Besturing van een flexibele assemblage en lascel met behulp van MAP 3.0 en MMS

Scheffer, F.

Award date:
1992

Faculteit Elektrotechniek
Vakgroep meten en regelen

Besturing van een flexibele assemblage en lascel met behulp van MAP 3.0 en MMS

door F. Scheffer

Rapport van het afstudeerwerk
uitgevoerd van augustus 1991 tot april 1992
in opdracht van prof. ir. F.J. Kylstra
onder begeleiding van ir. C.A.M. van den Brekel.
Eindhoven, 21 april 1992.

Scheffer F. Besturing van een flexibele assemblage en lascel met behulp van MAP 3.0 en MMS.

Afstudeerrapport, vakgroep ER, Faculteit Elektrotechniek, april 1992.

De vakgroep ER van de faculteit Elektrotechniek werkt onder andere aan het FALC-project. Met de FALC wordt geprobeerd om een 'dunbemande' automatische lascel te realiseren. De bijdrage van de vakgroep ER ligt in de besturing van de lascel. De cel wordt bestuurd aan de hand van het CAM-reference model. Een groot deel van de onderdelen van de besturing is in verschillende apparaten ondergebracht. De communicatie vindt, op twee verbindingen na, plaats via MAP/MMS. De verschillende onderdelen van de besturing worden aangestuurd met recepten. Deze recepten kunnen als toestandstabellen worden ingevoerd in het systeem. De toestandstabellen worden in een onvertaalde vorm geïnterpreteerd. In de lascel bestaat het hoogste besturingsnivo uit de operator, die de orders voor de cel op een voor hem vriendelijke manier kan in- en uitvoeren via een user-interface.

Scheffer F.: Controlling a flexible assembly and arc welding cell using MAP 3.0 and MMS. [In Dutch]

M.Sc. Thesis, Measurement and Control Section ER, Electrical Engineering, Eindhoven Universisty of Technology, The Netherlands, Apr. 1992.

The Measurement and Control Section of the department of Electrical Engineering takes part in the FALC-project. The goal of the project is building a scarsely manned flexible assembly and arc welding cell. The section ER contributes to the project by building the control system. The control system is based on the CAM-reference model. A major part of the control levels of the model is placed in different devices. The communications between the devices use MAP/MMS. The different levels are controlled by recipes. These recipes can be added to the system in the form of state tables. The text of the state tables is executed by an interpreter. An operator has been installed as the highest control level in the cell. The operator can place and execute orders via a user-interface.

Inhoud

Bijlagen

1. Inleiding

De vakgroep ER van de faculteit Elektrotechniek werkt onder andere aan het FALC-project. Dit project wordt uitgevoerd in samenwerking met de vakgroep WPA van de faculteit Werktuigbouwkunde, DAF, Philips en TNO. Met de FALC, de flexibele assemblage en lascel, wordt geprobeerd een 'dunbemande' fabricagecel te realiseren. Deze cel heeft een besturing nodig, die zo veel mogelijk zelfstandig beslissingen neemt.

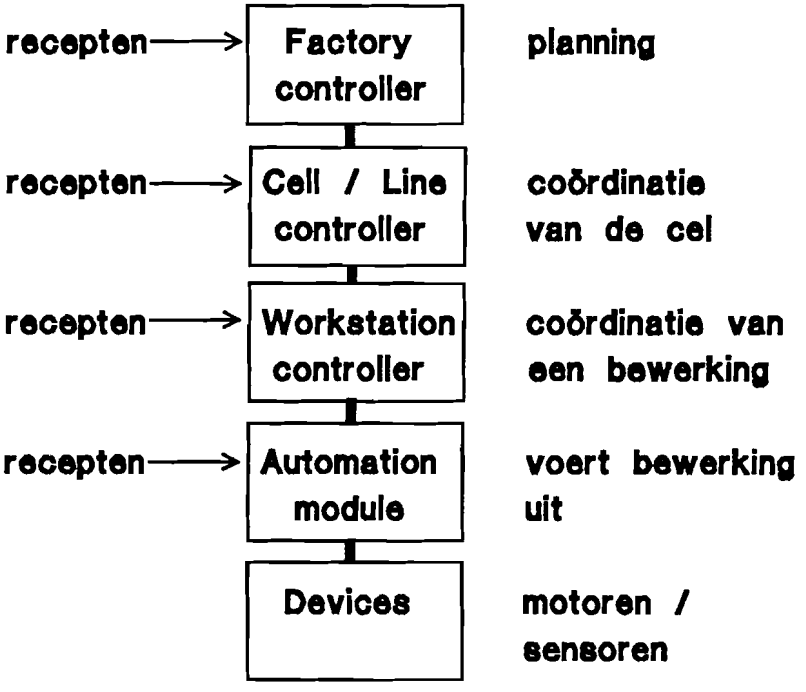
2. Opbouw van de FALC

De FALC is een fabricagecel, gespecificeerd in [1]. Het gedeelte, dat de produktie uitvoert, bestaat op dit moment uit een lasrobot en een transportsysteem [2]. Op papier staan nog een in- en uitvoerstation, die de produkten aan- en afvoeren. De laatste twee handelingen worden nu door de operator verricht. De verplaatsing over het transportsysteem en het lassen met de lasrobot worden automatisch uitgevoerd.

De besturing van de FALC is hiërarchisch opgebouwd. Dat wil zeggen, dat een ondergeschikte één opdrachtgever heeft en dat een opdrachtgever meerdere ondergeschikten kan hebben. De ondergeschikten mogen bovendien niet rechtstreeks met elkaar communiceren. De opbouw van de FALC hangt sterk met het gebruikte besturingsmodel samen.

2.1. Het besturingsmodel

De besturing is gebaseerd op het CAM-reference model [3]. Dit model is ontwikkeld door Philips en Digital. De verschillende bestuurlijke nivo's staan in figuur 2.1. Op elk nivo wordt de besturing uitgevoerd door een zogenaamd produktie besturingssysteem (PCS).



Figuur 2.1: De bestuurlijke nivo's in het CAM-reference model

Het produktiebesturingssysteem maakt gebruik van recepten, waar in beschreven staat hoe het betreffende nivo moet handelen. Een produktie voorbereidingssysteem (PPS) levert deze recepten aan. Een ander onderdeel van de besturing, het produktie evaluatiesysteem (PES), verzamelt produktiegegevens. Het is de bedoeling om de onderdelen van de besturing modulair op te bouwen.

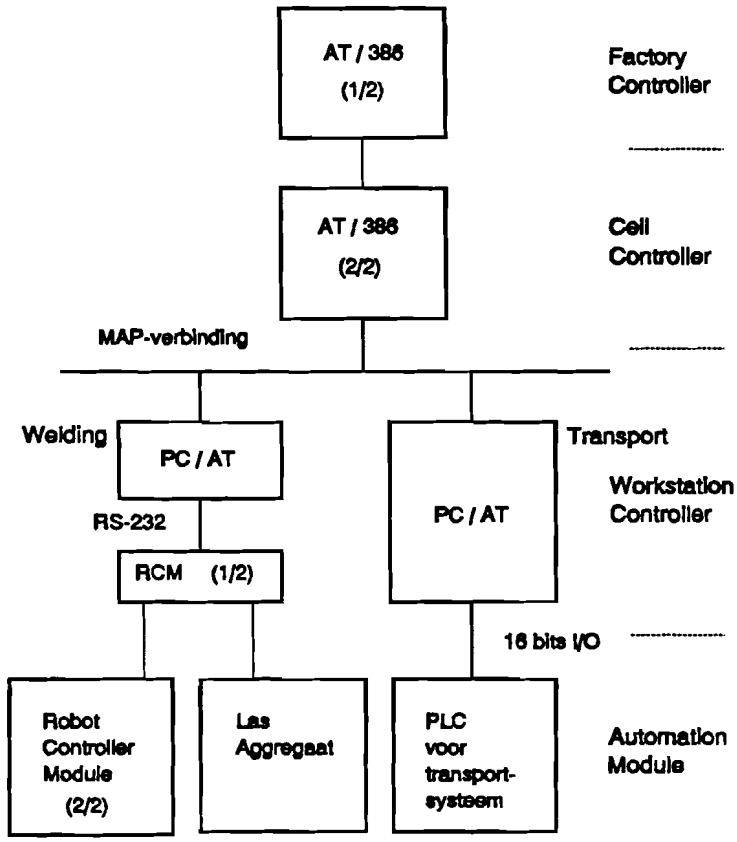
2.2. De onderdelen van de FALC

De bestuurlijke nivo's uit figuur 2.1 zijn ook in de FALC terug te vinden. In de FALC zijn drie groepen van devices te herkennen, namelijk de devices om te kunnen lassen, de devices in de lasrobot en de devices in het transportsysteem. De groepen devices worden elk bestuurd door een automation module. In het besturingsmodel coördineren workstations een bewerking. De bewerkingen in de cel zijn het lassen van een produkt en het vervoeren van een produkt. Het workstation welding, die het lassen coördineert, bestuurt de automation modules, die de bewegingen van de robot en lasapparatuur aansturen. Het workstation transport bestuurt de automation module, die de devices in het transportsysteem aanstuurt. Een cell controller coördineert de hele cel, in dit geval dus de beide workstations. De cell controller stuurt de workstations zo aan, dat de bewerkingen aan het produkt in de juiste volgorde worden uitgevoerd. De controller voert daarnaast ook een planning uit, als meerdere produkten tegelijk in bewerking zijn. De factory controller bepaalt wanneer een produkt geproduceerd wordt. De controller deelt orders uit aan de cell controller.

De functionaliteit van de bestuurlijke nivo's is geïmplementeerd in verschillende apparaten. De precieze verdeling staat in figuur 2.2.

Het workstation welding is zowel in een robot controller module (RCM), als in een PC/AT geïmplementeerd. Vanuit de robot controller worden zowel het lasaggregaat als de motorsturingen van de lasrobot aangestuurd. Die aansturing gebeurt via robotprogramma's. Het andere deel van dit workstation is geïmplementeerd in een PC/AT, die aan de hand van een opdracht van de cell controller robotprogramma's kiest. In het vervolg wordt deze PC/AT PC welding genoemd.

De funktie van de factory controller, zoals die in [4] beschreven staat, wordt alleen uitgevoerd door de operator. De operator voert daar geheel zelfstandig zijn planning uit. In de huidige versie van de cel wordt de operator geholpen door een orderverwerking, waar aan hij een sequentie van orders opgeeft. Een order is in dit geval een recept voor de cell controller. De orderverwerking verzorgt de vrijgave van elk van de orders. Zowel de orderverwerking, als de cell controller zijn geïmplementeerd in de AT-386. In de AT-386 zijn de twee bestuurlijke nivo's in gescheiden software modules ondergebracht.



Figuur 2.2: Verdeling van de besturingsnivo's over de apparaten.

2.3. De netwerkverbindingen

De drie PC's zijn met elkaar verbonden via een MAP-netwerk [4] (zie figuur 2.2). Het MAP-netwerk (Manufacturing Automation Protocol) is een token-bus netwerk, dat ontwikkeld is door General Motors. Via dit netwerk zouden elk van de aangesloten apparaten, dus ook de twee PC/AT's, met elkaar kunnen communiceren, maar dit gebeurt in de praktijk niet, omdat het hiërarchische besturingsmodel die direkte communicatie niet toelaat.

MAP is gebaseerd op het OSI Reference Model [5]. Door de gebruikte hardware en software [6], heeft de gebruiker van het netwerk in de praktijk te maken met de twee uiterste lagen van dit model, de fysische laag en de applicatielaag. De fysische laag is de kabel, waarmee de computers met elkaar worden verbonden. De applicatielaag biedt netwerkservices aan het gebruikersprogramma. Als applicatielaag wordt MMS gebruikt [1]. Manufacturing Message Specification is gedefinieerd in de ISO-standaard 9506.

De PC welding is door een RS232-verbinding verbonden met de robot controller. De applicatielaag van deze verbinding biedt DNC-services [7]. Deze zijn heel anders dan

de MMS-services. De PC welding vormt dus, naast een gedeelte van het workstation, ook een interface tussen het MAP-netwerk en de RS232-verbinding.

