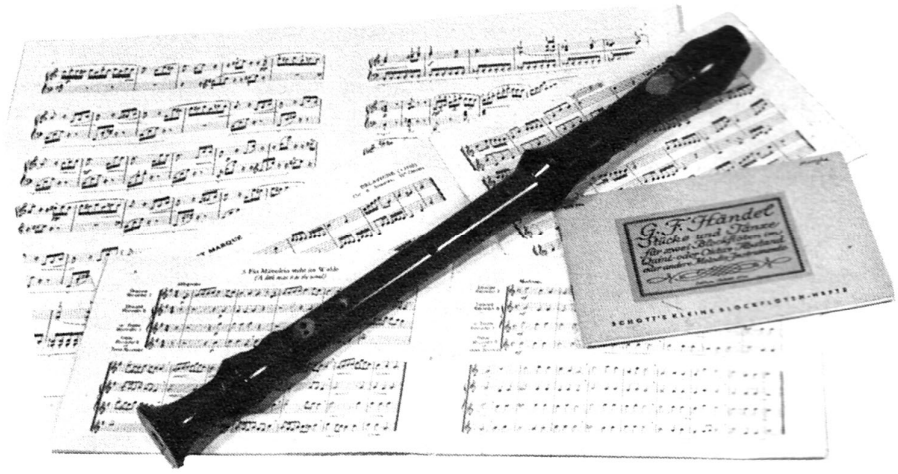
D.M. de Boer

De Melodiant



De Melodiant draait op een door u gegeven commando een complete melodie af, van maximaal 256 tonen. Het display laat hierbij zien welke noot ten gehore wordt gebracht. Hierdoor is de Melodiant een zeer veelzijdig instrument. Zo kan men b.v. samen met de Melodiant duetjes spelen, door van tevoren één van beide stemmen in te typen. Het voordeel boven een bandrecorder is, dát de melodie in elk gewenst tempo kan worden gespeeld, wat bij het instuderen gemakkelijk kan zijn. Wat dacht u van de Melodiant als muziekleraar? Op het display verschijnt een noot, de leerling moet nu proberen deze noot te zingen of te fluiten. Later kan dan ter controle de luidspreker aangezet worden. Ook zal de Melodiant feilloos de moeilijkste ritmes voorspelen, naar behoefte langzaam of snel.
Maar hiermee zijn de mogelijkheden nog niet uitgeput. We kunnen de Melodiant zo snel laten spelen, dat de tonen afzonderlijk niet meer herkenbaar zijn. Hierdoor kunnen de meest vreemde geluidseffecten ontstaan.

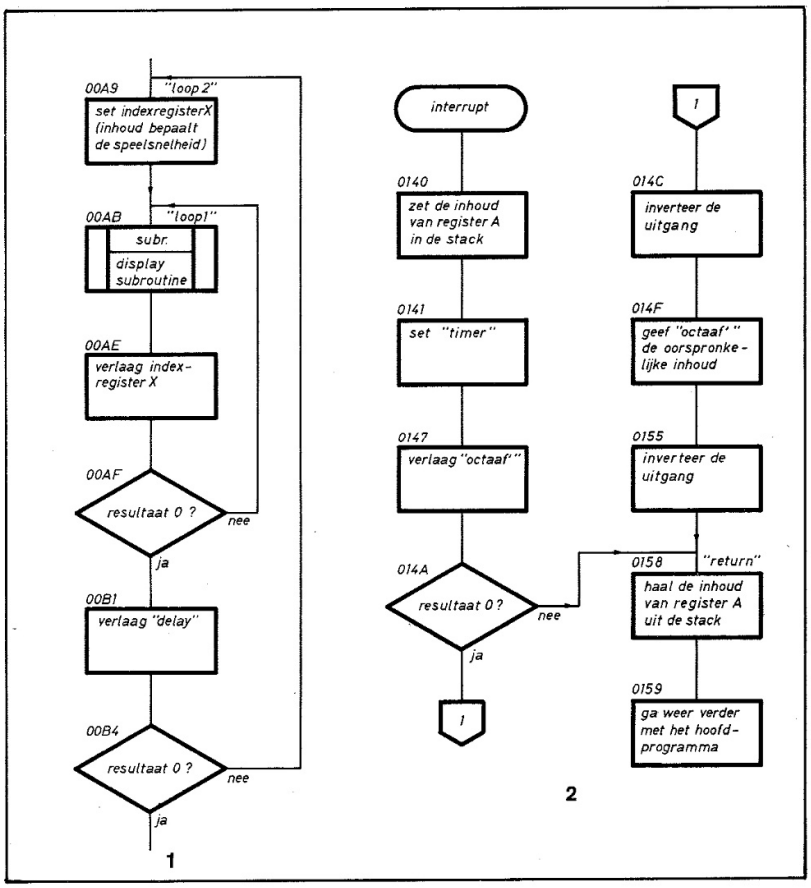
De flowchart en het programma
Bij de bespreking van de Melodiant maken we steeds gebruik van een flowchart en een programma. De flowchart geeft alle acties die de microprocessor moet verrichten overzichtelijk weer.
In het programma kunnen we daarentegen duidelijk zien welke instructie op welk adres staat. Bovendien hebben sommige geheugenplaatsen een naam gekregen, omdat wij een naam makkelijker kunnen onthouden dan een adres. In het programma (zie lijst 1)

treffen wij zes kolommen aan. De eerste kolom geeft het adres. De tweede kolom geeft de inhoud op die betreffende geheugenplaats. Voor het programmeren hebben we in principe genoeg aan deze twee kolommen. Als we het program ma echter willen volgen en begrijpen hebben we de andere kolommen ook nodig. In de derde kolom staat de naam die een geheugenplaats kan hebben, deze naam wordt ook wel 'label' genoemd. (De meeste adressen zijn naamloos.) De vierde kolom geeft in afgekorte vorm de instructie die door de microprocessor moet worden uitgevoerd, terwijl daarachter in de vijfde kolom (cursief) staat welke adresseer techniek is gebruikt. Een verklaring van de afkortingen en adresseertechnieken vindt u in de algemene beschouwing van de KIM 1. Onder de cursief gedrukte tekst staat steeds de naam (of het adres) van een geheugen waar de operand in staat. Alleen bij de immediate addressing staat hier de operand zelf. Het $ teken geeft hierbij aan dat het getal hexadecimaal geschreven is.
De laatste kolom geeft een toelichting op de diverse programmastappen. Deze kolom geeft ook een verbinding met de flow chart, omdat in de flowchart ongeveer dezelfde uitdrukkingen gebruikt worden. De flowchart geeft zoals gezegd een overzicht van de volgorde waarin verschillende programmadelen worden doorlopen (afb. 4). Boven elk blok is steeds een adres met daarbij eventueel een naam van dat adres ge geven. Dit adres geeft weer aan op welk stuk programma het betreffende blok betrekking heeft. Op deze manier is al tijd makkelijk elk stuk programma terug te vinden.

