is. De parity-bit kan ingevoerd worden tesamen met de gegevens in de ponsbandlezer. De controle op de pariteit kan uitgevoerd worden in reg.Aup, reg.Alow, het instructieregister en de outputregisters. In reg.Aup en reg.Alow kan de pariteit na een arithmetische bewerking hersteld worden. De registers worden nu evenals de geheugenwoorden 11 bits lang. Een voorbeeld van een parity-check wordt in figuur 5.1. gegeven. Het 10 FF's lange register Aup is voorzien van een PC(parity check)-FF. De indicator is uit als de pariteit goed is. Een controle kan uitgevoerd worden naar even of oneven pariteit naargelang de input bij de "exclusive or" van de 9e FF een "0" of een "1" is. De pariteit kan hersteld worden na een arithmetische bewerking via de gegeven poorten. Geen eenvoudige regels zijn te geven om de pariteit tijdens een arithmetische bewerking te controleren. Als een pariteitsfout ontdekt wordt kan een programma-alarm ingeschakeld worden.

Het is de bedoeling, dat zodra een programma-alarm optreedt, de machine stopt met de bewerkingen. Door middel van indicatorlichtjes kan op het bedieningspaneel kenbaar gemaakt worden tengevolge waarvan een programma-alarm geïnitieerd is. De inhoud der registers, de stand der indicatoren en de uit te voeren operatie kunnen eveneens via lichtjes op het bedieningsplateau aangegeven worden. De mogelijkheid moet bestaan om de machine op twee manieren te starten:

a. na indrukken van de startknop vervolgt de machine automatisch het programma;

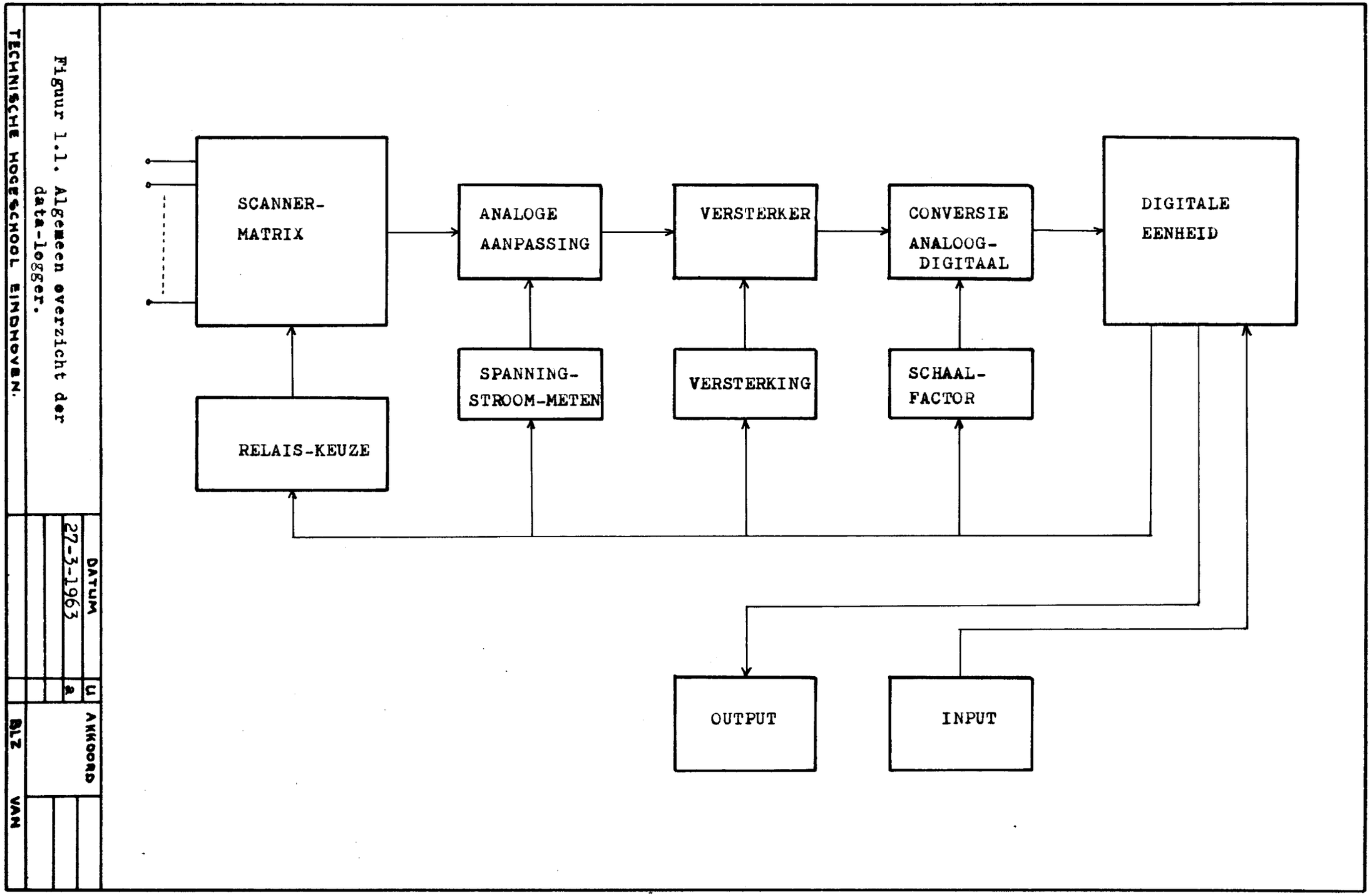
b. na indrukken van de startknop voltooit de machine één bij een operatie behorende bewerking (d.w.z. de klokpulsgenerator geeft één klokpuls af). De indicatorlichtjes op de bedieningslessenaar geven dan de mogelijkheid de fout tengevolge waarvan de machine gestopt is, op te sporen.

6. SLOTOPMERKINGEN.

Het gepresenteerde ontwerp is ontstaan in nauwe relatie met het beschrevene in de rapporten genoemd op pagina III. Ir. J.Hoogeveen heeft medegewerkt aan de in hoofdstuk 2.2., 2.5., 2.6. en 4 beschreven onderwerpen. De hoofdstukken 2.4, 3.3. en 3.4. zijn van de hand van J. S. Stenbit M.S. Volgende rapporten zullen de realisatie beschrijven van het digitale deel der data-logger.

A.G.M. Geurts, april 1963.

Technische Hogeschool
Eindhoven.
Afdeling EC B.