2.4. Manufacturing Message Specification

Een van de standaarden, die als applicatielaag van een MAP-netwerk gebruikt kan worden is MMS. Via MMS-services kunnen operaties worden opgedragen aan een Virtual Manufacturing Device. Dit VMD is een abstract model van een apparaat, de implementatie van MMS zorgt voor de werkelijke aansturing.

MMS kent verschillende objecten en services ten behoeve van deze objecten. Enkele objecten zijn domains, program invocations en operator communications.

2.4.1. Domains

Een domain is een object, dat een functionaliteit van het VMD beschrijft. Een domain kent verschillende attributen, waaronder een status en een inhoud. Als inhoud kan bijvoorbeeld het operating system, een programmatekst of gegevens voor een programma dienen. De status hangt sterk samen met de services voor een domain. Services bestaan voor up- en downloading, vernietiging en het opvragen van de attributen. De status heeft dan ook de volgende mogelijkheden: LOADING, COMPLETE, INCOMPLETE, READY en IN-USE. Een precieze omschrijving is te vinden in [4].

2.4.2. Program invocations

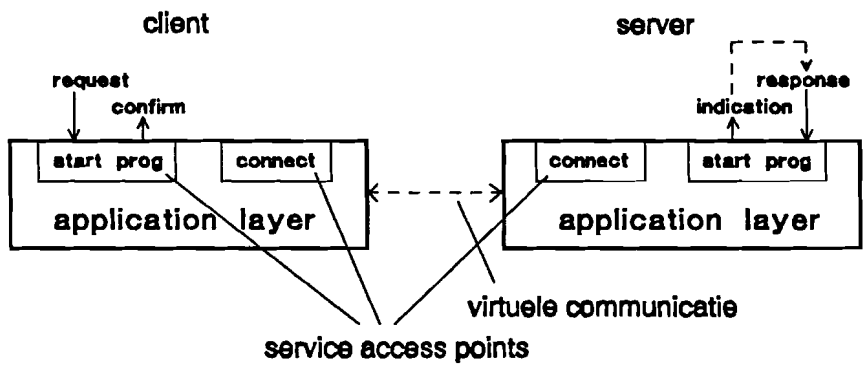
Een program invocation is een object, dat een programma-uitvoering beschrijft. De attributen omschrijven referenties naar domains en de toestand van een program invocation. De program invocation state kent de volgende mogelijkheden: IDLE, RUNNING, STOPPED, STARTING, STOPPING, RESUMING, RESETTING en UNRUNNABLE [4]. Een program invocation beschrijft alleen de programma-uitvoering. De programmatekst en verdere benodigdheden om een programma uit te voeren zijn in domains ondergebracht.

2.4.3. Operator communications

Met behulp van de operator communications kunnen gegevens met de operator worden uitgewisseld. Via de output-service kunnen berichten naar een operator station worden geschreven. Door middel van de input-service kunnen van de operator gegevens worden gevraagd. Deze services bieden slechts simpele mogelijkheden voor communicatie.

2.5. Netwerkaansturing

De services ten behoeve van de verschillende objecten bestaan uit bevestigde en onbevestigde services. De services zijn bereikbaar via zogenaamde service access points. De services kunnen worden bestuurd door de dienstprimitieven request, indication, response en confirm. Een voorbeeld van de opeenvolging van gebeurtenissen bij een bevestigde service staat in figuur 2.3.



Figuur 2.3: Services van de applicatielaag

Het voorbeeld uit de figuur beschrijft de start-service voor een program invocation. De dienstprimitieven worden in MMS-EASE [6] door de taal C aangestuurd via een zogenaamd paired-primitive interface. In C is de opeenvolging als volgt:

- De client roept de library funktie voor de start-service aan. Deze funktie verzorgt een request, die een indication aan de server-kant tot gevolg heeft.
- Bij de server wordt na ontvangst van de indication naar een indication funktie gesprongen, waar de server zijn code voor de verwerking kan plaatsen.
- Na verwerking door de server van de indication roept de server een library funktie voor de response aan. De server kan kiezen uit een normale response en een fout-response. Aan de laatste response worden door de ISO 9506 gedefinieerde foutcodes meegegeven. Deze response heeft aan de client-kant een confirm tot gevolg.
- Bij de client wordt na ontvangst van de confirm naar een confirm funktie gesprongen, waar de client zijn code voor de verwerking van de foutcodes kan plaatsen.

Een onbevestigde service gebruikt alleen de primitieven request en indication.

3. Het user-interface

In de FALC wordt de planning van orders voor de cel, de taak van factory controller, verzorgd door de operator. Om de planning goed uit te kunnen voeren, heeft de operator een overzicht nodig van de orders, die hij wil gaan uitvoeren. De factory controller legt zijn orders op aan de cell controller. Een cell controller geeft informatie terug over het verloop van de orders. Die informatie zal dus aan de operator moeten worden gepresenteerd. Een order bestaat uit een recept, zoals dat in 2.2 is genoemd. Zoals in 4.6 zal blijken, zullen deze recepten op een gebruikersvriendelijke manier aangemaakt moeten kunnen worden. Een user-interface kan hierin voorzien.

3.1. Algemene opzet

Voor een algemene basis van het user-interface is in de literatuur gezocht naar standaard interfaces voor geautomatiseerde produktie. Als resultaat zijn enkele beschrijvingen gevonden, maar op het gebied van standaardisatie is er niets uitgekomen. Een van die beschrijvingen [8] geeft aan hoe een interface voor een plasticverwerkende machine er uit kan zien. Naast deze beschrijving is ook de terugkoppeling van de toekomstige gebruikers als informatiebron gebruikt.

Het interface verdeelt de informatie over verschillende schermen en lagen. De lagen geven naar onder toe een steeds verdere detaillering van het functioneren van de cel. Binnen die lagen kunnen meerdere schermen te zien zijn op een zelfde nivo van detaillering. De onderste lagen laten het bovenste gedeelte van het scherm vrij, zodat de bovenste lagen ook nog te zien zijn. Een laag kan volledig zichtbaar gemaakt worden, door naast het zichtbare deel van die laag op een knop te drukken. Dit drukken kan met een light-pen, een muis, of, als er een touch-screen is geïnstalleerd, met de vingers. De AT-386, die voor het interface gebruikt wordt, heeft geen touch-screen, zodat bediening door aanwijzing afvalt. De AT heeft een VGA-kaart. Deze videokaart heeft geen aansluiting voor een light-pen. Een muis zou aangesloten kunnen worden, maar de kans op vervuiling van de snelheidsopnemer is erg groot, vandaar dat naar een andere oplossing is gezocht.

De operator gaat met het user-interface ook recepten schrijven, zodat het toetsenbord in ieder geval gebruikt zal worden. De functietoetsen staan het dichtst bij het scherm. Vandaar dat, voor het zichtbaar maken van de lagen, functietoetsen gekozen zijn. Binnen een laag kan met de cursortoetsen links en rechts van scherm gewisseld worden. De knoppen naast elk zichtbare deel van een laag zijn vervangen door beschrijvingen van funktietoetsen onder aan het scherm, zo dicht mogelijk bij de eigenlijke toetsen.

Omdat de funktietoetsen onder aan het scherm naast elkaar staan, zijn de lagen ook horizontaal gestapeld (zie bijlage A). Als lagen nu gestapeld zijn, worden de titels van de laag

er onder afgedekt. Vandaar, dat verkorte versies van die titels ook bij de beschrijving van de funktietoetsen vermeld staan.

3.2. De verschillende schermbeelden

Uit de funkties, die het user-interface moet kunnen verrichten, zijn de volgende drie lagen afgeleid:

overzicht:

- het overzicht van de FALC

berichten:

- berichten van de cell controller
- berichten van het workstation Welding
- berichten van het workstation Transport

opdrachten:

- order (zet een order in de orderlijst)
- verwijder (verwijder een order uit de orderlijst)
- start (ga de orderlijst uitvoeren)
- afsluiten (sluit de bewerkingen van de cel af)
- recept (bewerk een recept)

Deze verschillende lagen zijn uitgetekend in bijlage A en staan in de volgende paragrafen beschreven.

3.2.1. Het overzicht van de FALC

De overzichtlaag informeert de operator over de algemene status van de cel. In de laag wordt op het scherm schematisch het transportsysteem en de lasrobot getoond. Hiermee kan worden aangegeven, dat één of beide workstations in rust, bezig of in alarmtoestand is. Die verschillende toestanden worden getoond met kleuren. De toestand van de cell controller kan worden afgeleid uit de bovenste regel van het interface, waar staat of een recept wordt uitgevoerd. Uit de kleur van een alarmlamp rechts onder in het scherm kan worden afgeleid of de hele cel zich in alarmtoestand bevindt.

3.2.2. Berichten

Met behulp van de berichtenlaag kan de operator de cel op een gedetailleerde manier in de gaten houden. In de bovenste regel wordt met behulp van de kleuren aktief en normale-tekst aangegeven, welk scherm in deze laag de operator voor zich heeft. De funktietoets, die deze laag aktiveert, wisselt van kleur tussen aktief en in rust, afhankelijk van het feit of er berichten moeten worden gelezen.

3.2.3. Opdrachten

De operator kan met behulp van de opdrachtenlaag de cel besturen. In deze laag kan de operator de planning uitvoeren, door een orderlijst samen te stellen. Een order bestaat uit de naam van een recept, dat uitgevoerd moet gaan worden. De orderlijst is kontinu in beeld.

Ook het aanmaken van recepten is in deze laag verwerkt. Met de opdracht recept kan de operator de recepten voor de verschillende stations aanmaken. Het systeem zorgt er voor, dat de recepten op de juiste plaats terecht komen (zie 4.6).

3.3. Kleuren

De kleuren zijn gekozen volgens de aanwijzingen uit [9],[10] en de terugkoppeling van de toekomstige gebruikers. De criteria voor de gekozen kleuren zijn:

- Bescheiden kleurgebruik.
- Consistent kleurgebruik.
- Goed onderscheid tussen de kleuren voor rust, bezig en alarmtoestand.
- Goede leesbaarheid van de tekst.

In eerste instantie zijn de volgende kleuren gekozen:

- Normale tekst: zwart op wit
- Aktieve keuze: wit op zwart
- Onderwerp in rust: lichtblauw op wit
- Onderwerp bezig: donkerblauw op wit
- Alarmtoestand: helder wit op rood

Bij gebruik in een afgeschermde ruimte blijken deze kleurencombinaties te voldoen. Bij inval van helder zonlicht op het scherm is het contrast van de combinatie lichtblauw op

wit te laag. Het onderwerp is in dit geval in rust, zodat geen gevaar uit kan gaan van dit lage contrast.

3.4. Hardware

In [11] wordt een user-interface beschreven, dat een text-mode gebruikt. In de resolutie die gebruikt wordt, 80 kolommen bij 43 regels, biedt de EGA-adapter verschillende video-pagina's. Daarbij heeft de text-mode een snelheidsvoordeel boven van de grafische mode. De VGA-adapter, die in de AT-386 geïnstalleerd is, heeft veel meer tekstresoluties ter beschikking dan de videokaart, gebruikt in [11]. De adapter heeft een resolutie tot maximaal 132 kolommen bij 60 regels, maar dit blijkt onleesbaar op de aangesloten monitor. Het leesbaarheidscriterium leidt in eerste instantie tot de keuze van 80 kolommen bij 43 regels.

De gekozen vertikale resolutie hangt af van het totaal aantal beeldlijnen en het aantal beeldlijnen binnen een karakter. Bij de resolutie van 43 regels zijn deze getallen respectievelijk 350 en 8. De karakterset, door de adapter gedefiniëerd voor karakters van 8 beeldlijnen, heeft een kleine regelafstand. Op de aangesloten monitor blijken dan twee opeenvolgende regels moeilijk te onderscheiden.

Het aantal beeldlijnen, dat een VGA-adapter genereert, kan gekozen worden uit 200, 350, 400 en 480 lijnen. De beeldfrequentie is 70Hz, behalve bij de resolutie van 480 lijnen, dan is de beeldfrequentie 60Hz. De laatste vertikale resolutie valt voor het user-interface af, omdat dan het beeld op de aangesloten monitor knippert. Om genoeg ruimte te maken voor de opbouw van het interface is gekozen voor 400 lijnen.