De werking

De KIM 1 is zo geprogrammeerd dat hij op de als uitgang geprogrammeerde aansluiting PAS een pulssignaal van de juiste frequentie genereert. Bij elke voort te brengen toon moet het programma de frequentie op deze uitgang de juiste waarde geven, hij moet de toon de juiste lengte geven, en ten slotte moet de naam van de noot op het display worden gezet. Nu kan de microprocessor ontzettend veel, maar hij kan niet meer dan één ding tegelijk. We gaan daarom als volgt te werk: Allereerst zorgt een klein stukje pro gramma ervoor dat één van de zes displays het goede symbool weergeeft. Steeds als dit stukje programma wordt doorlopen licht het volgende display op. (Het display wordt gemultiplext). De tijd die de microprocessor nodig heeft om dit stukje programma te door lopen (ruim 1 ms) wordt nu gebruikt als tijdseenheid voor de lengte van de te spelen toon. Deze display subroutine wordt, om een behoorlijke tijdsduur te kunnen krijgen, een aantal malen doorlopen. Dit wordt bereikt door een bepaald getal in het indexregister te zetten (afb. 1). Vervolgens wordt na het doorlopen van de displayroutine het indexregister met één verminderd. Zolang het resultaat hiervan geen '0' is, wordt steeds teruggesprongen naar de display subroutine. Op deze manier kan de display sub routine, afhankelijk van de begininhoud van het indexregister X, 1 tot 256 x worden doorlopen, zodat een tijd kan worden bereikt van ongeveer 1 tot 256 ms. Omdat deze tijd nog veel te kort is wordt het hele proces nog een keer herhaald. Nu echter niet met behulp van het indexregister X, maar met een geheugenplaats in het externe geheugen. Deze geheugenplaats noemen we 'delay' (adres 0111). We kunnen nu de eerste loop

weer 1 tot 256 x doorlopen en de lengte van de noot is nu met 2 loops te regelen van ongeveer 1 ms tot ruim 1 minuut. De geheugenplaats *delay' wordt steeds door het program ma gevuld met de gewenste lengte van de noot. Door de begininhoud van het indexregister X te wijzigen (op adres QOAA) kan de speelsnelheid worden geregeld. Het stukje programma wat bij deze loops hoort staat op de adressen OOAQ tot en met OOBS. Voor de duidelijkheid zijn deze adressen ook getekend in de flowchart van afb. 1.



1 De flowchart van de wachtcyclus 2. Dit gedeelte zorgt voor de lengte van de gespeelde noot.
2 De flowchart van het interrupt programma. Dit programma werkt geheel onafhankelijk van het hoofdprogramma.

Het maken van een toon

We hebben nu gezien hoe het program ma zorgt dat de toon de juiste lengte krijgt. Dat is echter niet genoeg. We moeten nl. ook nog een toon van de juiste frequentie maken. Het principe is heel eenvoudig. Voor elke noot is de trillingstijd T (=%) uitgerekend. Steeds als deze tijd verstreken is, wordt de uit gang van de Melodiant even '1' gemaakt. Op deze manier ontstaat een pulsspanning van de juiste frequentie op de uitgang. Er is echter een probleem. Gedurende de tijd dat de toon moet klinken is het programma

druk met het displayen van de naam van de noot en met het bepalen van de juiste lengte van de noot. Daarom wordt het maken van de toon ‘uitbesteed’.De KIM 1 heeft nl. een interval timer. Deze timer kan door het programma met een bepaalde waarde worden ge laden. De timer werkt vanaf dat mo ment geheel onafhankelijk van het programma. Na het verstrijken van de ge programmeerde tijd (in dit geval de trillingstijd T van de betreffende noot) wordt een ‘interrupt request’ gegenereerd. Dat wil zeggen, dat het lopende programma even onderbroken wordt en dat een tweede programma automatisch wordt gestart.
Dit tweede programma heeft een looptijd van ongeveer 50 us. De flowchart is getekend in afb. 2 terwijl het interrupt programma in lijst 2 is gegeven. Ook in afb. 2 zijn weer de programma adressen bijgetekend, zodat de lezer het programma gemakkelijk kan vol gen. Het interrupt programma begint met het wegzetten van de inhoud van register A. Wanneer het hoofdprogram ma weer gestart wordt moeten immers alle registers weer in hun oorspronke lijke staat worden gezet. Vervolgens wordt de timer weer gestart, zodat een trillingstijd T later opnieuw een interrupt kan ontstaan. Wat nog niet ter sprake is gekomen, is de manier waarop de tonen van lagere octaven gegenereerd worden. We gaan altijd uit van de trillingstijden van het hoogste octaaf. Lagere octaven worden dan gemaakt door die hoogste toon 2, 4 of 8 keer te delen. Hoeveel maal de frequentie ge deeld moet worden staat in het geheu gen ’octaaf’ ” (adres 0114). Afhankelijk van de inhoud van dit geheugen wordt op de output, elke keer dat het interrupt programma wordt doorlopen, een puls gezet ("octaaf’ ” = 1), of gebeurt dit slechts één op de 2, 4 of 8 keer (“octaaf ” = 2, 4 of 8).

Als we even teruggaan naar afb. 2, dan zien we dat ”octaaf’ * verminderd wordt met 1 (adres 0147). Alleen als het resultaat hiervan '0’ is, wordt de output geïnverteerd, "octaaf’ ” wordt weer gevuld met de oorspronkelijke inhoud en: de output wordt weer geïnverteerd. Op deze wijze ontstaat eens in de 1, 2, 4 of 8 perioden tijden een puls van 12 us aan de uitgang. Wanneer dit gebeurd is wordt de inhoud van register A weer teruggezet, zodat het hoofdprogram ma weer gewoon vervolgd kan worden. De laatste opdracht ‘return from Interrupt’ zorgt ervoor dat het hoofdprogramma weer wordt vervolgd alsof er niets aan de hand is geweest.