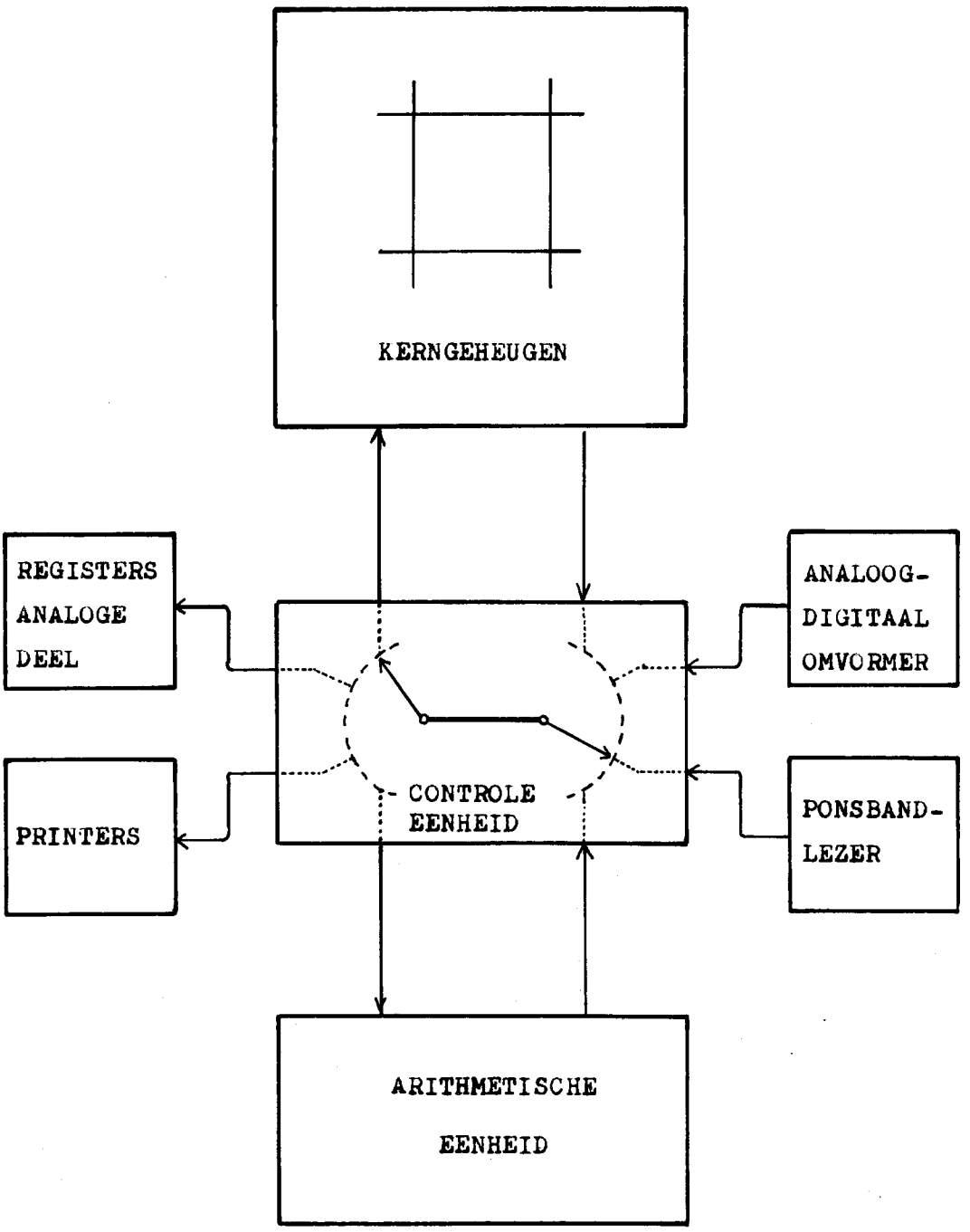
Figuur 1.1. Algemeen overzicht der data-logger.

TECHNISCHE HOGESCHOOL EINDHOVEN.

DATUM	U
27-3-1963	a

AKKOORD

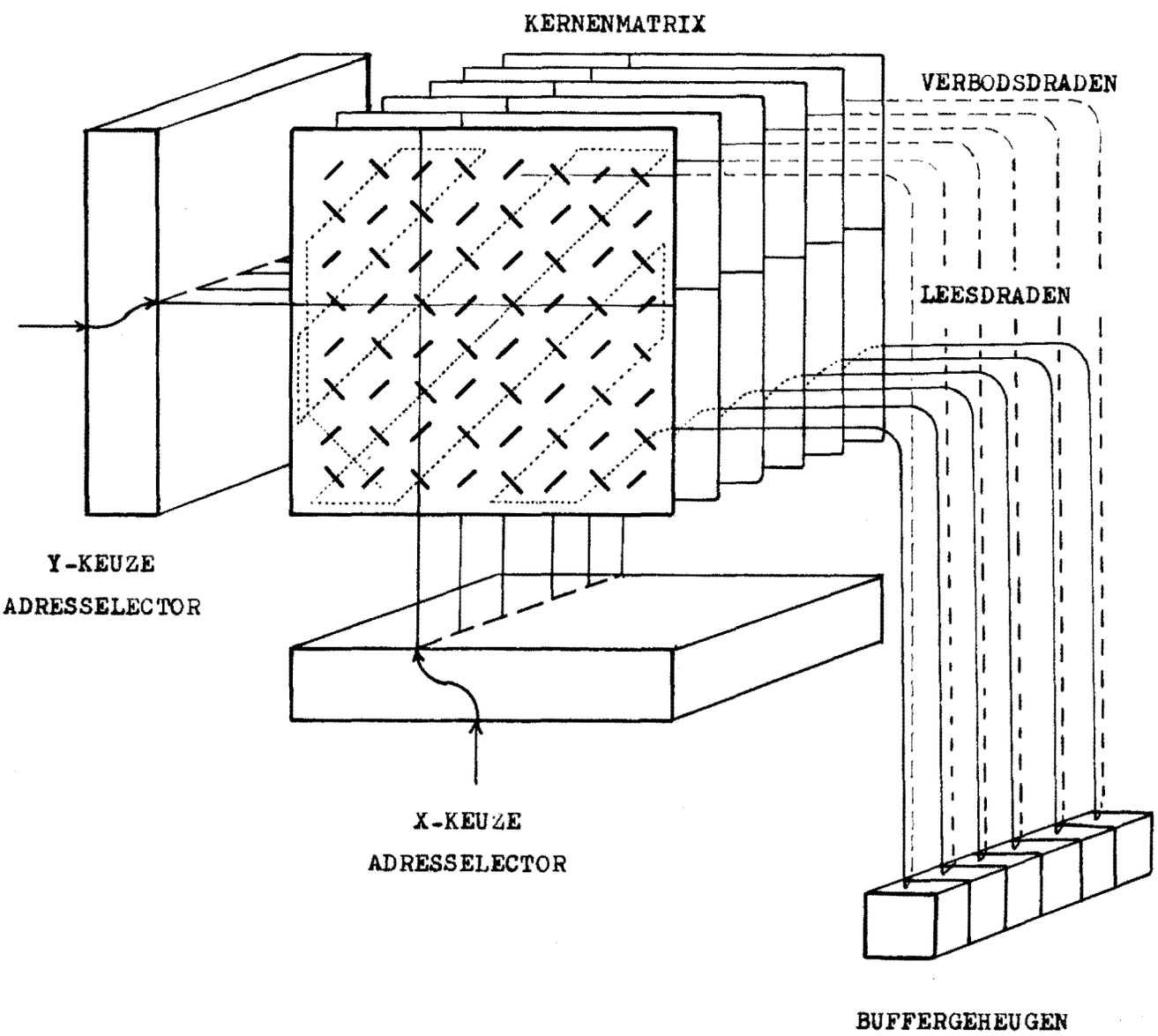
BLZ VAN



Figuur 1.2. Algemeen overzicht van het digitale deel.