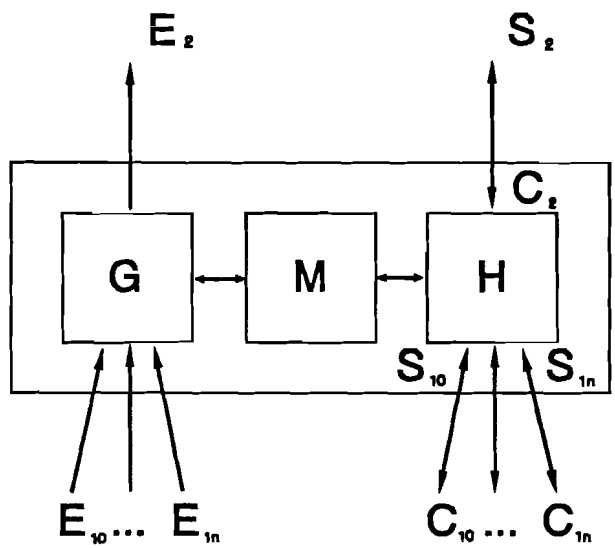
De karakterset, die een VGA-adapter gebruikt, kan een willekeurige hoogte aannemen tot een maximum van 32 beeldlijnen. De karakterset met een hoogte van 8 beeldlijnen kan dus worden uitgebreid met lege beeldlijnen om extra ruimte tussen de regels te krijgen. Om in de buurt te komen van de vertikale resolutie van 43 regels is gekozen voor 10 beeldlijnen per karakter. Het scherm bestaat nu uit 80 kolommen bij 40 regels.

4. Recepten

Elk nivo in het besturingsmodel wordt aangestuurd met behulp van recepten. Met deze recepten wordt een decompositie van taken van de cel gerealiseerd. Voor het ontwerp van de verwerking van recepten is inzicht nodig in het model van een controller in het besturingsmodel.

4.1. Controllers in het CAM-reference model

Elke controller in het CAM-reference model kan meerdere controllers van een lager nivo aansturen. Het model van een controller staat in figuur 4.1. Dit model is gebaseerd op een model gedefinieerd in [12].



Figuur 4.1: Een laag in het CAM-reference model.

De beschrijving van de modules is als volgt:

- De H-module in dit model splitst een opgelegde opdracht, in de figuur C_2, in deelopdrachten voor het onderliggende nivo (C_{10} ... C_{1n}). De module vangt daarnaast informatie met betrekking tot de uitvoering van de opdrachten op in de vorm van de signalen S_{10} tot en met S_{1n}. In de FALC wordt het splitsen in deelopdrachten verzorgd door de verwerking van de recepten. Die deelopdrachten zijn vaak uitgevoerd als netwerk-requests. De verwerking van informatie over de uitvoering van taken gebeurt in de FALC door de opvang van een confirm van de bijbehorende netwerk-request.
- De G-module vangt statusinformatie op (E_{10} ... E_{1n}) en geeft informatie door aan de bovenliggende laag (E_2). Die statusinformatie bestaat uit informatie, die niet direkt gerelateerd is aan de uitvoering van de deelopdrachten uit de H-module. In de

FALC wordt deze module gevormd door de opvang van de ongevraagde statusinformatie van de onderliggende laag.

- De M-module, het zogenaamde wereldmodel, vertaalt, met behulp van de kennis over de produktieomgeving, informatie tussen de H- en de G-modules. Deze vertaling is nodig, om een verwachting uit te spreken over de statusinformatie en om met behulp van de statusinformatie beslissingen te kunnen nemen over de taakdecompositie.

Het wereldmodel is te illustreren aan de hand van de start-service van MMS. Als een start-request wordt uitgevoerd, dan kunnen, of een bevestiging, of bepaalde foutcodes worden terugverwacht. Welke foutcodes kunnen worden terugverwacht, wordt met het wereldmodel bepaald aan de hand van het feit dat een start-request is uitgevoerd. Als foutcodes binnenkomen, kunnen ze met het wereldmodel vertaald worden naar begrijpelijke informatie voor de H-module. Die informatie zou kunnen bestaan uit gestart, alarm, bezig en dergelijke.

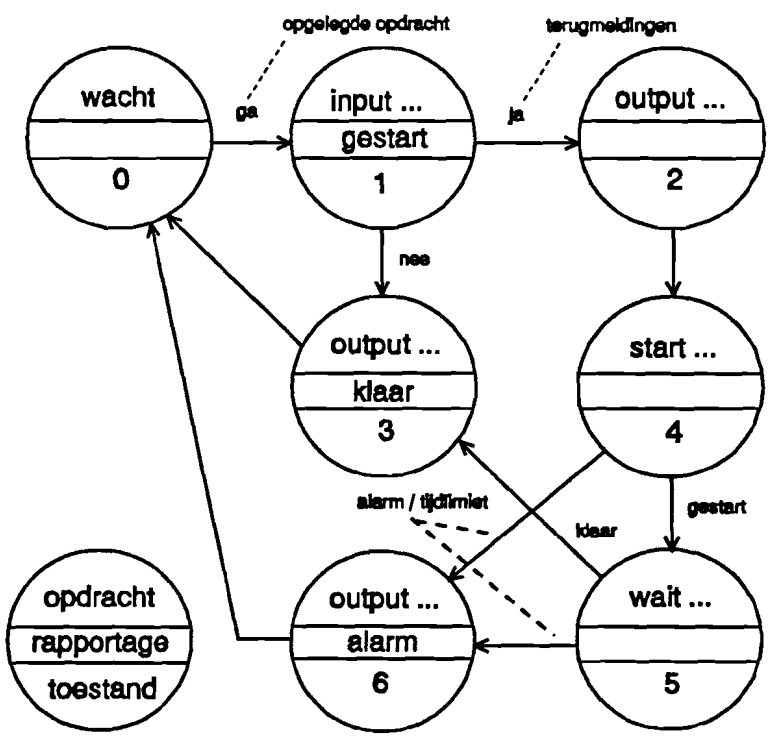
4.2. Toestandsmachines

De H-module in een controller krijgt via het wereldmodel en de signalen S terugkoppeling. Met die terugkoppeling kan de H-module beslissingen nemen. Het nemen van beslissingen kan worden geïmplementeerd met behulp van een standaardprogrammeertaal. Deze implementatie moet dan gebeuren door programmeurs. In [12] wordt voorgesteld om het nemen van beslissingen door middel van toestandsmachines te implementeren. Met deze toestandsmachines worden opdrachten uitvoerd en rapportages gegeven aan de hand van terugmeldingen van het onderliggende nivo. Deze toestandsmachines zijn grafisch voor te stellen en waarschijnlijk ook door niet-programmeurs op te stellen.

De grafische voorstelling van de toestandsmachines, zoals die in [12] is voorgesteld, lijkt niet consequent. In de bollen, die de toestanden voorstellen, staan de opdrachten, die naar de onderliggende laag worden gegeven. Bij de toestandsovergangen staan echter zowel terugmeldingen van het onderliggende nivo als de rapportage naar het bovenliggende nivo. Bij de toestandsovergangen staan dus zowel die dingen, die de overgang veroorzaken, als de door het besturingsnivo zelf geïnitieerde rapportage.

Om de zelf veroorzaakte rapportage van externe invloeden te scheiden, zijn de rapportages naar de bollen verhuisd. De vorm van toestandsmachines, zoals die verder gebruikt gaat worden, is geïllustreerd met figuur 4.2. In de figuur wordt een recept van de cell controller beschreven. In het recept wordt het volgende uitgevoerd:

- Aan de operator wordt een vraag gesteld met behulp van de input-opdracht. Als de operator met ja antwoordt, dan wordt het recept verder uitgevoerd, is het antwoord nee, dan wordt een melding op het scherm gezet, klaar gerapporteerd en gestopt met de uitvoering.
- Op het scherm wordt een melding gezet met de output-opdracht, bijvoorbeeld dat de uitvoering is begonnen.
- Tijdens de verdere uitvoering wordt een recept op een workstation gestart. Komt als terugmelding dat het recept op het workstation is gestart, dan wordt het recept verder uitgevoerd. Komt er geen antwoord binnen een gestelde tijd of komt er een alarmmelding, dan wordt een melding op het scherm gezet, alarm gerapporteerd en de uitvoering van het recept gestopt.
- Is het recept gestart, dan wordt gewacht totdat een melding van het workstation komt. Komt de melding klaar, dat wordt de uitvoering van het recept gestopt. Komt een alarmmelding of komt geen melding binnen een gestelde tijd, dan worden in dit geval dezelfde maatregelen genomen als na de start-opdracht.



Figuur 4.2: Een recept als toestandsmachine.

De toekomstige operators kunnen met behulp van het user-interface zelf recepten schrijven. Het denken in toestanden en in toestandsovergangen blijkt voor hen bij eerste

confrontatie geen probleem te zijn. Bovendien blijken de recepten, die in [4] zijn aangemaakt, direkt naar toestandsmachines om te zetten.

4.3. Toestandstabellen

De toestandsmachines uit 4.2 zijn niet direkt als recept aan een controller aan te bieden. Daarvoor worden ze omgezet naar een, voor een computer leesbare, tabel. De volgorde van de kolommen in de tabel is in overeenstemming met de vorm van het toestandsdiagram uit figuur 4.2. De opzet van de tabel staat in figuur 4.3.

toestands nummer	opdracht	rapportage	criterium	volgende toestand

Figuur 4.3: Vorm van de toestandstabel.

De interpretatie van de tabel is als volgt:

- In een bepaalde toestand wordt een opdracht uitgevoerd en een rapportage gegeven.
- Na uitvoering van de opdracht wordt aan de hand van een criterium naar een volgende toestand gesprongen.

Nadat tabellen zijn opgesteld, moeten deze uitgevoerd kunnen worden. De opzet van de tabellen is star. Een interpreter voor deze tabellen is daarom betrekkelijk eenvoudig.

De interpreter uit het produktie besturingssysteem, beschreven in [4], is vervangen door een interpreter voor bovenbeschreven tabellen. De opzet van de interpreter is flexibel; te geven opdrachten en bijbehorende criteria zijn eenvoudig toe te voegen (zie 6.3).

4.4. Receptentaal

De verzameling opdrachten, zoals die in [4] gedefinieerd is, heeft enkele namen van MMS-services als basis. Daarnaast zijn er opdrachten om de PLC te besturen en een opdracht, waarbij gewacht wordt op een gebeurtenis. Zoals in 5.2 wordt uitgelegd, zijn de twee opdrachten **initiate** en **conclude** niet meer nodig.

4.4.1. Opdrachten

De cell controller heeft produkten nodig. In de FALC worden de produkten aangevoerd door het invoerstation. Dit funkties van dit station worden nu waargenomen door de operator. Het is dus noodzakelijk, dat er een opdracht is, die de operator kan vragen, om

een bepaald onderdeel van buiten de cel aan te voeren. De opdracht **input** is gemaakt, om via het user-interface vragen te stellen. De cell controller bestuurt het invoerstation, zodat de opdracht **input** alleen in recepten voor de cell controller kan worden gegeven.

De operator voert ook taken van de factory controller uit. De cell controller heeft een opdracht nodig, om de factory controller te informeren over zijn toestand. Met de **output**-opdracht kan de cell controller meldingen via het user-interface aan de operator doorgeven.

Een opsomming van de opdrachten met hun omschrijving is te vinden in tabel 4.1. Op de uitvoering van de **wait**-opdracht wordt in 5.4 teruggekomen.

Tabel 4.1: Omschrijving van de opdrachten uit de receptentaal

opdracht	controller	omschrijving
start	Cell Welding	start een recept op onderliggend nivo
wait	Cell Welding Transport	wacht op terugmelding van onderliggend nivo
activate	Transport	druk een baanvak- of plaatsknop op de PLC in
reset	Transport	druk een baanvak- of plaatsknop op de PLC uit
output	Cell	geef tekst door aan operator
input	Cell	stel vraag aan operator en wacht op antwoord

4.4.2. Criteria

De criteria, met behulp waarvan naar een volgende toestand wordt gesprongen, worden afgeleid van wat in de werkelijkheid kan gebeuren. Als een opdracht in een recept voorkomt, dan zal in principe voor elk criterium, dat bij die opdracht hoort, een regel tekst in het recept moeten komen. In de praktijk moet dus een afweging worden gemaakt tussen het aanbod van mogelijke criteria aan de receptenprogrammeur en de daar mee samenhangende moeilijkheid van programmeren.