Hoe de muziek in het geheugen staat
We hebben nu gezien hoe onze computer de lengte en de frequentie van de toon bepaalt. Het programma doet dit echter niet op eigen houtje. Hij kijkt steeds in een speciaal hiervoor gereserveerd stuk geheugen welke noot gespeeld moet worden en hoe lang. Dit stuk geheugen loopt vanaf adres 0200 tot en met O3FF.
De totale ruimte voor de melodie is dus 512 bytes. Elke toon neemt 2 bytes in beslag, zodat 256 tonen (of rusten) geprogrammeerd kunnen worden. De code is nu als volgt gekozen: Het eerste byte codeert de toonhoogte, het twee de byte de toonlengte. Voor de toonhoogte is met de code uitgegaan van de ‘gewone’ toonladder van C.

Rust	0	stop	8
C	1	Cis	9
D	2	Dis	A
E	3	Eis	B
F	4	Fis	C
G	5	Gis	D
A	6	Ais	E
B	7	Bis	F

Voor de verhogingen wordt bij de code van de betreffende noot 8 opgeteld (hexadecimaal). Dit heeft verschillen de voordelen. Allereerst is het bij het intypen van de muziek gemakkelijk om dat nu de verhogingen steeds 2 knopjes hoger liggen dan de oorspronkelijke noot. We willen b.v. een Fis intypen. Mensen die een beetje in de muziek zitten weten dat de F de 4e noot uit de toonladder is, de Fis ligt nu 2 knopjes hoger. In het begin is het even wennen, maar het coderen van een melodie gaat na verloop van tijd steeds sneller. We hebben nu van het eerste byte alleen de eenheden gebruikt voor het coderen van de noot in het hoogste octaaf. Al eerder hebben we vermeld dat de lagere octaven worden verkregen door deling. Hoeveel maal gedeeld moet worden kunnen we nu aangeven op de 16tallen van het eerste byte (16tallen omdat we werken in het 16tallig stelsel). De octaven geven we dus aan met een 1 voor het hoogste octaaf, en 2, 4 of 8 voor lagere octaven. Een E in het laagste octaaf wordt dus: 83, een F in het op één na hoogste octaaf wordt 24, een G in het hoogste octaaf wordt 15.
Er worden verder nog 2 belangrijke codes gebruikt, nl. O geeft een rust. Hierbij wordt de lengte van de rust op dezelfde manier bepaald als bij een noot. De code 8 is een stopteken, en moet aan het einde van de melodie staan.
De lengte van de noot staat in het 2e byte. Hier kunnen we waarden invullen van 01 tot FF. Het is heel eenvoudig om de lengte te coderen. Stel dat de snelste noot is een 1/32 noot (3 vlaggetjes aan de stok). Deze noot geven we de waarde 01. Een 1/16 noot wordt nu 02, 1/8 noot wordt 04, 1/4 noot wordt 08, 1/2 noot wordt 10 (hexadecimaal is 08 de helft van 10!!!) en een hele noot wordt 20. Ook ingewikkelde ritmes met triolen zijn te programmeren door elke maat in een voldoende aantal stukjes te hakken. We leggen er nog even de nadruk op, dat deze geheugenruimte (0200-03FF) géén deel uitmaakt van het programma. Dit stuk geheugen moet worden beschouwd als de 'bladmuziek' waar vanaf de computer speelt.

Het decoderen van de muziek
Alvorens een noot ten gehore kan worden gebracht moeten de codes door de computer worden vertaald. De ver schillende grootheden van een bepaal de noot worden steeds op een paar vaste geheugenplaatsen gezet nl.:

Naam	adres
T	0110
delay	0111
octaaf	0112
octaaf’	0114
display	011C
hoogte	0120
toon	0122
I	0123
S	0124