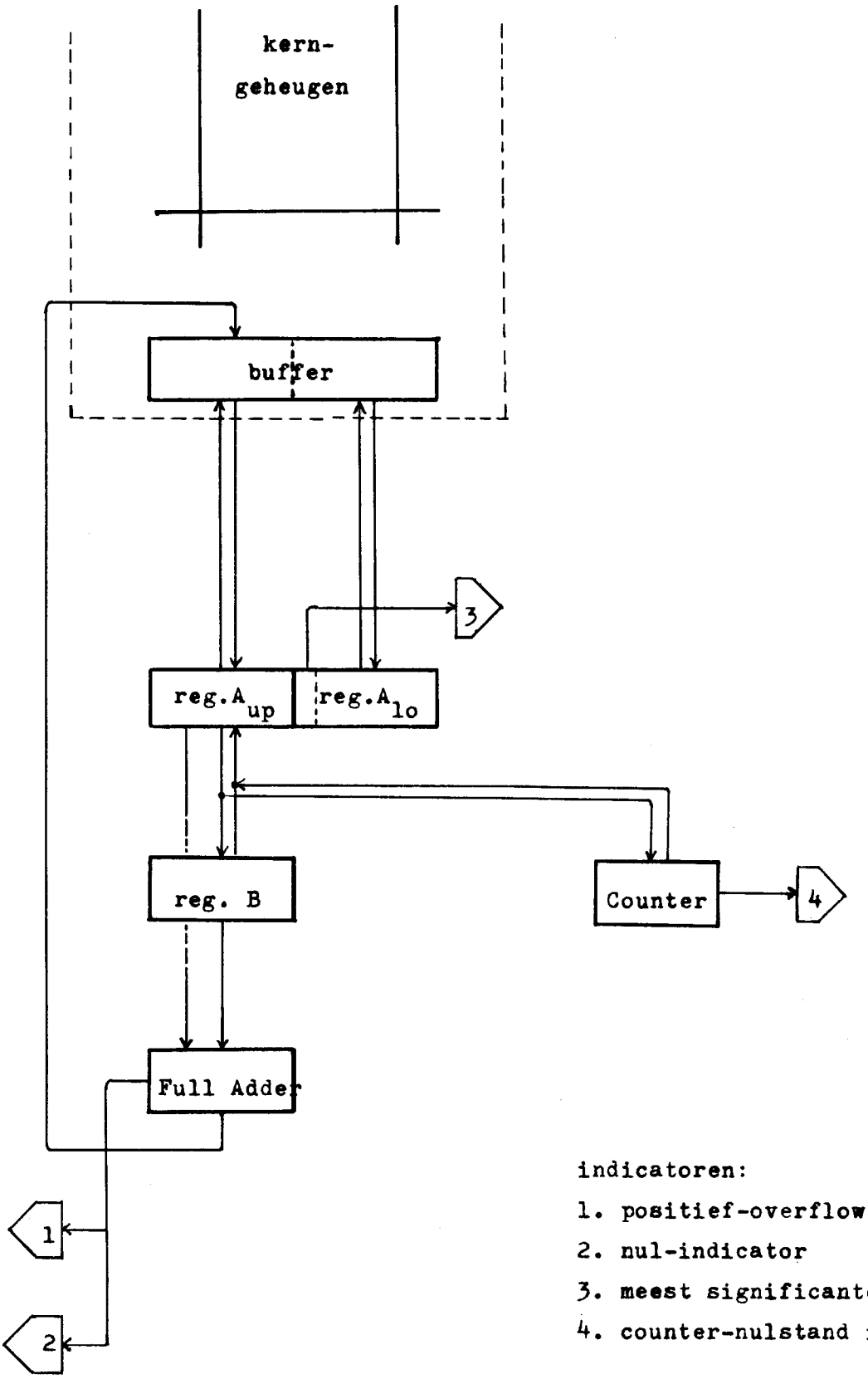
DATUM	U	AKKOORD	
~~27-3-1963~~	a		

TECHNISCHE HOGESCHOOL EINDHOVEN. BLZ VAN



Figuur 1.3. Algemeen overzicht van het kerngeheugen.

DATUM	U	AKKOORD	
27-3-1963	a		
TECHNISCHE HOGESCHOOL EINDHOVEN.		BLZ	VAN

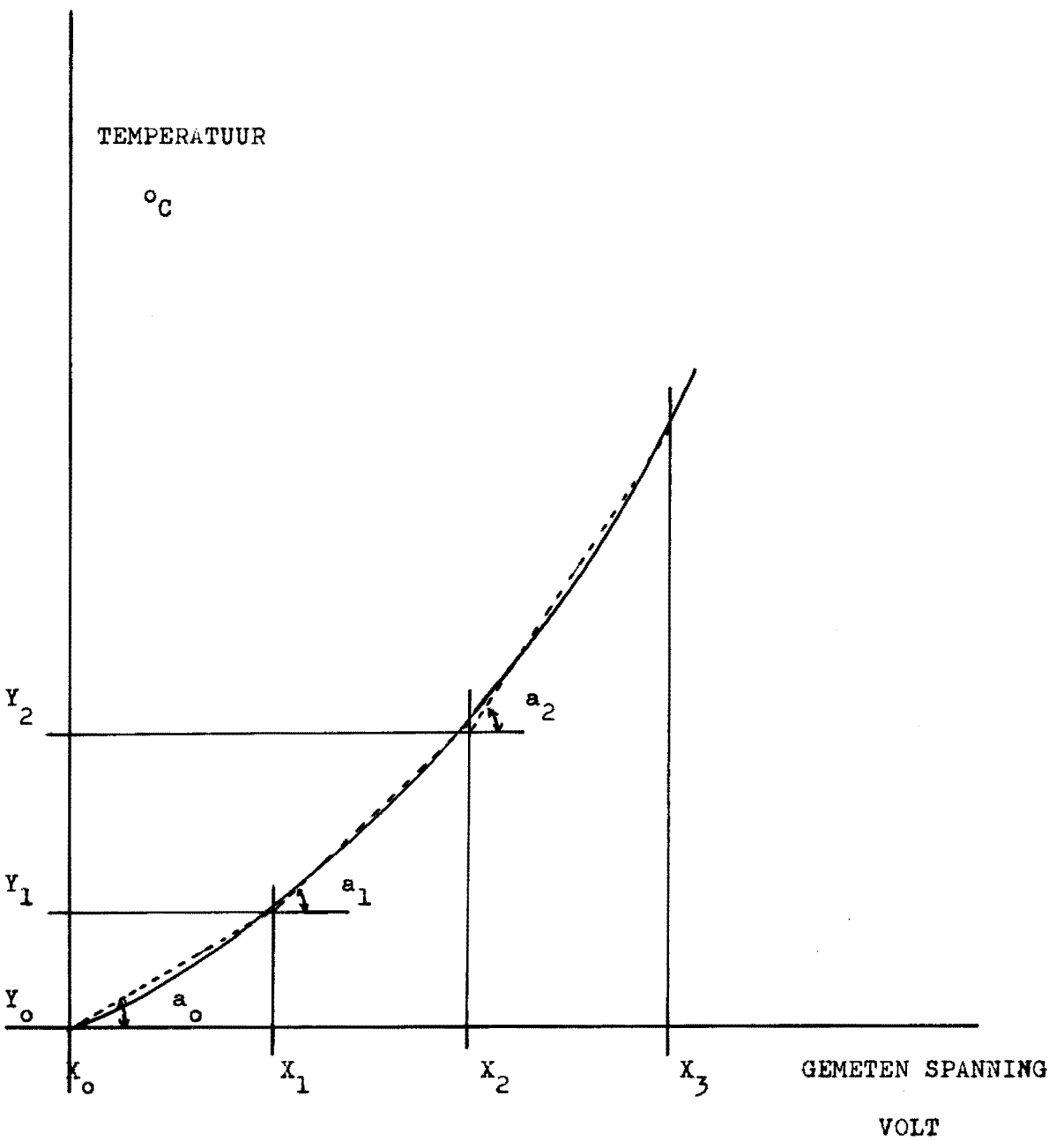


indicatoren:

1. positief-overflow indicator
2. nul-indicator
3. meest significante bit ind.
4. counter-nulstand indicator.

Figuur 1.4. Algemeen overzicht arithmetische eenheid.