In eerste instantie zijn weinig criteria gekozen bij de opdrachten om ervaring op te doen in het nemen van beslissingen op deze manier. In tabel 4.2 zijn ze opgesomd.

Tabel 4.2: Oorspronkelijk gekozen criteria

opdracht	bijbehorende criteria
start	gestart alarm tijdlimiet
wait	klaar alarm tijdlimiet
activate	(geen criterium)
reset	(geen criterium)
output	(geen criterium)
input	answer=..

In de praktijk blijken sommige criteria altijd naar bepaalde toestanden te leiden. Het criterium **alarm**, waarmee een werkelijk ernstig alarm is bedoeld, leidt altijd tot stoppen van het recept. Ook het criterium **tijdlimiet**, waarmee bedoeld wordt dat een bevestiging van de opdracht uitblijft, leidt tot het zelfde. In dit geval reageert een workstation of automation module niet meer, waardoor een order niet uit te voeren is.

Het criterium **tijdlimiet** lijkt wel nuttig als meerdere workstations bestaan, die de zelfde bewerkingen uit kunnen voeren. Als wordt geconstateerd, dat een workstation niet meer reageert, dan kan naar een andere worden uitgeweken.

Doordat het criterium **alarm** altijd naar een zelfde toestand leidt, wordt in de besturing nu het criterium **alarm** automatisch verwerkt, zonder tussenkomst van het recept. De rest van de criteria wordt wel via het recept verwerkt. Een gedetailleerde omschrijving van de receptverwerking is te vinden in bijlage H.

Welke criteria wel nodig kunnen zijn, kan worden afgeleid uit wat met een recept eigenlijk wordt uitgevoerd. Het produktie besturingssysteem werkt met de recepten om zijn taak in het besturingsmodel uit te voeren.

De cell controller coördineert de workstations en plant wanneer deze aktief zullen zijn. Volgens [13] heeft een cell controller te maken met de volgende teruggemelde fouten:

- Te laat beëindige operaties. Hiermee kan eventueel de planning worden aangepast.
- Niet aanwezige of slechte onderdelen van het produkt. Een nieuw produkt zal moeten worden aangevoerd.
- Kapot gereedschap. Als de aanvoer van gereedschap ook door de cell controller wordt gecoördineerd, dan zal dit moeten gebeuren.
- Workstation fouten. Het workstation kan niet meer in de planning worden opgenomen en het zal moeten worden gerepareerd.

In de FALC wordt een tijdlimiet opgelegd om te kunnen constateren, dat een workstation niet meer reageert. Vandaar, dat aan de hand van de melding, dat een operatie te laat beëindigd is, de planning niet wordt aangepast.

De workstations coördineren een bewerking. De devices, die via de automation modules worden aangestuurd, kennen een bepaald werkgebied. Om verschillende bewerkingen op een produkt uit te kunnen voeren, zullen de produkten overgedragen moeten worden. De werkgebieden van devices, aangestuurd door verschillende workstations, zullen dus overlappen. Deze overlap wordt in het vervolg het overdrachtsgebied genoemd.

In [13] wordt met een besturingsmodel gewerkt, waar de overdracht van produkten door de workstations zelf gebeurt. In de FALC wordt de uitvoering van recepten op een work-

station pas begonnen als de cell controller van het andere workstation heeft begrepen, dat de overdracht kan plaats vinden. De cell controller moet dus ook weten, of het ene workstation gevaar oplevert voor de andere. Onder normale omstandigheden is dit te garanderen, door de programmatuur zo te schrijven, dat de devices van de workstations niet een gebied komen, dat buiten het overdrachtsgebied en in elkaars werkgebied ligt. Of dit ook geldt onder buitengewone toestanden, zal uit terugmeldingen moeten blijken.

In de FALC is de manipulatortafel het overdrachtsgebied tussen de workstations. De robotprogramma's kunnen zo worden opgesteld, dat de robot niet buiten de manipulatortafel opereert. De robot kent hardware-eindschakelaars, die het werkgebied van de robot aangeven. Dit werkgebied reikt tot over de voorlangslopende transportbaan. Als bij het workstation welding de melding binnenkomt, dat een hardware-eindschakelaar is geraakt, dan kan het workstation concluderen, dat zijn devices gevaar opleveren voor de transportbaan. Het transportsysteem kan alleen binnen het werkgebied van de robot komen, als produkten naar of van de manipulatortafel komen. De cell controller geeft hiertoe alleen op de juiste tijden opdracht, zodat geen gevaarlijke situaties kunnen ontstaan.

In het FALC-project wordt ook gedacht aan veiligheid, met name aan het feit, dat iemand het werkgebied van de robot kan binnendringen. In dit geval zou de robot stilgezet en het lassen afgebroken moeten worden. Dit betekent echter een pauze in de uitvoering van een recept. Op de cell controller zou de opgelegde tijdlimiet kunnen verstrijken, terwijl, nadat de persoon is vertrokken, de produktie misschien normaal kan doorgaan. Meer over veiligheid is te vinden in [14].

Uit het bovenstaande blijkt, dat belangrijke informatie van de workstations voor de cell controller kan zijn:

- Overschreden tijdlimiet.
- Slecht of niet aanwezig produkt.
- Workstation uitgevallen, geen gevaar buiten eigen werkgebied.
- Workstation uitgevallen, wel gevaar buiten eigen werkgebied.
- Workstation in pauze-toestand.

Een workstation genereert deze informatie door middel van rapportage. In het geval van workstation welding kunnen de laatste drie punten worden afgeleid uit de alarmcodes, die de robotcontroller stuurt. Het eerste punt wordt door middel van tijdbewaking door de cell controller verzorgd. Of een produkt wel of niet slecht is, kan onder andere worden afgeleid of een las is afgebroken. Is dit het geval, dan is de verdere verwerking van dit produkt niet zinvol. De PC welding kan niet detecteren, of gelast wordt, omdat een deel van het workstation welding ook in de robotcontroller is geïmplementeerd. Een oplossing

zou kunnen zijn, om in de robotcontroller de aansturing voor het lasaggregaat en de motorsturing te scheiden, zodat de PC welding deze aansturingen appart kan besturen.

De receptverwerking zal nog moeten worden gewijzigd, om de bovengenoemde informatie te kunnen aanmaken en verwerken.

4.5. Verband tussen recepten en MMS-objecten

Het produktie besturingssysteem wordt aangestuurd met recepten, met andere woorden, recepten zijn de programma's voor dat systeem. Via MMS worden de recepten voor de workstations aangestuurd, zodat deze recepten op MMS-objecten geprojecteerd moeten worden.

MMS kent het object program invocation, maar dit dekt alleen de beschrijving van de programma-uitvoering. De status van de uitvoering van een recept wordt met dit object gemodelleerd. Een program-invocation maakt van domains gebruik om het programma werkelijk uit te laten voeren. In die domains kunnen bijvoorbeeld programmacode of gegevens staan. Het gebruik van die domains wordt aan het programma overgelaten, die de MMS-services verwerkt en aanstuurt, het gebruikersprogramma.

Een program-invocation heeft in ieder geval programmacode nodig, in het geval van de FALC is dit de tekst van een recept. Elk bestand met een recept voor een workstation wordt dus aangemeld als domain in een PC/AT. Elk recept bestaat nu uit slechts een bestand, dus tussen een program-invocation en een domain is een één op één relatie. Die relatie moet wel door het gebruikersprogramma worden gelegd. In 4.6 wordt op deze relatie teruggekomen.

4.6. Aanmaken en downloaden van recepten

Een operator schrijft recepten voor de cell controller en de workstations. De operator bedient de cel vanaf de AT-386 via het user-interface en wordt op die manier afgeschermd van de direkte bediening van de PC/AT's. Doortrekken van die afscherming naar het schrijven van de recepten is dan een volgende stap. Via programmatuur op de AT-386 worden dus recepten geschreven. Deze programmatuur moet onderscheid maken tussen de recepten van de verschillende controllers. De recepten worden onderscheiden, door ze in verschillende directories op de harddisk van de AT-386 te plaatsen. Via een normale editor kan dit onderscheid niet worden gemaakt, omdat dit van de operator eist, dat hij kennis heeft van de directory structuur. Het user-interface kan, met kennis van de directory structuur, de editor aansturen. De bediening wordt dan voor de operator vriendelijker.

Het besturingsprogramma van de AT-386 waarin het user-interface als module is verwerkt, neemt veel geheugen in beslag. De aangeroepen editor heeft dus weinig geheugen ter beschikking. Het is mogelijk, om met utility's het besturingsprogramma uit het geheugen te verwijderen als de editor wordt aangeroepen, maar dit is niet zonder risico. De library-funkties van MMS-EASE kunnen namelijk met interrupts werken [6]. Het besturingsprogramma werkt niet met interrupts, maar de handleiding geeft geen uitsluitsel of de interrupts hardwarematig kunnen worden uitgeschakeld. Vandaar, dat gekozen is voor een eenvoudige editor.

Wanneer de editor is aangeroepen, kan alleen maar met de editor worden gewerkt. Het besturingsprogramma kan dus geen informatie van het netwerk meer verwerken. Om te voorkomen, dat informatie binnenkomt, worden de logische verbindingen via het netwerk afgebroken voordat de editor wordt aangeroepen. Nadat de editor door de operator is verlaten, zullen de verbindingen weer moeten worden opgebouwd, om verder te kunnen werken met de cel.

Als de recepten zijn geschreven, staan ze op de harddisk van de AT-386. Recepten voor de workstations moeten dan nog worden overgebracht. Dit laatste is van de AT-386 af gezien downloaden. De tekst van een recept voor een workstation is ondergebracht in een domain in een PC/AT. Bij MMS zijn verschillende funkties ondergebracht, die gezamenlijk het apparaat vormen om de inhoud van een domain te uploaden en te downloaden:

- initiate download sequence
- download segment
- terminate download sequence
- initiate upload sequence
- upload segment
- terminate upload sequence

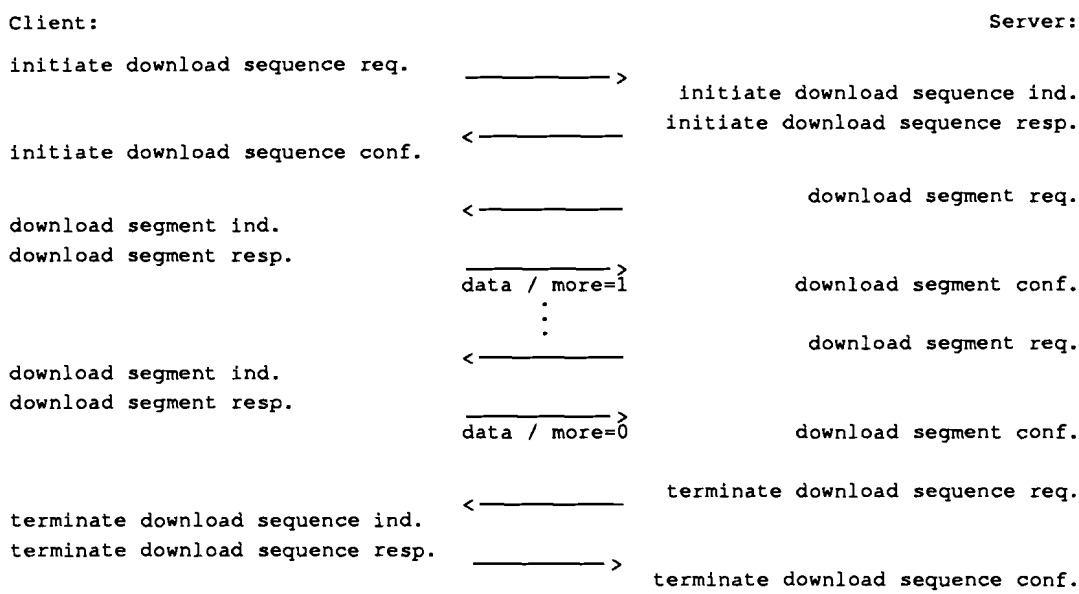
De achtereenvolgende gebeurtenissen bij het downloaden staan in figuur 4.4. Het linker gedeelte van de figuur kan in MMS-EASE worden vervangen door een funktie-aanroep, namelijk:

mv_download()

Dit is een virtual-machine-funktie, die niet deel uitmaakt van het paired-primitive-interface, zoals dat staat beschreven in 2.5. Deze funktie verdeelt zelf de aangeleverde data in meerdere segmenten en verzorgt de more-bits. Elke download segment indication en de terminate download sequence indication beantwoordt de funktie bovendien zelf. Na de opeenvolging van gebeurtenissen wordt naar een confirm-funktie gesprongen:

u_mv_download_conf()

In deze funktie kan worden gekeken of de opeenvolging goed verlopen is.