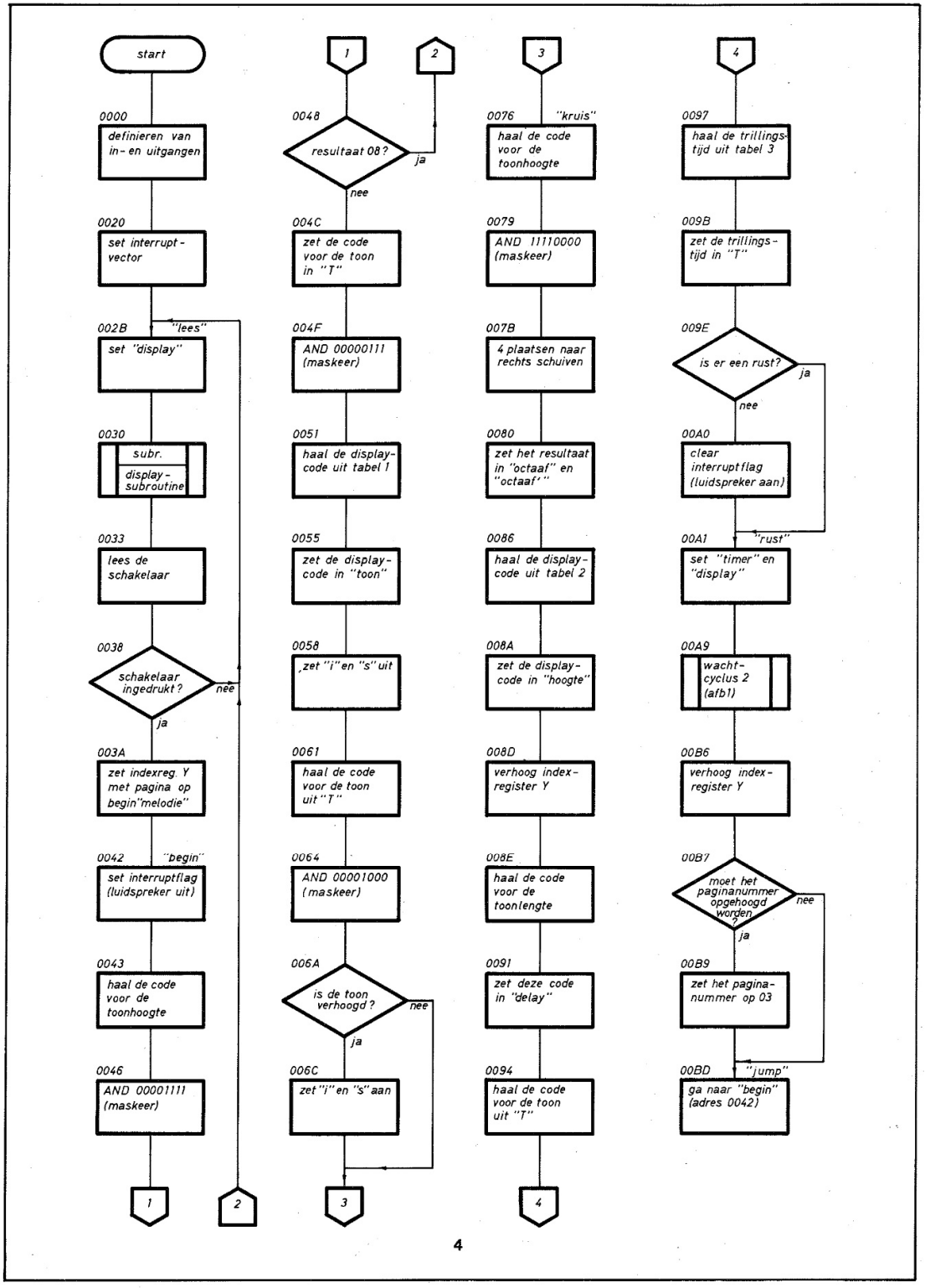
In ’T’ komt uiteindelijk de trillingstijd te staan, in ‘delay’ de lengte van de noot, in octaaf” en "octaaf’ ” het aan tal delingen. Hierbij wordt octaaf’ ” tijdens het programma steeds met 1 verminderd tot 0, "octaaf” blijft zijn waarde houden.
In display’ komt tijdens het ten gehore brengen van een melodie de waarde C te staan. Met dit getal wordt een reeks geheugenplaatsen aangeduid waarin reeds in gecodeerde vorm de te displayen grootheden staan. Twee van die geheugenplaatsen krijgen tijdens de muziek steeds een andere waarde.
Dit zijn de geheugenplaatsen ‘hoogte’ en ’toon’ waarin resp. de displaycode voor het octaaf staat en de display code voor de toon. De flowchart uit afb. 4 laat het hele hoofdprogramma zien.
Het laatste stuk hiervan (vanaf adres 00a9) is het eigenlijke muziekprogram ma en werd al behandeld bij ’de werking’. Het decodeergedeelte begint op adres 0042. Aan de hand van de flowchart (afb. 4) zullen we de verschillende handelingen doornemen.
Het eerste wat het programma doet is het zetten van de interruptflag. Dit houdt in, dat interrupts vanaf nu ge negeerd worden. Er zullen ook geen pulsen meer op de uitgang komen (de pulsen werden gegenereerd door het interruptprogramma). De vorige toon wordt dus als het ware uitgezet. Ver volgens wordt het eerste byte van de nieuwe toon in register A gezet. Index register Y houdt bij welke toon aan de beurt is. Er is hier gebruik gemaakt van de indirect indexed addressing, dit wil zeggen dat het 2e byte van de instructie (adres 0044) een adres op pagina 00 geeft (00EA). De inhoud van dit adres wordt opgeteld bij de inhoud van het indexregister Y. Het resultaat van deze optelling is het uiteindelijke adres op een pagina die in dit geval op adres 00EB staat. In ons geval komt het hierop neer: Het indexregister Y geeft het adres op pagina 02 (de inhoud van 00EA is 00, de inhoud van 00EB is 02). Na elke cyclus wordt het indexregister Y verhoogd, zodat steeds de goede toon wordt opgehaald. Wanneer de toonhoogtecode in register A staat wordt de ’AND’ bewerking uitgevoerd met het binaire getal 00001111. Dit houdt in, dat na deze bewerking de 16tallen O zijn gemaakt en dat alleen de eenheden over zijn gebleven. De code die het octaaf aangeeft is dus verdwenen. (Een 0 in een inpoort geeft altijd een 0 aan de uitgang.) Ver volgens wordt de overgebleven code vergeleken met 08. Wanneer het over gebleven getal inderdaad 8 was (stop code) zal het

Z-bit in het statusregister ‘1’ worden en zal naar de ’ready’ cyclus worden gesprongen. Wanneer er geen stopcode stond gaan we gewoon door en wordt de code van de noot in geheugen T (adres 0110) gezet. Later wordt deze code dan vertaald in een trillingstijd. Vervolgens wordt er weer een AND bewerking uitgevoerd, nu met 00000111 (binair). Het gevolg hiervan is dat ook de informatie weg is, die aan geeft of de noot al dan niet verhoogd was. De overblijvende code kan nu van 0 tot 7 lopen en staat voor de letters C t/m B. Met deze code gaan we de betreffende letter op het display zet ten. De displayroutine kijkt voor het displayen van deze letter naar geheugenplaats ‘toon’. Om een bepaalde let ter op het display te krijgen moeten we de goede code op deze geheugen plaats zetten. De code voor ’C’ is b.v. 39. We moeten dus de codes 0 t/m 7 omzetten in voor het display begrijpeliike codes. Hiervoor hebben we in het computergeheugen een tabel gemaakt waarin deze codes staan. De tabel loopt van adres 012F tot en met adres 0136 (zie lijst 3). Hoe krijgen we nu de goede code?
Wel, hiertoe maken we gebruik van de indexed addressing. Allereerst zetten we de code die in register A staat in het indexregister X (adres 0051, zie het programma). Op adres 0052 wordt nu de inhoud van een geheugenplaats in register A gezet. Het adres van deze geheugenplaats is de som van de inhoud van het indexregister X en 012F. Op de geadresseerde geheugenplaats staat nu dus de gewenste code. Deze code wordt eerst in register A gezet (adres 0052), en vervolgens in de eindbestemming ‘toon’ (adres 0122). Als voorbeeld zullen we als noot de Dis nemen in het laagste octaaf. De code in het geheugen is: 8A. Op adres 0046 (afb. 4 en lijst 1) wordt de ’AND’ bewerking uitgevoerd. Na deze bewerking is de overgebleven code OA, deze code wordt in 'T’ gezet (adres 004C). Nu wordt voor de tweede maal de AND bewerking uitgevoerd, en er wordt weer een bit 0’ gemaakt. Van 0A blijft nu slechts 02 over. Deze 02 is de code voor ’D’. Op adres 0051 wordt deze 02 in het indexregister X gezet. Op adres 0052 wordt de inhoud van adres 02 012F 0131 via register A op geheugenplaats ‘toon’ gezet, met het gevolg dat een ’D’ op het display verschijnt. Het achtervoegsel ’is’ wordt door het hierop volgende stukje programma verzorgd. Allereerst wordt er ’00’ gezet op de geheugenplaatsen ‘i’ en ’s’. Deze geheugenplaatsen worden door de displayroutine gebruikt voor het 4e en 5e display.
Op deze displays moeten bij halve tonen 1 de letters ‘is’ verschijnen. Door een ’0’ in deze geheugens te zetten wordt het display gedoofd. We gaan er dus aanvankelijk van uit dat we met een hele noot te maken hebben. Op adres 0061 wordt de code (in ons voorbeeld 0A) | die in ’T’ stond weer in register A gezet. Opnieuw wordt de AND bewerking uitgevoerd, nu worden alle bits op een na ’0' gemaakt. Het ene bit dat overblijft is ’1’ bij een verhoogde toon, en ’0’ bij een nietverhoogde toon. In afb. 4 (adres 006A) zien we dat indien er sprake is van een verhoogde toon op de adressen 0123 en 0124 de code gezet wordt voor resp. dei ende s. Indien er geen verhoging was wordt dit stuk overgeslagen, zodat de betreffende displays donker blijven. De benodigde codes voor het displayen van de noot zijn nu verzorgd. i Vervolgens gaan we de code voor het octaaf decoderen. Eerst wordt weer de complete code uit het geheugen gehaald (adres