	DATUM	U	AKKOORD	
	27-3-1963	a		
TECHNISCHE HOGESCHOOL EINDHOVEN.			BLZ	VAN

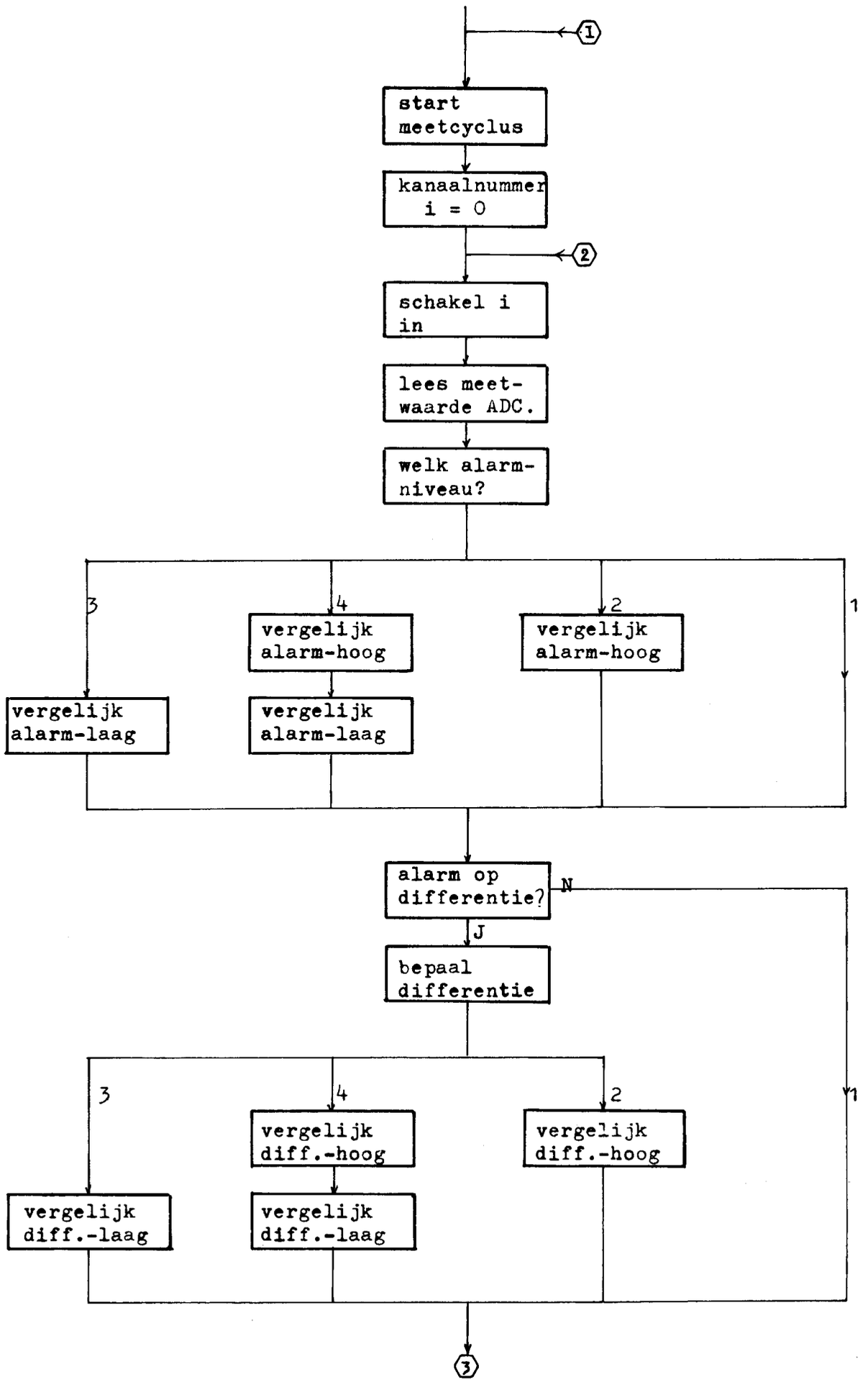


X = startwaarde

Y = initiele waarde

a = increment

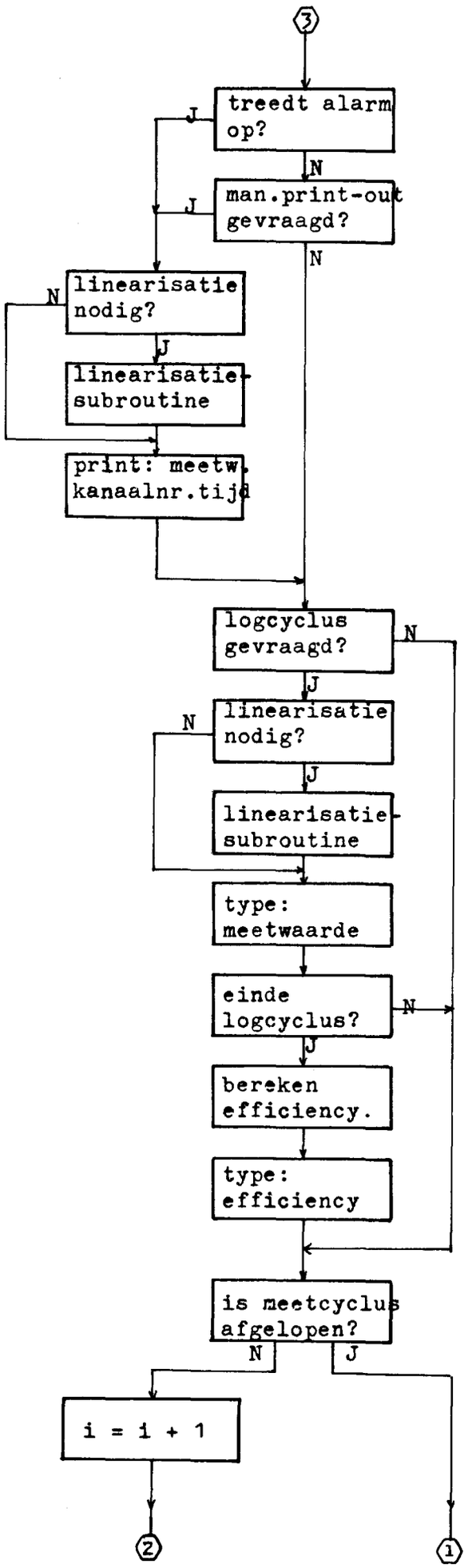
Figuur 3.1. Linearisatie van Thermokoppels.	DATUM 9-4-1963	U a	AKKOORD	
TECHNISCHE HOGESCHOOL EINDHOVEN.			BLZ	VAN



Figuur 4.1.a. Algemeen overzicht Hoofd-programma. Flow-diagram.