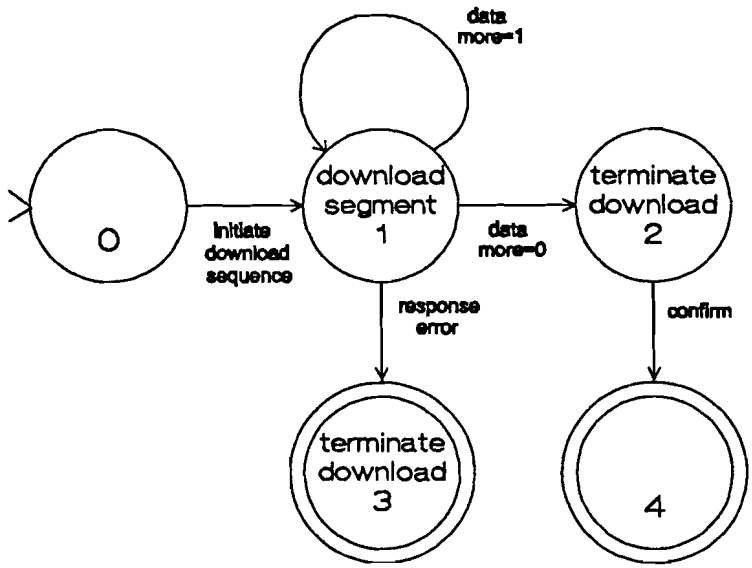


Figuur 4.4: Opeenvolging van gebeurtenissen bij downloaden.

Zoals in het rechter deel van de figuur 4.4 te zien is, moet de server ook requests uitvoeren. Met andere woorden, de server moet ook initiatief nemen. Dit kan, door speciaal voor het downloaden een toestandsmachine te maken, die op gang gebracht wordt door de client via de initiate download sequence request. Die toestandsmachine kan aan de hand van de domain-toestanden akties ondernemen. De domain-toestanden zijn:

- domain non existent
- domain loading
- domain ready
- domain in use
- domain complete
- domain incomplete

De werking van de toestandsmachine staat uitgebeeld in figuur 4.5. Details over de implementatie zijn te vinden in bijlage G.



Figuur 4.5: Toestandsmachine op de server bij het downloaden

Nadat de tekst van een recept als domain is aangemeld, moet een program invocation worden aangemaakt op een PC/AT, om het recept uit te kunnen voeren. Dit kan vanuit de AT-386 via de MMS-service CreateProgramInvocation. Een program invocation heeft een naam. Zoals in 4.5 staat, bestaat een relatie tussen een program invocation en een domain. In [4] wordt die relatie gelegd door een configuratiebestand, waarin beschreven staat, welk domain bij welke program invocation hoort. De receptenschrijver moet in dit geval zowel het bestand met het recept, als het configuratiebestand aanpassen om een recept in uitvoering te kunnen brengen. De identificatie van een domain kan ook in het domain zelf gebeuren, door een gedeelte van de inhoud van het domain daarvoor te reserveren. Een apparaat kan dan door zoeken in domains zelf de relaties leggen.

De identificatie van een domain gebeurt op dit moment door op een regel de naam van de bijbehorende program invocation te plaatsen. De opbouw van recepten is precies uitgelegd in bijlage E. In de toekomst kunnen voor de uitvoering van recepten meerdere domains nodig zijn. De identificatie zal dan uitgebreid moeten worden, zodat ook duidelijk is, waarvoor het domain gebruikt gaat worden. Het gebruikersprogramma moet weten welk domain in uitvoering moet worden genomen.

Aangezien downloaden van domains tijd kost, gebeurt het downloaden nu alleen, als een program invocation in uitvoering wordt gebracht vanuit de cell controller en als deze invocation niet aanwezig is. Als een program invocation niet aanwezig is, wordt in alle domains op de harddisk van de AT-386 gezocht naar de regel met de program invocation name. Het juiste domain wordt dan overgezonden.

Het zoeken in een domain blijkt veel meer tijd te kosten dan het overzenden er van. Als veel program invocations ontbreken, dan wordt veel tijd verspeeld met het zoeken in

bestanden. Een ander nadeel van deze methode is, dat de tekst van de recepten op meerdere plaatsen geldig moet zijn. Een betere oplossing kan zijn, dat het produktie voorbereidingssysteem bij het opstarten van de cel alle recepten als domain aan de PC/AT's aflevert en de bijbehorende program invocations aanmaakt. Het zoeken in de recepten hoeft dan maar een keer te gebeuren. Deze laatste oplossing is nog niet in het besturingsprogramma van de AT-386 verwerkt.

5. Communicatie

De communicatie tussen de apparatuur bestaat uit het opzetten en afbreken van logische verbindingen, verzenden van requests en responses, ontvangen van confirms en indications. Om transparantie voor het gebruikersprogramma te creëren, is het noodzakelijk, de communicatie zoveel mogelijk via MMS-berichten af te handelen. Dit, ook als de communicatie niet via het MAP-netwerk plaats vindt.

De operator heeft geen kennis van de netwerk-services. Toch is het nodig, dat hij de cel kan bedienen en recepten kan schrijven. Hij moet dus zonder specifieke kennis met de funkties van de verschillende besturingsnivo's kunnen werken.

5.1. Standaardisatie

De communicatie tussen de cell controller en de workstations vindt plaats via MMS-services. De aansturing van deze services is standaard. De aansturing van de services via de RS232 verbinding voldoet niet aan deze standaard, terwijl een aantal services wat funktie betreft wel overeenkomt. Een aantal DNC-services, die wat funktionaliteit overeenkomen met MMS-services, staat in tabel 5.1.

Tabel 5.1: Overeenkomstige netwerkservices

DNC-services	MMS-services
load + start	start
programma status	unsolicited status
alarmcode	unsolicited status

De funkties voor aansturing van de DNC-services zijn anders opgebouwd dan de MMS-request funkties, ze wachten zelf op bevestiging van de robot controller. Die funkties kunnen dan ook zelf de teruggekomen foutcodes verwerken. Om nu een vorm van standaardisatie te realiseren, vertalen die funkties de foutcodes naar de ISO-standaard 9506. De programmatuur, die deze funkties aanroept, kan dan op het punt van verwerking van foutcodes dus gelijk zijn. Het is natuurlijk mogelijk, om zoals in het MMS-EASE pakket, met funkties service acces points aan te maken. Dit heeft echter als consequentie, dat het interface tussen de DNC-services en het gebruikersprogramma geheel herschreven zou moeten worden.

Zowel via de MMS-services als via de DNC-services wordt ongevraagde statusinformatie gebruikt voor de doorgifte van foutcodes. Naast de standaard statusinformatie, gedefinieerd in de ISO-standaard, bestaat ook een Robot Companion Standaard [15]. Deze

standaard is toegespitst op de aansturing van robots. De additional-detail-parameter van de statusinformatie [4] is door deze standaard gedefinieerd volgens figuur 5.1.

```
RobotStatusDetail ::= Sequence {
     robotVMDState                [0] IMPLICIT RobotVMDState
     robotSpecificStatus          [1] IMPLICIT RobotSpecificStatus
     robotSpecificStatusMask      [2] IMPLICIT RobotSpecificStatusMask
                                    DEFAULT '11111'B
     }

RobotVMDState ::= INTEGER {
     robot-idle                        (0),
     robot-loaded                      (1),
     robot-ready                       (2),
     robot-executing                   (3),
     robot-motion-paused               (4),
     manualInterventionRequired        (5)
     }

RobotSpecificStatus ::= BITSTRING {
     safetyInterlocksViolated          (0),
     anyPhysicalResourcePowerOn        (1),
     allPhysicalResourcesCalibrated    (2),
     localControl                      (3),
     unitsOfMeasure                    (4)
     }
```

Figuur 5.1: Additional detail van de statusinformatie.

De robotSpecificStatusMask bepaald welke bits van de RobotSpecificStatus geldig zijn.

Via DNC-services komen foutcodes binnen van de robotcontroller, vandaar dat, om de berichtenstroom te standaardiseren, is gekozen voor de vertaling van die foutcodes naar de bovengenoemde standaard statusinformatie. Die vertaling gebeurt met behulp van een array, die een foutcode vertaalt naar de eerste twee bytes van de additional-detail-parameter (zie bijlage I). Het bitmasker blijft altijd gelijk en is bepaald door de informatie die de foutcodes kunnen verstrekken. De volgende bits hebben geen informatie:

- anyPhysicalResourcePowerOn (niet te detecteren via DNC)
- unitsOfMeasure (niet van belang voor de DNC-services)

Bij het opstarten van de cel worden voor de bits bepaalde waarden aangenomen. De waarden worden afgeleid uit het feit dat de robot controller in de mode Automatisch Extern kan worden gebracht. In deze mode kan de controller via de DNC-services worden bestuurd. De procedure, om de controller in de mode Automatisch Extern te brengen, brengt met zich mee, dat de robot in positie gevaren moet worden. Dit kan alleen als geen eindschakelaars worden geraakt. Uit de mode kan dus worden afgeleid, dat:

- safetyInterlocksViolated = 0
- allPhysicalResourcesCalibrated = 1
- localControl = 0

Het RobotVMDState deel van de additional-detail-parameter wordt gedeeltelijk bepaald door de status van de programma-uitvoering in de robot controller. De programma-status,

die door de DNC-services naar het workstation wordt gestuurd, wordt vertaald naar deze byte. De foutcodes zorgen eventueel voor de status manualInterventionRequired.

De workstations sturen statusinformatie in de vorm van een standaard MMS-bericht. Deze vorm kent geen additional-detail-parameter. De informatie over de toestand van programma-uitvoering van het workstation kan dus niet in deze parameter worden geplaatst. Voor deze informatie is een byte gereserveerd van de local-detail-parameter. Deze informatie is al gedefinieerd in [4]. De definitie heeft als nadeel, dat deze niet overeen komt met de definitie voor de status van de cell controller, terwijl de uit te voeren akties van een workstation vrijwel gelijk zijn [13]. De statusinformatie is nu als volgt gedefinieerd:

RECIPE_IDLE	RECIPE_LOADED
RECIPE_READY	RECIPE_EXECUTING
RECIPE_PENDING	RECIPE_ERROR
RECIPE_WAIT_ACK	

In [4] is de toestand RECIPE_WAIT_ACK weggelaten. Die toestand is daar alleen voor de cell controller gebruikt. De toestand wordt gebruikt om op een bevestiging van een MMS-request te wachten. Bij de DNC-services is dit op zich niet nodig, maar met het oog op de toekomst is deze definitie overgenomen, als eventueel verdere standaardisatie wordt doorgevoerd.

5.2. Logische verbindingen

Alleen als een logische verbinding is opgezet via het netwerk, kan communicatie plaats vinden. Omdat op elk tijdstip, tijdens het bedrijven van de cel, een besturingsmodule statusinformatie kan zenden, is het noodzakelijk, dat de verbindingen altijd open zijn als de cel in bedrijf is. De logische verbindingen moeten dus worden opgezet als de operator de cel aan zet. Om de operator hier van af te schermen, gebeurt het opzetten automatisch.