0076). Via een AND bewerking worden nu de eenheden afgevoerd (deze AND bewerking wordt ook wel ’maskeren’ genoemd). Van een code 8A zou nu dus 80 overblijven. Vervolgens schuiven we 4 x naar rechts waardoor 16tallen op de plaats van de eenheden komen te staan (80 wordt nu dus 08). De over gebleven code wordt opgeslagen in "octaaf” en ”octaaf’”. Vervolgens wordt ook nu weer in een tabel opgezocht welke displaycode gebruikt moet worden. Deze tabel staat in het geheugen op de adressen 0137 tot en met 018F. De displaycode wordt weggezet op adres 0120 (‘hoogte’) en de display routine zorgt dat op het eerste display een aanduiding van het octaaf komt. We zijn inmiddels op adres 008D. Hier wordt het indexregister Y opgehoogd, zodat we nu het 2e byte van de geprogrammeerde noot zien. Ook deze code wordt naar binnen gehaald en op geheugenplaats ‘delay’ gezet. Deze code hoeft verder niet gedecodeerd te worden. We moeten nog wel de code die nu in ’T’ staat (een getal tussen 00 en 0F) omzetten in een trillingstijd. Ook dit gebeurt weer m.b.v. een tabel. De tabel staat op de adressen 0100 t/m 010F (lijst 3) en het decoderen gebeurt op de nu wel bekende manier. De ge tallen in de tabel zijn hexadecimaal en geven een trillingstijd als veelvoud van 8 us. De timer van de KIM 1 is nl. door het programma afgesteld op een teltijd van 8 us. Op adres 09E wordt ten slotte gekeken of we te doen hebben met een rust. In ’T’ staat dan de waarde ‘00’. Indien er geen rust is, wordt de interruptflag ‘0’ gemaakt, zodat er weer interrupts kunnen ontstaan. Hierdoor zal de nieuwe toon door de luidspreker klinken. Indien er wel een rust is, zal de interruptflag ’1’ blijven, zodat het stil blijft. Omdat bij de eerste toon pas een interrupt kan ontstaan nadat de timer is ingesteld, wordt op adres 00A1 de timer één keer gestart.
De volgende keren wordt dit verzorgd door het interruptprogramma. Als laatste stap van dit decodeergedeelte wordt geheugenplaats ‘display’ gevuld met de waarde 1C, waardoor de displayroutine het juiste stukje geheugen zichtbaar maakt. Vanaf dit punt begint het eigenlijke programma, dat reeds bij de werking’ werd besproken.

Het begin van het programma

Bijna alle programma’s moeten beginnen met het definiëren van verschillende zaken. Omdat dit het minst interessante deel (niet minder belangrijk!) is, behandelen we dit kort. Allereerst worden de verschillende aan sluitingen van de KIM1 gedefinieerd als in of output. Hiertoe wordt in een speciaal data direction register een ’1’ geschreven als de overeenkomstige aansluiting als output moet dienen en een '0’ als het een ingang moet zijn. Op adres 20 wordt de interruptvector gedefinieerd. Met andere woorden: hier wordt aan de microprocessor verteld dat hij bij een interrupt moet springen naar adres 0140, waar het interruptprogramma begint.



De wachtcyclus 1

Op adres 002B begint de wachtcyclus 1. Allereerst wordt er op geheugenplaats ‘display’ (adres 011C) de waarde ’12’ gezet. Hierdoor zal de displayroutine niet een gespeelde noot zichtbaar maken, maar het woord ’READY’, dat in gecodeerde vorm staat op de

geheugenplaatsen 0116 t/m 01 1B (lijst 3). Ver volgens wordt naar de displaysubroutine gesprongen, die uiteindelijk zorgt dat het woord 'READY' op het display verschijnt. Op adres 0033 worden alle aansluitingen PA® t/m PA7 gelezen. De bedoeling is, dat we alleen de toestand van de startschakelaar lezen. Daarom worden alle andere bits weer naar '0' gedwongen (door de AND bewerking). Zolang de schakelaar niet is ingedrukt, zal dit stukje programma steeds worden herhaald. Bij een start signaal wordt, nadat indexregister Y op '0' is gezet, en de pagina op 02, (de melodie begint op adres 0200) begonnen met het decodeergedeelte.

De display subroutine

Elke keer dat naar deze subroutine wordt gesprongen, wordt slechts 1 van de 6 displays aangezet. Doordat bij elke sprong naar deze subroutine steeds een volgend display genomen wordt zien we 6 oplichtende displays. In werkelijkheid worden de displays dus gemultiplext. Om het programma van de subroutine helemaal te snappen, zou eerst uitgebreid moeten worden uitgelegd hoe de displays op de KIM1 zijn aangesloten. Omdat dit buiten het bestek van dit artikel valt zullen we alleen de flowchart even bespreken. In een later artikel zal dan dieper op de displays en het toetsenbord ingegaan worden. De flowchart van deze subroutine staat in afb. 5. We beginnen met het opbergen van de inhoud van register A en indexregister Y, zodat deze registers door de subroutine vrij kunnen worden gebruikt. Vervolgens wordt er een teller opgehoogd, die zorgt dat nu het volgende display wordt aangezet. Als deze teller op 7 staat (display 7 be staat niet) wordt de teller teruggezet op 1.
Vervolgens wordt gekeken op welke geheugenplaats de displaycode staat. Dit wordt bepaald aan de hand van.de inhoud van geheugen 'display' en aan de hand van de eerder genoemde teller. Elk display moet immers een ander symbool weergeven. Vervolgens wordt deze displaycode aan het display toegevoerd, zodat het gewenste symbool op het display verschijnt. Hierna begint een wachtcyclus, zodat de gewenste vertraging van ongeveer 1 ms ontstaat. Deze vertraging wordt gemaakt met een loop. Het principe hiervan is reeds besproken bij 'de werking'. Als laatste worden de registers A en Y weer in hun oorspronkelijke staat gebracht en wordt weer teruggesprongen naar het hoofdprogramma.