DATUM	U	AKKOORD	
16-4-1963	a		

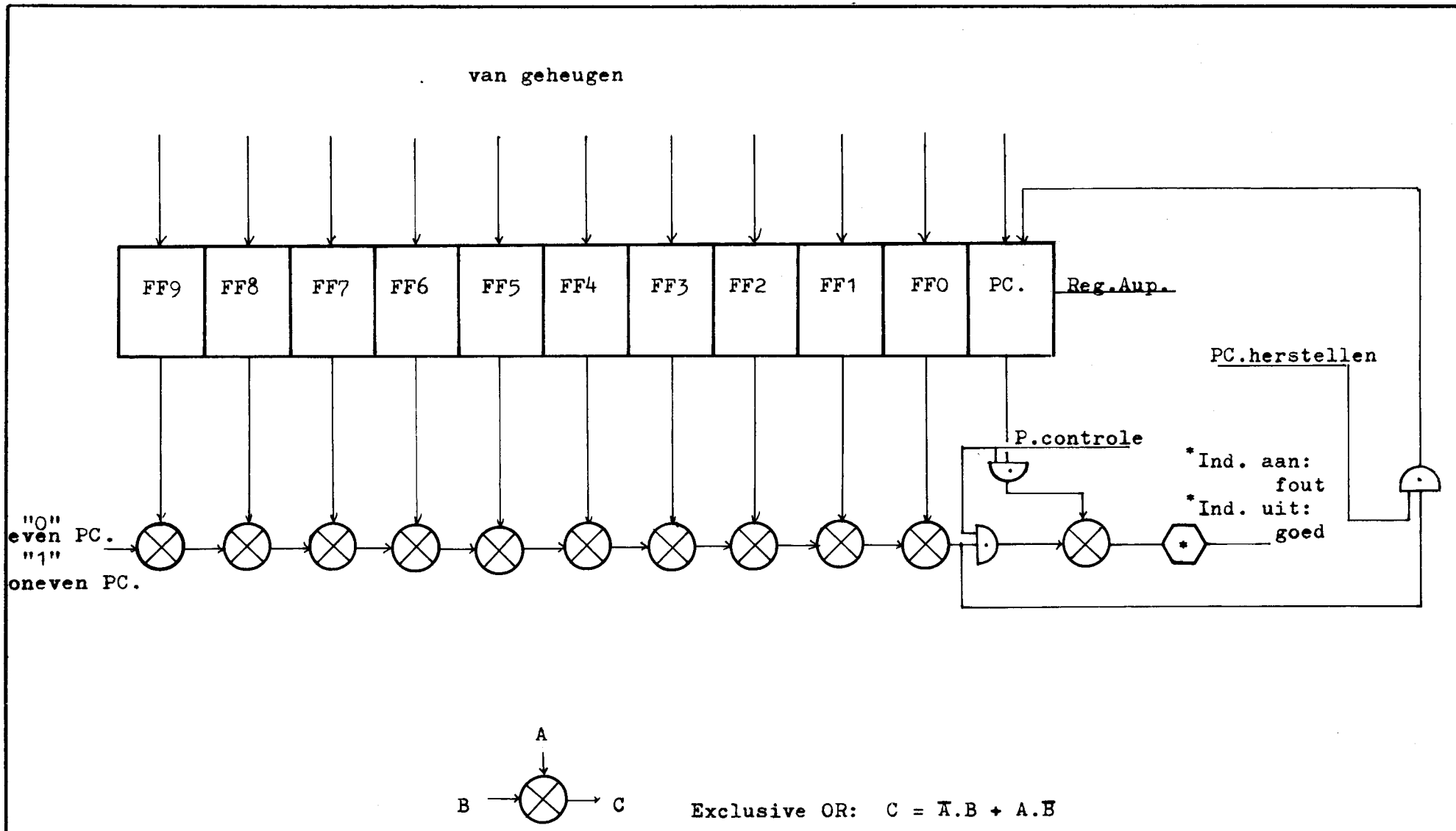
TECHNISCHE HOGESCHOOL EINDHOVEN. BLZ VAN



Figuur 4.1.b. Algemeen overzicht Hoofd-Programma. Flow-diagram.

DATUM	U	AKKOORD	
16-4-1963	a		

TECHNISCHE HOGESCHOOL EINDHOVEN. BLZ VAN



Exclusive OR: $C = \bar{A}.B + A.\bar{B}$

Figuur 5.1. Parity-controle en herstellen.

TECHNISCHE HOGESCHOOL EINDHOVEN.
BLZ VAN
AKKOORD
U
DATUM 10-4-1963
a

BIDEC—A Binary-to-Decimal or Decimal-to-Binary Converter*

JOHN F. COULEUR†

Summary—Simple, high-speed devices to convert binary, binary coded octal, or Gray code numbers to binary coded decimal numbers or vice versa is described. Circuitry required is four shift register stages per decimal digit plus one 30-diode network per decimal digit. In simple form the conversion requires two operations per binary bit but is theoretically capable of working at one operation per bit.

Introduction

THE following article is a description of devices to convert between the binary and decimal form of numbers. The method of conversion imposes no limitation on the number of digits and, in proposed advanced form, requires a time equal to the clock period times the number of digits to perform the conversion.

BIDEC is the result of a need for a fast, simple, binary-to-decimal converter to be used as the output buffer of a scale factor computer to reduce missile flight data. A study of conventional techniques indicated that time available for the conversion ruled out the use of counters. Complexity, size, and cost made the conventional division by 10 undesirable, and word length eliminated matrix conversion. Several other special methods of converting binary to decimal and decimal to binary were found,[1] but they either had inherent speed limitations or became increasingly complex with the number of digits to be converted.

The resulting search for an entirely new approach to the conversion of both binary to decimal and decimal to binary led to the formulation of the principles of operation of BIDEC, the original embodiments of which will be treated in this article. In the simplest form, either a binary or reflected binary number of any length in serial or parallel form may be converted to a binary coded decimal number, in a length of time equal to 2× number of binary digits × shifting time. Circuit requirements consist of four shift register stages per decimal digit, one, 30-diode logic per decimal digit, and a clock to count shift pulses. A change in wiring enables the conversion of decimal to binary, and an increase in complexity permits either conversion in the same device.

Theory

Making use of the fact that the decimal value of any digit in a binary number is 2^{N-1}, where N is location of the digit in the number, the decimal value of a binary number may be determined by adding appropriate powers of 2 as indicated by the presence of 1's in the binary number. The powers of 2 may be readily calculated by doubling one $N-1$ times or by multiplying the 1 in the binary number by two $N-1$ times.

A shift register is a convenient method of multiplication of a binary number by 2 since shifting the entire register once towards the most significant end doubles the value of all binary digits in the register if the stages are assigned arbitrary values of multiples of 2. With assignment of values to the stages of the shift register being somewhat arbitrary, there is no reason why the shift register may not be broken up into groups of four stages to be called decades. The four stages within the decades are assigned values of 1, 2, 4, and 8 ascending in the direction of shift, but the decades are assigned decimal values of 1, 10, 100, etc., ascending in the same direction. If a binary number of N digits is shifted into the register (most significant digit first), the most significant digit will be shifted and supposedly doubled $N-1$ times, and the following digits will also be shifted and doubled an equally appropriate number of times.

Obviously, the number in the register is not the decimal equivalent of the binary number and a quick examination will show why. The appropriate power of 2 was to be computed by doubling at each shift, but when a 1 was shifted from the "8" of any decade to the "1" of the next higher decade it only increased in value from 8 to 10. (Remember, the assignment of values was arbitrary.) Accordingly, a value of 6 was lost somewhere, and if the computation is to hold true, the 6 must be replaced. Therefore, 6 is added to the decade from which the one was shifted.

After the shifting is completed, only the necessity to clear out numbers over 10 in the decades by subtracting 10 and adding 1 to the next higher decade remains, and the conversion from binary to decimal has been accomplished.

While the above process performs the desired conversion, it is quite cumbersome to instrument and several simplifications may be made. Remembering that shifting to the left doubles the number in the register, 3 may be added to a decade when a "1" is sensed in the "8" stage since the subsequent shift will transfer the "1" to the next higher decade and double the 3 to the necessary 6.

To eliminate the possibility of a carry from the addition propagating into the next higher decade with the resulting necessity for adding 6 again, and to eliminate the clearing operation at the end of the conversion, it is desirable to prevent a decade from ever containing a number greater than 10. If the number in the register is 5, 6, or 7 prior to the shift, it will become a number

* Manuscript received by the PGEC, July 22, 1958. Revised manuscript received, August 7, 1958.

† General Electric Co., Syracuse, N. Y.

[1] Richards, Arithmetic Operations in Digital Computers, *et al.*

greater than 10 after the shift, and 5 should be subtracted from the decade and a 1 shifted into the next higher decade. In practice, this could consist of setting the "8" stage to "1" and subtracting 5, prior to the shift. This is the equivalent of adding 3, however, since $+8-5$ is equal to $+3$. A rule then develops: if the number in any decade is 8 or greater, add 3 before shifting, and if the number is 5, 6, or 7, add 3 prior to shifting. Or, in simpler terms, if the number in a decade is 5 or greater, add 3 prior to shifting. Since the number after shifting can never be greater than 9, only five possible cases exist to which 3 must be added: 5, 6, 7, 8, or 9. There is no carry to a higher decade, and each decade can stand as a separate entity.