Voor het automatisch opzetten van de verbindingen, is het nodig, dat van te voren bekend is, naar welke AR-names (Application Reference) een verbinding moet worden opgezet. Hiertoe is een configuratiebestand van het demonstratieprogramma van Concord [6] uitgebreid. In dit bestand staat welke AR-names aan welke kanalen bekend moeten worden gemaakt. Eventueel kunnen sommige kanalen in LISTEN-mode worden gezet. Dit configuratiebestand is nu uitgebreid met een vierde onderdeel, waarin bekend wordt gemaakt, naar welke AR-names een verbinding moet worden opgezet. Een voorbeeld van het bestand staat in figuur 5.2.

```
#1 AR NAMES TO BE ACTIVATED
CELL_CONTROLLER
#2 AR NAMES TO BE REGISTERED ON CHANNELS
CELL_CONTROLLER
CELL_CONTROLLER
#3 CHANNELS TO BE LISTENING
#4 CHANNELS TO BE CONNECTED TO
WELDING
TRANSPORT
#5 END OF FILE
```

Figuur 5.2: Het bestand SUIC.CFG

In figuur 5.2 staat het configuratiebestand van de cell controller. In dit geval worden verbindingen gemaakt naar de workstations WELDING en TRANSPORT.

Zoals in 4.6 is gebleken, is het nodig dat, tijdens de uitvoering van de editor de logische verbindingen niet aanwezig zijn. Hier uit blijkt, dat zowel het opzetten als het opheffen van de logische verbindingen automatisch moet gebeuren. De workstations kunnen niet zelfstandig werken, zonder dat ze met de cell controller kunnen communiceren. Daarom heffen de PC/AT's hun verbindingen naar onderliggende apparaten ook op. Op deze manier kunnen zich bij die onderliggende apparaten meldingen opstapelen. Daarom wordt bij het weer opzetten van de verbindingen gecontroleerd welke meldingen uitstaan. Dit opzetten zal moeten gebeuren, als de cell controller en de workstations weer met elkaar kunnen communiceren.

Doordat de verbindingen automatisch worden opgebouwd, kan in de receptentaal nu niet meer naar kanaalnummers worden gerefereerd, zoals in [4]. Het systeem wijst immers automatisch de kanaalnummers toe. De kanaalnummers zijn nu vervangen door de AR-names. De verschillen worden geïllustreerd in figuur 5.3.

```
vorige versie:                  huidige versie:

Initiate 0 Welding              Start Welding dunne_plaat
Start    0 dunne_plaat
Conclude 0
```

Figuur 5.3: Receptentaal met en zonder kanaalnummers.

De logische verbindingen moeten tijdens het bedrijven van de cel kontinu aanwezig zijn. Er zijn drie mogelijkheden om dit te bereiken.

De eerste mogelijkheid bestaat hier uit, dat elk apparaat bij het aanzetten zijn verbindingen naar onderliggende apparaten opzet. Deze mogelijkheid heeft als nadeel, dat geen tweede keer verbindingen kunnen worden gezet. Zoals boven is gebleken, is dit wel nodig.

Als tweede mogelijkheid kunnen op elk nivo recepten worden aangemaakt, die de verbindingen naar onderliggende apparaten opzetten en daar ook een soortgelijk recept starten. De verbindingen worden op een recursieve manier opgebouwd. Het nadeel van deze methode is, dat in de receptentaal opdrachten moeten voorkomen, die met verbindingen te maken hebben. De operator, die deze recepten maakt, zou juist van de verbindingen moeten worden afgeschermd.

Voor de derde mogelijkheid wordt eerst naar het besturingsmodel gekeken. In het CAM-reference model heeft een ondergeschikte één opdrachtgever. Als het opbouwen van verbindingen als opdracht wordt gezien, dan legt die opdrachtgever de verbinding naar zijn ondergeschikte op. Omdat er maar een opdrachtgever is, kan de ondergeschikte op dat moment zijn verbindingen opzetten naar zijn ondergeschikten. Op elk apparaat worden dan op een consistente manier de verbindingen opgezet. Het hoogste apparaat in de hiërarchie neemt het initiatief en wel als de cel wordt aangezet. Dat apparaat kan op deze manier ook het initiatief nemen om alle verbindingen op te heffen.

Voor de laatste mogelijkheid is gekozen, omdat de nadelen van de andere mogelijkheden niet aanwezig zijn.

De opbouw van de verbindingen kan mis lopen, als apparatuur niet reageert. Als dat gebeurt bij een verbinding via MAP, dan blijft bij de client een request uitstaan voor het opzetten van de verbinding. Die request kan niet worden opgeheven met MMS-EASE, zodat ook geen nieuwe poging kan worden gedaan om de verbinding op te zetten. Tot nu toe blijkt, dat een nieuwe poging pas kan worden gedaan als de client wordt gereset. Hetzelfde geldt voor een logische verbinding via RS-232. In de praktijk zal dit probleem zich weinig voordoen, als de cel met de vaste procedure volgens bijlage B wordt opgestart.

Bij het opzetten van de verbindingen wordt bepaald hoeveel stations aangesloten zijn aan de AT-386. Vandaar, dat pas na het opzetten van de verbindingen de meldingenschermen van het user-interface worden toegewezen. Meldingen, die tijdens het opzetten van de verbinding komen worden op het scherm van het betreffende station gezet.

5.3. Afhandeling van meldingen

Een opdrachtgever krijgt terugmeldingen in de vorm van bevestigingen of ongevraagde statusinformatie. Bij beide kan het voorkomen, dat een alarm wordt teruggemeld, waardoor de cel niet verder kan werken. Voorbeelden van oorzaken van die alarmen zijn:

- synchronisatiefouten: een besturingsmodule bevindt zich in een andere toestand als het systeem zou kunnen verwachten.
- noodstops: iemand heeft op de noodstop gedrukt of de robot heeft een aanvaring gehad.
- fouten in recepten
- uitval van besturingsmodules

Bij al deze fouten grijpt het systeem zelf in, buiten de verantwoordelijkheid van de receptenschrijver om. De enige verantwoordelijkheid van de receptenschrijver bij een uit-

zonderingstoestand is, als een terugmelding uitblijft. In hoofdstuk 4 is de receptentaal verder uitgewerkt.

Als een besturingsmodule uitvalt, dan krijgt de module, waarnaar de uitgevallen module een verbinding opgezet had, een abort-indication als melding. De module die de abort-indication als melding krijgt, stopt onmiddelijk zijn werkzaamheden en zet zijn ondergeschikten stop. Dit, omdat de cel niet meer in zijn geheel te coördineren is.

Bij andere fouten dan uitval, zou de fout zijn oorzaak kunnen hebben in de verbindingen. Vandaar dat bij voorkomen van die fouten de verbindingen worden afgebroken met behulp van abort. Dit heeft bovendien het gewenste effect, dat de cel wordt stilgezet.

Na een alarm kan de operator de cel weer op gang brengen door met het user-interface start te kiezen. De verbindingen worden dan weer opgezet. Als dan een verbinding niet gemaakt kan worden, wordt de cel niet op gang gebracht.

5.4. Synchronisatie

De besturingsnivo's, die recepten op het onderliggende nivo starten, wachten op een gegeven moment totdat het gestarte recept klaar is. In een recept kan worden gewacht met de opdracht **wait**. De opdracht wacht op een unsolicited status. Uit de gegevens in de local detail van de statusinformatie wordt het dan geldige criterium afgeleid. De unsolicited status wordt alleen gegenereerd na het einde van een recept en in uitzonderingstoestanden. Als statusinformatie binnenkomt, wordt een klaar-vlag gezet. Per kanaal is een vlag gereserveerd. Dit is voldoende zolang de ingekomen informatie meteen wordt verwerkt.

Via het netwerk ingekomen informatie wordt alleen behandeld, als een library funktie van MMS-EASE wordt aangeroepen. Deze funktie wordt aangeroepen in een loop, waar ook de receptverwerker en op de workstations de download-statemachines aangeroepen worden. Van de aangeroepen funkties wordt geëist, dat ze na een eventuele aktie, snel de controle aan de loop teruggeven. Zo worden meerdere onderdelen van het besturingsprogramma quasi gelijktijdig uitgevoerd. Aangezien ingekomen informatie maar een keer per loop wordt behandeld en in die loop die informatie ook meteen wordt verwerkt, is een klaar-vlag per kanaal voldoende. Als de informatie van de unsolicited status voor verschillende onderdelen van het besturingsprogramma een andere betekenis heeft, dan is een klaar-vlag per kanaal niet voldoende. Dit laatste is nu niet het geval.

Nu is **wait** zo uitgevoerd, dat bij de opdracht zelf de tijd wordt opgegeven, die maximaal gewacht gaat worden. Als nu meerdere recepten op het onderliggende nivo meteen achter elkaar worden gestart, zonder te wachten (zie figuur 5.4), dan zal op een gegeven moment achter elkaar op de verschillende recepten moeten worden gewacht. Als de tijdbe-

waking strak gehouden moet worden, dan zal van de tijd, die gewacht gaat worden, de tijd, die al gewacht is, moeten worden afgetrokken. De tijd, die moet worden afgetrokken, is de minimum tijd, die de recepten duren, waar al op gewacht is. Voor de receptenschrijver is het bepalen van de af te trekken tijd niet erg vriendelijk.

```
start welding   las_pijp  (duurt maximaal 100 sec)
start transport store     (duurt minimaal  15 sec)
wait  transport 20        (wacht maximaal  20 sec)
wait  welding   85        (wacht maximaal  85 sec)
```

Figuur 5.4: Wachten in een recept.

Het wachten op het einde van een recept op lager nivo hangt samen met de opdracht **start**. De parameter wachttijd zou van de opdracht **wait** naar **start** verplaatst kunnen worden. De tijden, die dan in het recept moeten worden ingevuld zijn dan alleen maximum tijden. Deze laatste, betere invulling van de receptentaal is nog niet geïmplementeerd.

6. Onderhoud aan de programmatuur

In een aantal onderdelen is rekening gehouden met uitbreiding van de mogelijkheden van de cel. In de volgende paragrafen staat een uitleg hoe die uitbreidingen kunnen worden uitgevoerd.

6.1. Extra verbindingen

Wanneer een nieuw workstation wordt aangemaakt, zal ook naar de apparatuur, waar in dit workstation is ondergebracht, automatisch een verbinding moeten worden opgebouwd. Dit kan door in het configuratiebestand **suic.cfg** van de AT-386 een extra regel toe te voegen met de AR-name van dit workstation (zie 5.2). De toevoeging heeft automatisch tot gevolg, dat een extra meldingenscherm op het user-interface wordt gereserveerd.

Aan andere onderdelen van het programma moet ook duidelijk gemaakt worden, welke AR-names in gebruik zijn. Dit is aan te geven met de bestanden **suic.dib** en **tpy.dib**. De beschrijving van deze bestanden is te vinden in [4]. Daarnaast wordt door de software geheugen toegewezen per kanaal. Het aantal kanalen moet hiervoor als konstante toegewezen worden aan **DEFAULT_NUM_MMS_CHAN** in het bestand **mms_llp.h**.

6.2. Onderhoud aan het user-interface

Voor de operator is het wenselijk, dat sommige stations op het overzichtscherm van het user-interface worden gezet. In de variabele **pictures**, in de module **screen.c**, staat welk station bij welk plaatje op het overzichtscherm hoort. Een voorbeeld van invulling van **pictures** staat in figuur 6.1. De AR-name van een station wordt automatisch als titel gebruikt. Alleen als door toevoeging van een regel in **suic.cfg** een verbinding naar een AR-name wordt opgebouwd, verschijnt het bijbehorende plaatje op het scherm.

```
/* pictures of the stations */
struct stationview pictures[maxstations] =
{ { 2,10,  "WELDING",   2,15,    weldingchar }, /* WS welding, a robot            */
  { 8,35,  "TRANSPORT", 2,15,    transchar }    /* WS transport transportsystem   */
};
    positie station-    positie  coördinaten-
    titel   naam        plaatje  tabel
```

Figuur 6.1: Plaatjes op het voorbeeldscherm

6.3. Onderhoud aan de receptentaal

Zoals in 4.3 beschreven staat, worden in een recept opdrachten uitgevoerd en de bijbehorende criteria geëvalueerd. Voor elke mogelijke opdracht in een recept, is in de module

criteria.c een C-funktie voor de uitvoering van de opdracht en een C-funktie voor de evaluatie van de criteria opgenomen. Daarnaast is het noodzakelijk, om te weten welke criteria bij welke opdracht geldig zijn en van welke type die criteria zijn. Om de funkties en de criteria bij de opdracht te kunnen zoeken, is als constante een linked-list in de module opgenomen. De elementen van de linked-list staan toegelicht in figuur 6.2.

```
typedef struct
{ char *name;
  enum { defined, decimal, string } type;
} CRITERIUM;

struct LOCRITERIA
{ struct    LOCRITERIA *next;
  char      *command;          /* commandonaam */
  void      (*comm)();         /* funktie voor commando */
  int       (*eval)();         /* funktie voor evaluatie */
  CRITERIUM critarr[];         /* de geldige criteria */
};
```

Figuur 6.2: C-structuren voor de receptentaal.