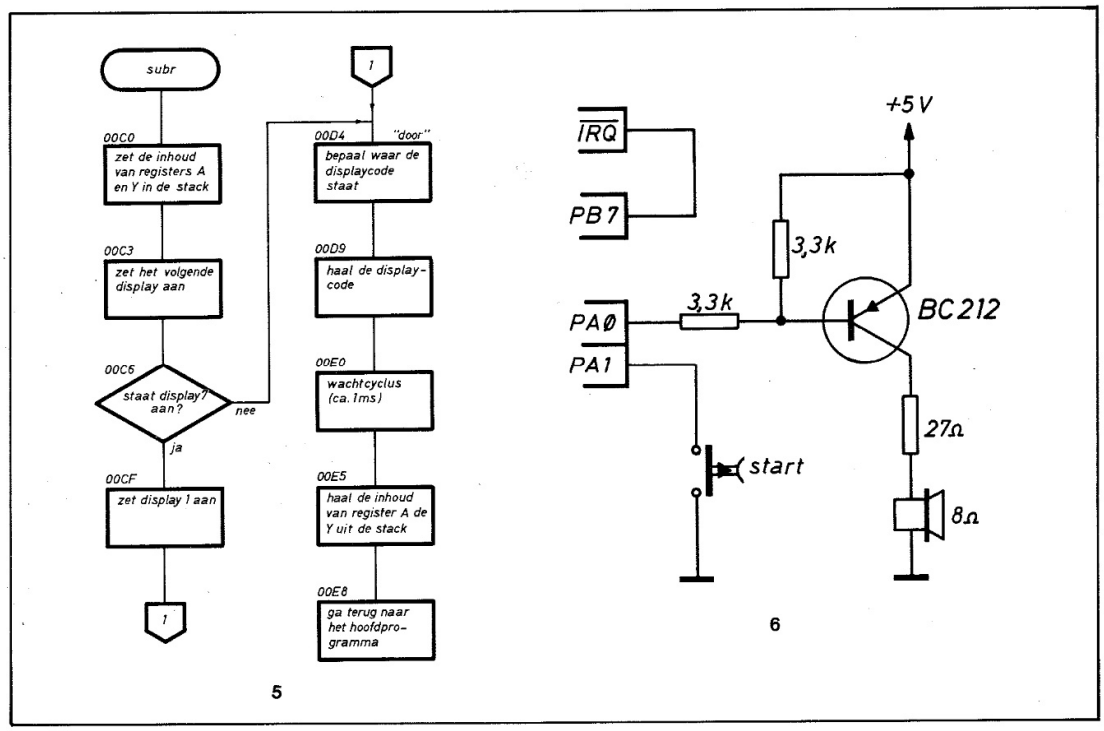
Het totale programma

We hebben nu het hele programma stukie bij beetje besproken. We zijn bewust niet bij het begin begonnen, maar bij de kern van het programma.
Dit heeft als voordeel dat het programma logisch opgebouwd kan worden.
Voor het overzicht zullen we de delennog een keer in de juiste volgorde noemen.

Het **hoofdprogramma** bestaat uit:

a.Het begin op de adressen 0000 tot 002A. Dit deel definieert in- en uitgangen en de interruptvector (afb. 4 en lijst 1)
b. De wachtcyclus 1, op de adressen 002B tot 0039. Dit deel wordt door lopen zolang de

startknop niet is in gedrukt. Tijdens deze cyclus wordt steeds naar de displaysubroutine gesprongen zodat het woord 'READY' op de display verschijnt (afb. 4 en lijst 1).
c. Het decodeergedeelte op de adressen 0042 tot 00A8. Bij de aanvang van elke toon wordt dit gedeelte doorlopen om verschillende geheugenplaatsen te vullen met gegevens van de toon (lengte, trillingstijd, displaycodes) (afb. 4 en lijst 1).
d. De wachtcyclus 2 op de adressen 00A9tot 00B8. Dit stuk verzorgt de lengte van de noot en roept regelmatig de displaysubroutine aan, die op zijn beurt de naam van de noot op het display zet (afb. 1 en lijst 1). e. De displaysubroutine op de adressen 00C0 tot 00E8. Deze subroutine maakt de displaycodes welke in het geheugen staan zichtbaar op de displays (afb. 5 en lijst 1). Het interruptprogramma (afb. 2 en lijst 2). Dit is een geheel onafhankelijk programma, dat wordt gestart door de timer. Dit programma genereert de pulsen aan de uitgang en start op zijn beurt weer de timer.



De externe componenten

Hoewel het een echte programmeur tegen de borst stuit, zullen we echt even de soldeerbout ter hand moeten nemen. We moeten nl. een luidsprekertje en een drukknopje op de aan sluitingen van de KIM 1 monteren. Ook moeten we de uitgang van de timer met de IRQ (interrupt request) ingang verbinden. Afb. 6 laat ons zien hoe een en ander moet worden aangesloten. Voor de luidspreker is nog een klein transistorversterkertje geplaatst, maar u kunt natuurlijk ook de Melodiant aansluiten op uw eigen (hifi) versterker.

Gebruiksaanwijzing

Nadat u alle onderdelen op de KIM 1 hebt aangesloten, en de getallen uit de liisten 1... 3 hebt ingetypt (de getallen uit de tweede kolom moeten op de adressen uit de eerste

kolom komen te staan) kan het programma worden gestart. We voeren adres 0000 in, en drukken vervolgens op RS (reset) en GO. Als alles goed is verschijnt nu het woord 'READY' op het display. Als we nu op het externe knopje drukken, zal de melodie (indien u die er al in had gezet) uit het luidsprekertje klinken. Wanneer u langzamer of sneller wilt, drukt u op ST (stop) en u voert adres 00AA in. De inhoud van dit adres moet groter worden indien u langzamer wilt, en kleiner indien u sneller wilt. Het opnieuw starten gebeurt weer op de zelfde manier, en de melodie zal door de Melodiant in elk gewenst tempo gespeeld worden. Wat gaat dit kosten? Tot slot nog een voor vele lezers be langrijke vraag: 'wat gaat dit kosten?' Als u al een KIM 1 heeft, dan zijn de kosien zeer laag. Als u geen rommel doosje heeft, en u moet de onderdelen nieuw kopen, zeg ongeveer f 10,. Als u nog geen KIM 1 heeft, is het niet eer lijk om nu fl 998, (voor abonnees) bij dit tientje op te tellen. Uw KIM 1 is immers geen Melodiant! De KIM 1 kan behalve Melodiant ook een klok, doka timer, schaakcomputer, rekenmachine enz. zijn.