It should be noted that there is no limit to the number of decades which may be cascaded. One decade is required for each decimal digit in the final answer.

The conversion from decimal to binary is based on repeated division of the decimal number by 2 with the remainders forming the binary number. With the decimal number entered in a shift register, divided in decades similar to that used for the binary to decimal converter, the division by 2 is performed by shifting toward the least significant end of the register.

When a 1 is shifted from the "1" stage of decade to the "8" stage of the next lower decade, however, a division by two is not performed since the value of the 1 changes from 10 to 8 rather than 10 to 5. Obviously, a value of 3 has been gained, which must be subtracted. If, after a shift, a decade has a 1 in the "8" stage, it can only have come from the next higher decade and, accordingly, the rule may be formulated: if, after the shift pulse, the number in a decade is 8 or greater, 3 must be subtracted. For obvious reasons there are again five possible numbers from which 3 must be subtracted: 8, 9, 10, 11, and 12. There is no borrow to the next decade, and again each decade can stand as a separate entity. The number of shift pulses to be applied is equal to the number of binary digits in the largest number which can be contained in the register.

Binary-to-Decimal Converter

The process of converting a number in binary form to its decimal equivalent consists of shifting the number into a shift register having $4N$ stages grouped to form N decades, the contents of each decade forming a decimal digit in binary coded form, and the number of N decades required being determined by the decimal equivalent of the binary number. The conversion process consists of shifting the binary number into the register one bit at a time, most significant digit first, testing the content of each decade prior to each shift and adding binary 3 to any decade which contains binary 5 or greater.

Fig. 1 is a block diagram of the converter, Fig. 2 is a block diagram of the logic circuitry associated with each of the decades, and Fig. 3 is an illustration of the operation of the converter.

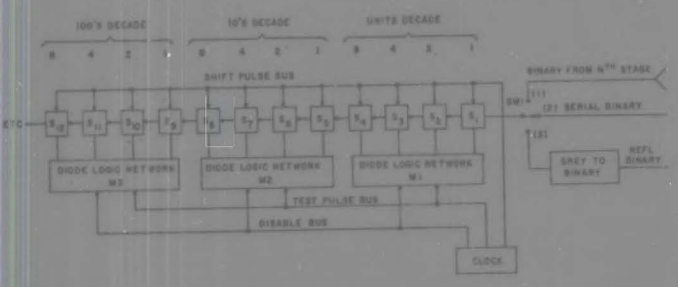


Fig. 1—Block diagram, binary-to-decimal converter.

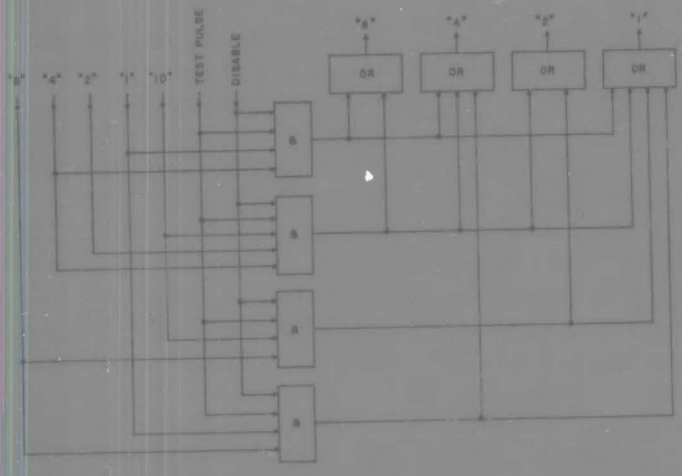


Fig. 2—Binary-to-decimal logic network.

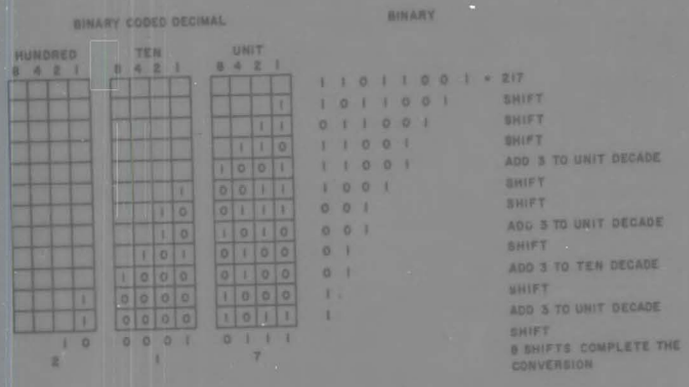


Fig. 3—Example binary to decimal. Note: prior to each shift, each decade is tested for 5 or greater, and, if so, 3 is added.

Referring to Fig. 1, there is shown a shift register which, for example, consists of 12 stages, S1 through S-12. The four stages S1–S4 plus network M1 form the units decade of the shift register, stages S5–S8 plus network M2 form the 10's decade, and stages S9–S12 plus network M3 form the 100's decade. Each stage of a decade, of course, will contain a binary "1" or a "0" and the stages within the decades have binary weights of 8, 4, 2, and 1 increasing in significance in the same direction as the decimal weights. For convenience, the stages within a decade will be referred to by their binary weight; *i.e.*, the "4" stage would refer to stages S3, S7, or S11. The shift register is connected to a clock, or source of shift pulses, in such a manner that each shift pulse shifts the number in the register one stage to the left, or toward a more significant value.

A second output of the clock preceding each shift pulse, is the test pulse applied to the diode networks shown in Fig. 2. In essence, the networks determine if the number in a decade is 5 or greater and, if so, add 3 by complementing the appropriate stages of the decade. For example:

0101 (5) should become 1000 (8),
0110 (6) should become 1001 (9),
0111 (7) should become 1010 (10),
1000 (8) should become 1011 (11), and
1001 (9) should become 1100 (12).

Examination will show:

If "4" and "1" are 1's, "8," "4," and "1" stages are complemented.
If "4" and "2" are 1's and "1" is 0, "8," "4," "2," and "1" are complemented.
If "8" is 1 and "1" is 0, "2" and "1" are complemented.
If "8" and "1" are 1, "4" and "1" are complemented.

The network contains the necessary logic to generate the proper pulses to be applied to the complementing inputs of the shift register stages. Many other networks, or adders, are available where the shift register stage does not lend itself to being complemented.

The clock contains a counter, and after a number of test and shift pulses equal to the number of binary digits in the word has been applied to the converter, the process is complete and the decimal number is in the converter in binary coded form.

The description of the conversion has been for a binary number in serial form. Switch 1 permits the conversion of a binary number in parallel form or a reflected binary number in parallel or serial form.

For the conversion of a binary number in parallel form, the binary number is read in parallel into the shift register with the most significant digit in the most significant stage (S12 in the example) and SW1 is put in position 1 connecting stage S1 to S12. Application of shift pulses causes the conversion to be accomplished as previously described. To prevent digits of the binary number from being converted prematurely, the networks are prevented from generating complementing pulses until the parallel-entered binary number has been shifted out of the decade. This is accomplished by means of the "disable" bus shown on the network in Fig. 2. The clock generates disable gates during the counting period in such a manner that the units decade network is "enabled" after three shift pulses, the 10's decade after six shift pulses, the 100's decade after nine shift pulses, etc., the number three deriving from the fact that three shifts are necessary to enter a number greater than 5 into a decade.

To convert either serial or parallel Gray code to decimal, the same converter is used with the addition of a serial Gray-to-binary converter at the input to stage S1. The Gray code converter consists of an "and" of the input stage and the test pulse. If the input stage is a "1," a pulse is generated which complements either a pre-input flip-flop, if the input is serial, or the last stage, if the parallel storage method is used.