Tot nu toe is de conventie aangenomen, dat de elementen van de linked list bij de funkties worden gezet, waar ze naar toe wijzen. Een nieuwe opdracht in de receptentaal kan als volgt worden toegevoegd:

- Vul een structuur **CRITERIUM** met criteria voor deze opdracht in en voeg deze vlak voor de definitie van de constante **CRITPTR** in de module **criteria.c** toe.
- Voeg de funktie, die de opdracht implementeert, daaronder aan de programmatekst toe.
- Voeg de funktie, die de criteria onderzoekt, na de vorige funktie toe. Gebruik verwijzingen naar de namen van de eerst ingevulde criteria.
- Vul een structuur **LOCRITERIA** in: het adres van de vorige structuur in de programmatekst, naam van de opdracht, funktie voor de opdracht, funktie voor de criteria en het adres van de bijbehorende **CRITERIUM** structuur.
- Wijzig aan het einde van de module de definitie van de constante **CRITPTR** in het adres van de laatste **LOCRITERIA** structuur.

7. Conclusies en aanbevelingen

Met gebruik van meer MMS-services, is de functionaliteit van de FALC uitgebreid. De verwerking van statusinformatie kan, afhankelijk van de informatie, automatisch of via een recept gebeuren. De vorm van een recept is hiervoor aangepast. Recepten zijn nu toestandstabellen, waardoor met behulp van statusinformatie beslissingen kunnen worden genomen. Deze toestandstabellen worden met een interpreter uitgevoerd, zodat de receptenschrijver nog altijd alleen met de eenvoudige receptentaal te maken heeft.

De beslissingen in de recepten worden genomen aan de hand van criteria. De criteria, gebaseerd op statusinformatie, blijken nog niet goed gekozen. De keuze van de criteria is een afweging tussen de mogelijkheden en de moeilijkheden van de programmering van de recepten. Om de juiste mogelijkheden van de recepten te bepalen, zal nog moeten worden onderzocht, welke criteria voor de beslissingen van belang zijn.

Het hoogste besturingsnivo van de FALC bestaat uit de operator. Deze operator legt orders op aan de cell controller. Een user-interface helpt de operator hierbij. De operator krijgt via dit interface informatie over de voortgang van de cel. Ook niet-PC-gebruikers kunnen via het user-interface de cel bedienen.

De operator kan via het user-interface ook de recepten schrijven voor de cell controller en de workstations. Het besturingsprogramma zorgt, nadat de recepten zijn geschreven, voor het downloaden van de recepten vanuit de AT-386 via het MAP-netwerk. Het downloaden blijkt sneller, dan het zoeken op harddisk naar het juiste bestand. Het lijkt dan ook verstandig, om altijd elk recept vanuit de AT-386 aan te leveren. Dit voorkomt ook de verdubbeling van geldige informatie.

De robotcontroller wordt aangestuurd met robotprogramma's. Om deze, net zoals de recepten voor de workstations, ook via de AT-386 aan te kunnen leveren, is het nodig om de robotcontroller ook via een VMD bereikbaar te laten zijn. Het communicatieprogramma van Concord ondersteunt meerdere VMD's, zodat een VMD voor de robotcontroller in een bestaande PC kan worden geïmplementeerd. Een implementatie van dit VMD levert bovendien een verdere standaardisatie van de netwerkaansturing op.

Literatuur

[1] Gransier T.A.G, Padt A. van der
Functionele Specificatie voor de besturing van de TUE Flexibele Assemblage en LasCel.
Eindhoven, TNO Institute of Information Technology for Production Automation, 1989.

[2] Wolf A.C.H. van der
Het onderzoekprogramma FALC.
In: Een studie naar een Flexibele Assemblage- en LasCel aan de TU te Eindhoven.
Technische Universiteit Eindhoven, 1992.

[3] Beukeboom J.J.A.J., Biemans F.P.M., Hehl C.J.G., Sjoerdsma S., Veen H.J. van.
Reference Model of Production Systems.
Eindhoven, Philips, CFT Report 01/89EN, 1989.

[4] Lammerts B.B.M.
Controlling a Flexible Assembly and Arc Welding Cell - Using MAP 3.0 and the MMS Application Interface.
Afstudeerrapport.
Eindhoven, TUE, Faculteit Elektrotechniek, Vakgroep ER, 1991.

[5] Tanenbaum A.S.
Computernetwerken.
Schoonhoven, Prentice Hall int., 1990.

[6] MMS Software Programmer's Reference Manual
Concord Communication Inc.
L7-M050-6200 (rev. 9)
Volume 1 en Volume 2.

[7] Vidal R.
Seriële Communicatie met een Industriële Robot.
Afstudeerrapport.
Eindhoven, TUE, Faculteit Elektrotechniek, Vakgroep ER, 1989.

[8] Michaeli W., Lauterbach M., Sinn T.
Rechnergesteuerte Spritzgießmachinen. Einheitliche Bedieneroberfläche an Bildschirmen.
Plastverarbeiter.
vol. 40, no. 5, 1989, p. 119-122.

[9] Schneiderman B.
Designing the user interface.
Amsterdam, Addison-Wesley, 1987

[10] Pace B.J.
Color combinations and contrast reversals on visual display units.
In: New frontiers for science and technology/
Ed. by Alluisi M.J., Groot S. de, Alluisi E.A..
Santa Monica, Human factors society, 1984
vol 2, p. 326-330.

[11] Letts S.
Use of the IBM personal computer in the man-machine interface to a nuclear research accelerator.
Computer Standards and interfaces.
vol. 6, no. 3, 1986, p. 331-340.

[12] Albus J.S., Barbera A.J., Nagel R.N.
Programming a Hierarchical Robot Control System.
Proc. 12th International Symposium on Industrial Robots.
Paris, 9-11 june 1982.

[13] Blonk P.
On the design of cell and workstation controllers for automated production systems.
Proefschrift.
Enschede, UT, 1991.

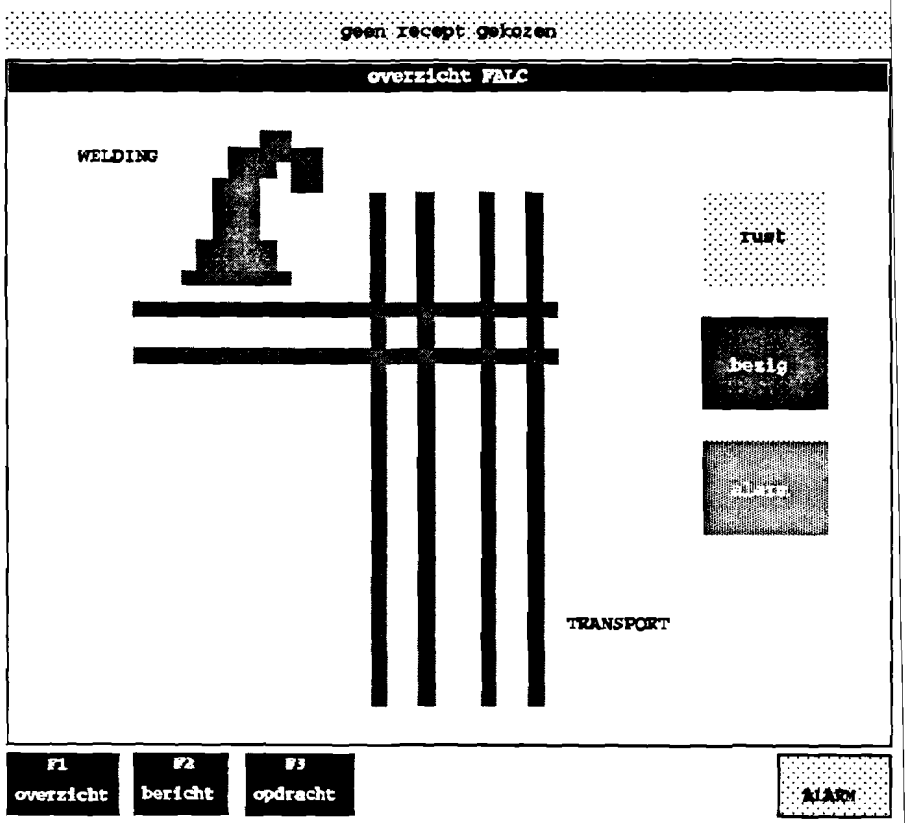
[14] Klop G.J.B.
Robotveiligheid; Praktische richtlijnen en wettelijke bepalingen.
Stageverslag WPA-1284
Eindhoven, TUE, Faculteit Werktuigbouwkunde, Vakgroep WPA, 1992.

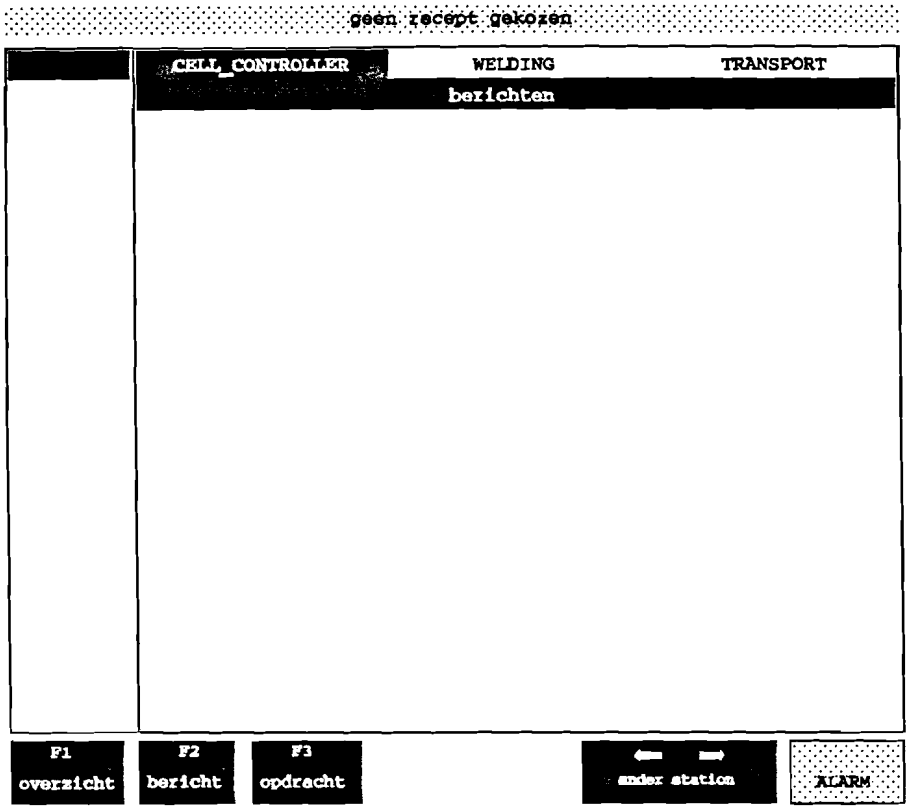
[15] Industrial Automation Systems for manufacturing environment.
Robot Companion Standard to MMS 9506/3.
ISO/TC, 1989.