Lijst 1 Het hoofdprogramma.

```
Hoofdprogramma
0000 A9     LDA  immediate  }
0001 01           $01       }  in/output def.
0002 8D     STA  absolute   }  PAD1
0003 01           PADD1     }
0004 17           1701      }
0005 A9     LDA  immediate  }
0006 7F           $7F       }  in/output def.
0007 8D     STA  absolute   }  PAD2
0008 41           PADD2     }
0009 17           1741      }
000A A9     LDA  immediate  }
000B 00           $00       }  begininhoud
000C 8D     STA  absolute   }  PAD2
000D 40           PAD2      }
000E 17           1740      }
000F A9     LDA  immediate  }
                              }
0010 1E           $1E       }  in/output def.
0011 8D     STA  absolute   }  PBD2
0012 43           PBDD2     }
0013 17           1743      }
0014 A9     LDA  immediate  }
0015 08           $08       }  begininhoud
0016 8D     STA  absolute   }  PBD2
0017 42           PBD2      }
0018 17           1742      }
0019 78     SEI  implied    }  luidspreker uit
001A EA     NOP  implied
001B EA     NOP  implied
001C EA     NOP  implied
001D EA     NOP  implied
001E EA     NOP  implied
001F EA     NOP  implied

0020 A9     LDA  immediate  }
0021 40           $40       }
0022 8D     STA  absolute   }
0023 FE           17FE      }
0024 17                     }  set interrupt
0025 A9     LDA  immediate  }  vector
0026 01           $01       }
0027 8D     STA  absolute   }
0028 FF           17FF      }
0029 17                     }

002A EA     NOP  implied
002B A9 lees LDA immediate  }
002C 12           $12       }
002D 8D     STA  absolute   }  set 'display'
002E 1C           display   }
002F 01           011C      }

0030 20     JSR  absolute   }  spring naar
0031 C0          subr.      }  displ.subroutine
0032 00          00C0       }
0033 AD     LDA  absolute   }
0034 00          PAD1       }
0035 17          1700       }  lees de schakelaar
0036 29     AND  immediate  }  en spring naar
0037 02          $02        }  'lees' indien niet
0038 D0     BNE  relative   }  ingedrukt
0039 F1          lees       }
003A A0     LDY  immediate  }  zet Y aan het begin
003B 00          $00        }  van de melodie
003C EA     NOP  implied
003D EA     NOP  implied
003E A9     LDA  immediate  }
003F 02          $02        }
                            }  pagina op 02
0040 85     STA  zero page  }
0041 EB          00EB       }
0042 78 begin SEI implied   }  luidspr. uit
0043 B1     LDA  (ind), Y   }  haal de
0044 EA          melody     }  toonhoogte
0045 EA     NOP  implied
0046 29     AND  immediate  }
0047 0F          $0F        }  maskeer
0048 C9     CMP  immediate  }
0049 08          $08        }  was er een
004A F0     BEQ  relative   }  stopteken?
004B DF          lees       }
004C 8D     STA  absolute   }  code voor de
004D 10          T          }  toon in T
004E 01          0110       }
004F 29     AND  immediate  }  maskeer
                            }
0050 07          $07        }
0051 AA     TAX  implied    }
0052 BD     LDA  abs, X     }  haal de display
0053 2F          tabel 1    }  code uit de tabel
0054 01          012F       }
```

```
0055 8D        STA  absolute  } display code
0056 22             toon     } in 'toon'
0057 01             0122
0058 A9        LDA  immediate }
0059 00             $00       }
005A 8D        STA  absolute  }
005B 23             i         } zet 'i' en 's'
005C 01             0123      } uit
005D 8D        STA  absolute  }
005E 24             s         }
005F 01             0124      }

0060 EA        NOP  implied
0061 AD        LDA  absolute  } code voor de
0062 10             T         } toon in A
0063 01             0110
0064 29        AND  immediate } maskeer
0065 08             $08       }
0066 EA        NOP  implied
0067 EA        NOP  implied
0068 EA        NOP  implied
0069 EA        NOP  implied
006A F0        BEQ  relative  } is er een kruis?
006B 0A             kruis     }
006C A9        LDA  immediate }
006D 06             $06       }
006E 8D        STA  absolute  }
006F 23             i         }

0070 01             0123      } zo ja,
0071 A9        LDA  immediate } zet 'i' en 's' aan
0072 6D             $6D       }
0073 8D        STA  absolute  }
0074 24             s         }
0075 01             0124      }
0076 B1 kruis  LDA  (ind), Y  } haal de
0077 EA             melody    } toonhoogte
0078 EA        NOP  implied
0079 29        AND  immediate } maskeer
007A F0             $F0       }
007B 4A        LSR  accum.    }
007C 4A        LSR  accum.    } schuif de
007D 4A        LSR  accum.    } 16-tallen
007E 4A        LSR  accum.    } naar de eenheden
007F EA        NOP  implied

0080 8D        STA  absolute  }
0081 12             octaaf    }
0082 01             0112      } set octaaf
0083 8D        STA  absolute  } en octaaf'
0084 14             octaaf'   }
0085 01             0114      }
0086 AA        TAX  implied   }
0087 BD        LDA  abs, X    } haal de
0088 37             tabel 2   } displaycode
0089 01             0137      } uit de tabel
008A 8D        STA  absolute  }
008B 20             hoogte    } displaycode
008C 01             0120      } in 'hoogte'
008D C8        INY  implied   }
008E B1        LDA  (ind), Y  } haal de
008F EA             melody    } toonlengte

0090 EA        NOP  implied
0091 8D        STA  absolute  } code voor de
0092 11             delay     } lengte in 'delay'
0093 01             0111      }
0094 AD        LDA  absolute  } code voor de
0095 10             T         } toon in A
0096 01             0110      }
0097 AA        TAX  implied   }
0098 BD        LDA  abs, X    } haal de
0099 00             tabel 3   } trillingstijd
009A 01             0100      } uit de tabel
009B 8D        STA  absolute  }
009C 10             T         } trillingstijd
009D 01             0110      } in 'T'
009E F0        BEQ  relative  } is er een rust?
009F 01             rust      }
```

```
00A0 58        CLI  implied   } luidspreker aan
00A1 8D rust   STA  absolute  }
00A2 0D             timer     } set timer (8 µs)
00A3 17             170D      }
00A4 A9        LDA  immediate }
00A5 1C             $1C       }
00A6 8D        STA  absolute  } set 'display'
00A7 1C             display   }
00A8 01             011C      }
00A9 A2 loop 2 LDX  immediate }
00AA 20 snelh.      $20       }
00AB 20 loop 1 JSR  absolute  }
00AC C0             subr.     }
00AD 00             00C0      }
00AE CA        DEX  implied   } begin
00AF D0        BNE  relative  } wachtcyclus 2
               2 loops        }
00B0 FA             loop 1    }
00B1 CE        DEC  absolute  }
00B2 11             delay     }
00B3 01             0111      }
00B4 D0        BNE  relative  }
00B5 F3             loop 2    }
00B6 C8        INY  implied   } indexreg. Y ophogen
00B7 D0        BNE  relative  } moet het paginanr.
00B8 04             jump      } opgehoogd worden?
00B9 A9        LDA  immediate } verhoog het
00BA 03             $03       } paginanummer van
00BB 85        STA  zero page } 02 naar 03 terug
00BC EB             00EB      } naar het begin
00BD 4C jump   JMP  absolute  }
00BE 42             begin     }
00BF 00             0042      }

00C0 48 subr.  PHA  implied   } begin display
00C1 98        TYA  implied   } subroutine
00C2 48        PHA  implied   }
00C3 EE        INC  absolute  } hoog de teller
00C4 42             PBD2      } op voor het
00C5 17             1742      } volgende displ.
00C6 AD        LDA  absolute  }
00C7 42             PBD2      }
00C8 17             1742      }
00C9 29        AND  immediate } display 7?
00CA 1E             $1E       }
00CB C9        CMP  immediate }
00CC 14             $14       }
00CD D0        BNE  relative  } zo ja, naar
00CE 05             door      } 'door'
00CF A9        LDA  immediate }
                              } zo nee,
00D0 08             $08       } zet teller op
00D1 8D        STA  absolute  } display 1
00D2 42             PBD2      }
00D3 17             1742      }
00D4 4A door   LSR  accum.    }
00D5 18        CLC  implied   }
00D6 6D        ADC  absolute  } bepaal de
00D7 1C             display   } geheugenplaats
00D8 01             011C      }
00D9 A8        TAY  implied   }
00DA B9        LDA  abs, Y    }
00DB 00             0100      }
00DC 01             displaycode naar
00DD 8D        STA  absolute  } het display
00DE 40             PAD2      }
00DF 17             1740      }

00E0 A0        LDY  immediate }
00E1 FF             $FF       }
00E2 88 loop 3 DEY  implied   } wachtcyclus
00E3 D0        BNE  relative  }
00E4 FD             loop 3    }
00E5 68        PLA  implied   } zet de registers
00E6 A8        TAY  implied   } weer goed, en
00E7 68        PLY  implied   } spring terug
00E8 60        RTS  implied   }
```