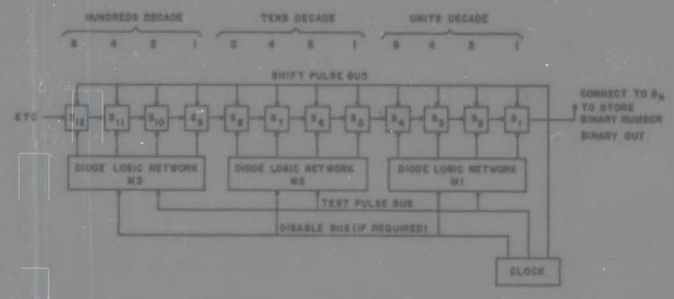


Fig. 4—Block diagram, decimal-to-binary converter. Note: shift register shifts to right.

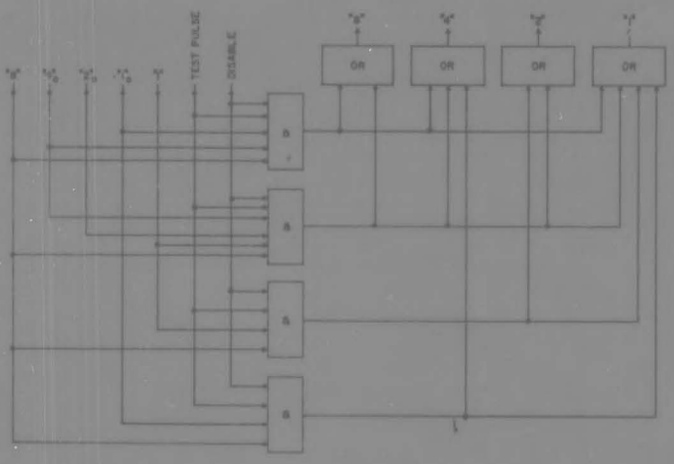


Fig. 5—Decimal-to-binary logic network.

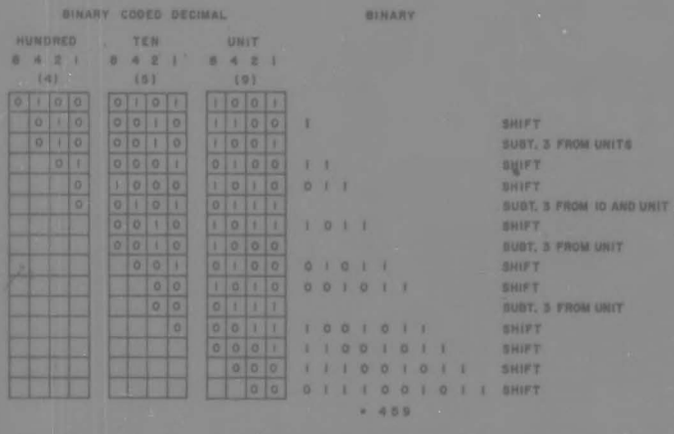


Fig. 6—Example decimal to binary. Note: prior to each shift, each decade is tested for 8 or greater, and, if so, 3 is subtracted.

Decimal-to-Binary Converter

The process of converting a number in decimal (binary coded decimal) form to its binary equivalent consists of shifting the number out of a shift register having $4N$ stages grouped to form N decades, the contents of each decade forming a decimal digit. The conversion process consists of shifting the binary number, one bit at a time, least significant digit first, out of the register, testing the content of each decade after the shift and

subtracting three from any decade containing eight or greater.

Fig. 4 is a block diagram of the converter, Fig. 5 is a block diagram of logic circuitry associated with each of the decades, and Fig. 6 is an illustration of the operation of the converter.

Referring to Fig. 4, there is a shift register and clock arranged in a manner similar to that of Fig. 1 except that the shift is toward the least significant digit, and the decimal number in binary coded form has been entered in the register in parallel form. A second output of the clock, following the shift pulse, is the test pulse, which is applied to the diode networks shown in Fig. 5.

The networks determine if the number in the decade is 8 or greater and, if so, subtract 3 by complementing the appropriate stages of the decade. For example:

1000 (8) should become 0101 (5).
1001 (9) should become 0110 (6).
1010 (10) should become 0111 (7).
1011 (11) should become 1000 (8).
1100 (12) should become 1001 (9).

Examination will show that:

If "8" is 1 and "4" and "1" are 0, "8," "4," and "1" are complemented.
If "8" is 1, "1" is 1, and "4" and "2" are 0, "8," "4," "2," and "1" are complemented.
If "8" is 1 and "1" is 1, "2" and "1" are complemented.
If "8" is 1 and "1" is 0, "4" and "1" are complemented.

The network contains the necessary logic to generate the proper complementing pulses.

After a number of shift and test pulses equal to the number of bits of the largest binary number which could be represented by a decimal number of N digits the conversion is complete, and the decimal number has been shifted out of the converter in serial binary form, least significant digit first.

If the decimal number is in serial form, it may be shifted into the register prior to the conversion. If the binary number is desired in parallel form, switch SW-1 is thrown to position 2 and the binary number circulated back into the shift register. However, disable gates must be generated to prevent complementing the binary number.

Conclusion

The circuit design of either of the converters is straight forward and there are many techniques available to provide the shifting and testing functions. The model built used flip-flop shift register stages and dc logic. As the stages complemented in the testing operation affect the logic, it is necessary to delay the stage inputs to the logic by approximately the width of the test pulse.

Further study indicated that theoretically, the speed of conversion could be doubled at the expense of some complexity by eliminating the "test" operation. Essentially, the presence of a 5 or greater in a decade would block the shift pulse to the "8," "4," and "2" stages of that decade, and these stages would be set to the number plus 3 as it would have been shifted. Using 1-mc stages, a 20-digit binary number or 6-digit decimal number could thus be converted in 20 μsec.

Diodeless Magnetic Shift Registers Utilizing Transfluxors*

NOAH S. PRYWES†

Summary—**Shift registers using magnetic cores have been conventionally designed with diodes to provide isolation between stages. It is possible to use transfluxors in a two-core-per-bit fashion so as to provide isolation between stages by magnetic means rather than with diodes. A design procedure for doing so is discussed.**

I. Introduction

MAGNETIC materials with a characteristic as shown in Fig. 1 can be used for information storage. Their main useful features are two opposite remanent magnetizations at saturation and a minimal coercivity H_c required for a domain change in a process of magnetization reversal.

To explain further the magnetic characteristic, consider, for instance, an initial situation of $+B_s$ and $H=0$ in Fig. 1. For a positive or a negative field intensity smaller than H_c, the nearly horizontal line at the top of the characteristic in Fig. 1 will represent the corresponding changes. At the removal of such a field, return to the magnetization of $+B_s$ and $H=0$ will occur. Thus any change of flux at the application of such a field was reversible. However, if a negative field in excess of H_c is applied, an irreversible flux change occurs along the nearly vertical part of the left of the major hysteresis loop shown in Fig. 1. The average rate of change of such flux along this line is approximately proportional to the amount by which the applied field exceeds the coercive field H_c. If the applied field is removed before the complete reversing of magnetization along the major hysteresis loop, the return to $H=0$ will occur along a

* Manuscript received by the PGEC, September 3, 1958.
† Moore School of Elec. Eng., Univ. of Pennsylvania. Formerly with Remington-Rand UNIVAC, Philadelphia, Pa.

LITERATUURLIJST

behorende bij DATA-LOGGER rapporten.

De literatuuropgaven zijn per rubriek gerangschikt in alfabetische volgorde. Voor zover bekend zijn bij boeken de coderingen der T.H.-bibliotheek aangegeven.

INDELING

1. Algemeen

1.1. Data loggers

1. "Automatic Control and Computer Engineering"
   V.V. Solodovnikov; Vol. I (EO 6109)
2. "Automation in Business and Industry"
   E.M. Grabbe (PG 5705)
3. "Digital Techniques for Computation and Control"
   Klein, Morgan, Aronson, (EO 5809)
4. "High Speed Data Processing"
   G.C. Gotlieb; J.N.P. Hume, (EC 5821)
5. "Automatic logging system speeds Data Reduction"
   Automatic Control, 1954, Aug.,Nov.,Dec., 1955, Jan.
6. "Automatic Data Handling Equipment For Wind Tunnels"
   J.R. Burchall, N. Fenner,
   British Communication and Electronic, Vol. 9 1962 Jan., p.p. 14-17.
7. "Collecting Process Data for an On-line Digital Computer"
   D.A. Fluegel, E.O. Tolin,
   Control Engineering, Vol. 8, 1961 April, p.p. 147-150
8. "Data-logging-Scanning-alarming-Calculating in power plants and Substations"
   B. Zambotti, Electrical World, 1961, November.
9. "The Evolution of Automation"
   American Power Conference, Chicago, Ill., 1959, Maart-April.
10. " Un Equipment Numérique de Mésure et de traitement des Informations pour grands Ensembles Industriels"
    M. Metivier, Automatisme, Tome VI, 1961 no. 7-8 p.p. 259-265;
    no. 9, p.p. 308-316.
11. "Garde System" American General-Electric
    Mynssen en Co., Amsterdam.
12. "A High Speed Digital Data Logging System" H. Fuchs, J.R. Wartell,
    Electronic Engineering, 1960. Vol. 32, p.p. 468-472.
13. "Hinkley-Point Temperature Scanning Equipment"
    R.I. Ostler, J.M. Tyrrill.
    Journal British I.R.E., Febr. 1962 p.p. 83-88.
14. "La Technique Moderne"
    J. Auricoste, J. Prades, Tome L IV nr. 3, p.p. 126-146.
15. "Revue Générale de l'electricité"
    J. Auricoste, 1961 October p.p. 477.

16. "Combustion", 1961 July.
17. "Electrical World", 1958, December 22.
18. "Power", 1960 May.
19. "The Steamengineer", London 1957, September.

1.2. Centrales

1. "Combustion Engineering" O. de Lorenzi (GG 5731)
2. "Powerstation Engineering and Economy"
   B.G.A. Skrotski, W.A. Vopat (GG 6004)
3. "Automatisering van Centrales - Overzicht der Technieken"
   M. Gerstel, C.A. IMRA, N.V. PHILIPS.
4. "Automatisme et Controle dans les Centrales Thermiques"
   R. Chaussard, Automatisme Tome III, nr. 6, nr. 7
   Tome V, nr. 10, nr. 11
   Tome VI, nr. 1, nr. 2.
5. "Computer Applications in the Electric Utility Industry"
   D.T. Braymer. Electrical World 1961, oktober.
6. "1710 Control System for Steam-Electric Generating Units"
   I.B.M.
7. "Solid State Computer Automates Powerstation Operation"
   I.S.A. Journal Vol. 5, 1958 oktober, nr. 10, p.p. 32 - 37.

2. Analoge deel

2.1. Algemeen

1. "Instrumentation in scientific research; Electrical input transducers"
K.S. Lion, McGraw-Hill (EC - 5909)

2.2. Relais

1. "Balanced Polar Mercury Relay"
J.T.L. Brown, C.E. Pollard; Bell System Technical Journal, New York, N.Y. Vol. 32, nr. 6, 1953, p.p. 1393-1411.
2. "Capability of sealed relays"
O.M. Hovgaard, AIEE - Trans. on Comm. and Electr. Vol. 75, 1956, p.p. 466 - 468.
3. "The Development of Automatic Manufacturing Facilities for Reed Switches" J.A. Horsford, AIEE - Trans. on Comm. and Electr., Vol. 75, 1956, p.p. 496 - 500.
4. "Development of Reed Switches and Relays" O.M. Horgaard, G.E. Perrault, Bell System Technical Journal, Vol. 34, nr. 2, 1955, p.p. 309 - 332.
5. "Elektrische vorgänge im Schutzgaskontact beim schalten Induktiver Lastkreise" K.L. Rau,
Nachrichtentechnische Zeitschrift B.14, 1961, H.5.
6. "Glass-enclosed Reed Relay" W.B. Ellwood,
Electrical Engineering, Vol. 66, 1947, p.p. 1104-1105.
7. "Mercury Contact Relays", J.T.L. Brown, C.E. Pollard;
Electrical Engineering, Vol. 66, 1947, p.p. 1106-1109.
8. "Rectifier Networks for Multiposition Switching"
D.R. Brown, N. Rochester.
Proceedings IRE, 1949, p.p. 139-147.
9. "Sealed contact reed relays" J.G. Bannochie, R.A.E. Fursey,
Automatic Telephone and Electr. Co. Journal, Vol. 14, 1958, p.p. 262-273.
10. "The symmetrical transistor as a bilateral switching element"
R.B. Trousdale, AIEE-Trans. on Comm. and Electr.,
Vol. 75, 1956, p.p. 400-403.
11. "A versatile miniature switching capsule" O.M. Hovgaard, W.J. Fontana.
Proceedings 1959 Electr. Comp. Conference, p.p. 32-37.
Zie ook: Bell Telephone System, Technical Publication, Monograph 3306.

2.3. Versterkers.

1. "Differential Amplifiers" J. Truxall
"Control-engineer's handbook" 6 - 29.
2. "Common-mode Rejection" R.S. Burwen
Electronics 32, 1959 June-July.
3. "Der Fitgo-Verstärker", H. Glever
Regelungstechnik, B 8, 1960, H.10, p. 371.
4. "Transistor Amplifiers for D.C. Signals"
Electronic Applications Vol. 19, 1958-1959, Nr. 1, p.p. 1-14.

2.4. Analoog-Digitaal omvormers

1. "Digital Techniques for Computation and Control"
Klein, Morgan, Aronson (EO 5809)

2.5. Diversen

1. "Reference tables for thermo-couples"
N.B.S. Circular 561.

3. Digitale deel

3.1. Algemeen

1. "Analog and Digital Computer Technology"
N.R. Scott, (EO 6003)
2. "Arithmetic Operations in Digital Computer"
Richards, (EO 5701)
3. "Organisation et Fonctionnement des Machines Arithmetiques"
H. Boucher (BP 6005)
4. "Les Techniques Binaires et le traitement de l'information"
H. Soubies - Camy (BP 6110).

3.2. Programmering

1. "Digital Computer Programming"
D.D. McCracken (BP 5706)
2. "Programmieren von Ziffern Rechenanlagen"
W. Knödel (BP 6114)
3. "Programming for Digital Computers"
J. Jeenel (BP 5902)

3.3. Diversen

1. "Bidec - A Binary-to-Decimal or Decimal-to-Binary Converter"
J.F. Couleur, I.R.E.-Trans. on Electr. Comp. 1958, dec., p.p. 313-316

4. Diversen

4.1. Rendement-berekeningen

1. "Combustion engineering" O de Lorenzi (GG 5731)
2. "Power Station Engineering and Economy"
   B.G.A. Skrotzki, W.A. Vopat (GG 6004)
3. "Thermodynamic properties of water and steam"
   M.F. Vukalovitch, VEB-Verlagtechnik, Berlin (DC 5815)
4. "Wasserdampftafeln" VDI-tebellen 1960 (FA 5616)
5. "Bestimmung der zustandgrössen von Wasserdampf und Wasser auf Digitalen Rechenautomaten"
   H. Hotes; Allgemeine Wärmetechnik, band 9, nr. 11/12
6. "Neue zustandgleichungen für Wasserdampf unter Berücksichtigung ihrer verwendung in Elektronische Rechenanlagen"
   B.W.K., 5 juni 1962, band 14.