Lijst 2 Het interruptprogramma.

```
Interruptprogramma
0140 48          PHA  implied     save A
0141 AD          LDA  absolute  }
0142 10               T         }
0143 01               0110      }  set timer
0144 8D          STA  absolute  }  (8 µs)
0145 0D               timer     }
0146 17               170D      }
0147 CE          DEC  absolute  }
0148 14               octaaf'   }  verminder octaaf'
0149 01               0114      }  met 1
014A D0          BNE  relative  }
014B 0C               return    }  resultaat 0?
014C EE          INC  absolute  }
014D 00               PAD1      }  Zo ja, inverteer
014E 17               1700      }  uitgang
014F AD          LDA  absolute  }
0150 12               octaaf    }
0151 01               0112      }  set octaaf'
0152 8D          STA  absolute  }
0153 14               octaaf'   }
0154 01               0114      }
0155 EE          INC  absolute  }
0156 00               PAD1      }  inverteer uitgang
0157 17               1700      }  opnieuw
0158 68 return   PLA  implied     A terug zetten
0159 40          RTI  implied     Terug naar het
                                  hoofdprogramma
```

Lijst 3 De door het programma gebruikte geheugenplaatsen.

```
De door het programma gebruikte geheugenplaatsen

00EA 00                beginadres van de melodie
00EB 02                paginanummer van de melodie

0100 00 tabel 3  rust  }
0101 EF          C     }
0102 D5          D     }
0103 BD          E     }
0104 B3          F     }
0105 9F          G     }  deze tabel geeft
0106 8E          A     }  de trillingstijd van elke noot
0107 7E          B     }  uit het hoogste octaaf.
0108 00          stop  }  Andere getallen geven een
0109 E1          Cis   }  andere stemming
010A C9          Dis   }
010B B3          Eis   }
010C A9          Fis   }
010D 96          Gis   }
010E 86          Ais   }
010F 77          Bis   }

0110     T             }
0111     delay         }  deze geheugens worden
0112     octaaf        }  door het programma
0113                   }  gevuld met gegevens
0114     octaaf'       }  over de te spelen toon
0115
0116 5C          =     }
0117 31          R     }  deze geheugens bevatten de
0118 79          E     }  displaycode voor het woord
0119 77          A     }  'READY', andere codes geven
011A 5E          D     }  andere letters.
011B 6E          Y     }
011C     display       }  geeft aan welk blok van
011D                   }  6 geheugens moet worden
011E                   }  gedisplayd
011F

0120     hoogte        }  displaycode van het octaaf
0121 00                }  display 2 uit
0122     toon          }  displaycode van de toon
0123     i             }  'i' of uit
0124     s             }  's' of uit
0125 00                }  display 6 uit

012F 00 tabel 1  uit   }
0130 39          C     }  displaycodes voor de
0131 5E          D     }  naam van de noot. De
0132 79          E     }  juiste code wordt door
0133 71          F     }  het programma opgezocht
0134 3D          G     }  en op 'toon' (0122)
0135 77          A     }  gezet.
0136 7C          B     }

0137 00 tabel 2  uit   }
0138 01          1     }  displaycodes voor de
0139 41          2     }  hoogte van het octaaf
013A 00          uit   }  De juiste code wordt
013B 48          4     }  door het programma
013C 00          uit   }  opgezocht en op
013D 00          uit   }  'hoogte' (0120) gezet.
013E 00          uit   }
013F 08          8     }

1700     PAD1          Op PAD1 is de luidspreker
                       en de schakelaar aangesloten
1701     PADD1         Richtingsregister voor PAD1
170D     timer         Ingestelde tijd = inhoud x 8 µs
1740     PAD2          De inhoud van PAD2 komt
                       op het display te staan
1741     PADD2         Richtingsregister voor PAD2
1742     PBD2          Bepaalt welke van de 6
                       displays oplicht
1743     PBDD2         Richtingsregister voor PBD2
```