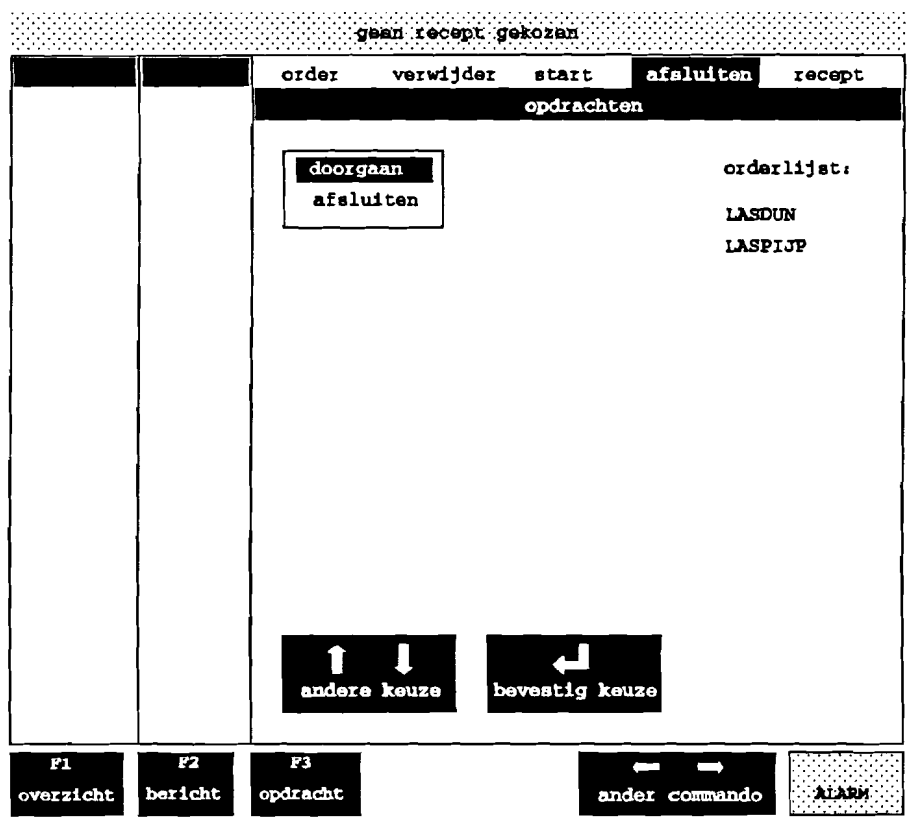
Woordenlijst

AR-name	Application Reference Name.
AT-386	IBM-compatible PC met 80386 processor
DNC	Direct Numerical Control.
EGA	Enhanced Graphics Adapter.
FALC	Flexibele Assemblage en LasCel.
LISTEN-mode	MMS-kanaalmode, waar externe verbindingsaanvraag is toegestaan.
MAP	Manufacturing Automation Protocol.
MMS	Manufacturing Message Specification.
OSI	Open Systems Interconnection.
PC/AT	IBM-compatible PC met 80286 processor
RS232	Een standaard voor digitale seriële verbindingen.
VGA	Video Graphics Array.

Bijlage A: Schermbeelden van het user-interface




geen recept gekozen
CELL_CONTROLLER
WELDING
TRANSPORT
berichten
F1
overzicht
F2
bericht
F3
opdracht
ander station
ALARM


geen recept gekozen
order
verwijder
start
afsluiten
recept
opdrachten
doorgaan
afsluiten
orderlijst:
LASDUN
LASPIJP
andere keuze
bevestig keuze
F1
overzicht
F2
bericht
F3
opdracht
ander commando
ALARM

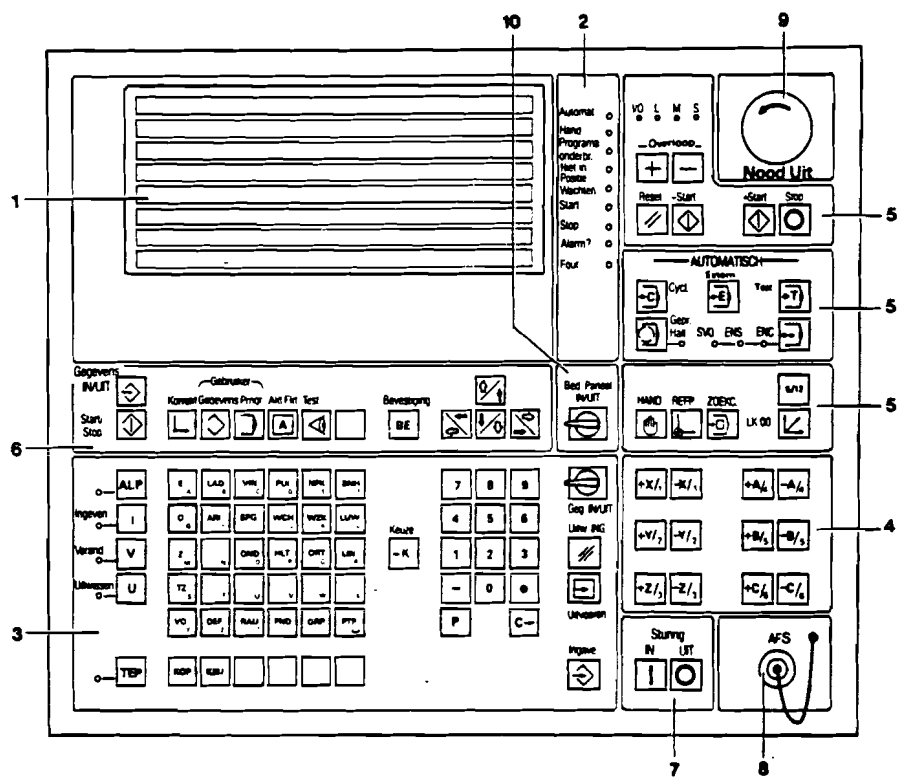
Bijlage B: Opstartprocedure van de cel

De cel wordt geheel automatisch aangestuurd nadat alle apparatuur is aangezet. De PC cell controller neemt het initiatief tot de aansturing, zodat alle andere apparatuur eerst moet worden geïnitialiseerd.

Om een logische volgorde aan te houden, wordt de procedure van onder naar boven in de besturingshiërarchie afgewerkt.

B.1. De KUKA robot controller

De robot controller kan via de RS232-verbinding worden aangestuurd, als voor de bediening DNC is geselecteerd. De software in de PC Welding eist bovendien, dat de controller in de mode Automatisch Extern staat. De handelingen om de controller in deze modes te zetten staan hieronder beschreven. De beschreven toetsen zijn te vinden aan de hand van onderstaande figuur.



Zet allereerst de hoofdschakelaar aan (groot en zwart aan de rechterkant).

Plaats de drie sleutels in de bestemde sloten, zet de vierkante sleutels horizontaal en zet de ronde sleutel vertikaal in de stand **T1**.

Druk de toets **Sturing IN** in veld 7. Na enige tijd antwoordt de controller met een klik en met het volgende scherm.

```
MAN SKO                                                    100 %
------------------------------------------------------------

RCM 815.03
```

Bevestig het bericht **RCM 815.03** met de toets **bevestiging** in veld 6. De controller antwoordt met het bericht **55 AANPASSING NIET KLAAR**. Druk nu de groene knop **Aandrijvingen IN** (boven de hoofdschakelaar). Na een klik in de kast komt de controller met het bericht **VOLLEDIG SYSTEEM SYNCHROON**. Verwijder dit bericht met de toets **bevestiging**.

Om in de mode AUtomatisch Extern te kunnen komen, moet de robot naar een referentiepunt worden gevaren. Hiertoe wordt robotprogramma nummer 10 uitgevoerd. Het uitvoeren gaat als volgt:

Toets **KEU** in veld 3. Onder aan in het venster wordt de regel **KZE HP "/AHP** getoond. Toets **10** via het numerieke toetsenbord. De regel wordt vervangen door **KZE HP 10 SN "/AD /DD /OP /CY**. Nadat de toets **ingave** is ingedrukt, is het programma geselecteerd. Het scherm ziet er nu als volgt uit:

```
MAN SKO HP 10                                  SN 1    100 %
------------------------------------------------------------



DEF HP 10*
```

In de mode MAN (manueel), waarin de controller nu staat, kan een robot programma slechts per regel worden uitgevoerd. Om het hele programma in een enkele keer uit te voeren, wordt de controller in de mode AUtomatisch Cyclisch gezet.

Draai de ronde sleutel een slag rechtsom naar **Auto** en toets **cycl** in veld 5. Nu kan programma 10 worden uitgevoerd.

Druk **+start**. Op het scherm verschijnt **RESET BEGINPUNT**. Druk weer **+start**. Op het scherm komt nu **DRUK START**. Doe dit en houdt de toets aan totdat op het scherm **DRUK START** verdwijnt en weer verschijnt. Druk dan nog een keer **+start** en wacht tot de controller uiteindelijk weer met **RESET BEGINPUNT** antwoordt. Nu is de robot naar het referentiepunt gevaren.

De mode AUtomatisch Extern wordt nu bereikt door **extern** in veld 5 te toetsen. De bediening via DNC kan nu als volgt worden gekozen:

Toets **Akt.Fkt.** in veld 6. Op het scherm verschijnen verschillende gegevens. Toets nu zolang $\Uparrow/\uparrow$ in veld 6, totdat het volgende scherm verschijnt:

```
AUE     HP 10                                        SN 6    100 %
------------------------------------------------------------------
EXTERN BEDRIJF
EX >
```

Toets nu **veranderen** in veld 3 en daarna de toets **keuze**, ook in veld 3. Nadat **ingave** is getoetst, is kan de controller via de RS232-verbinding worden bediend. Het scherm ziet er nu als volgt uit:

```
AUE DNC HP 10                                        SN 6    100 %
------------------------------------------------------------------
EXTERN BEDRIJF
EX > DNC
```

waarbij DNC in de bovenste regel knippert.

B.2. De PLC

De PLC kan extern worden bediend door een kaart in de PC, die de knoppen op de PLC-kast nabootst.

Nadat de hoofdschakelaar van het transportsysteem (naast de robot) is aangezet, kan de PLC worden gestart door de draaiknop van 0 naar 1 te draaien. Bij de hoofdschakelaar brandt nu nog een rode lamp. Deze lamp kan worden uitgezet door **RESET E.S.** te

toetsen. Gaat de lamp niet uit, dan is een van de noodstoppen ingedrukt. Toets in dit geval weer op **RESET E.S.**, nadat de noodstoppen met de sleutels zijn uitgezet.

Het systeem is nu bedrijfsklaar. Door op **START** op de PLC-kast te drukken worden de motoren gestart.

De draagblokken kunnen nog niet worden herkend, zodat tot nu toe de afspraak is aangehouden, dat de draagblokken in vaste volgorde op plaats W2 staan (zie kaart op de PLC). Die vaste volgorde bestaat uit blok 1 het dichtst bij de PLC en blok 2 er achter. Als de draagblokken niet in deze volgorde staan, zet ze dan in deze volgorde door de juiste baanvakken te aktiveren.

De PLC kan extern bediend worden door de witte knop onder de knoppen voor de baanvakken in te schakelen. Achter de knop gaat een lamp branden.

B.3. De PC's

Om de PC's te kunnen gebruiken voor de besturing, wordt de netwerkkaart geïnitialiseerd en het besturingsprogramma gestart. Voer de procedure eerst bij de PC's welding en transport uit en daarna bij de PC cell controller.

De computer wordt opgestart, door de computer aan te zetten, terwijl géén disk in de bovenste diskdrive aanwezig is. Na verloop van tijd komen op het scherm twee blauwe kolommen. Nu kunnen commando's aan de computer worden gegeven.

De netwerkkaart wordt geïnitialiseerd door achtereenvolgens op funktietoets **F2** en funktietoets **F6** te drukken. Na de initialisatie komen de twee kolommen weer op het scherm.

Met achtereenvolgens funktietoets **F2** en funktietoets **F7** wordt het besturingsprogramma gestart. De workstations zijn nu gereed voor gebruik.

De PC cell controller kan, als de opbouw van de netwerkverbindingen goed gaat, na een kleine wachttijd worden bediend via het user-interface. Zijn de netwerkverbindingen niet op te bouwen, dan meldt de PC cell controller dit en stopt de uitvoering van het besturingsprogramma. Zijn de andere apparaten eerder aangezet dan de PC cell controller, dan worden in principe de verbindingen opgebouwd.

De drie PC's kunnen ook worden geïnitialiseerd door achtereenvolgens de commando's

```
initmap
c:
cd c:\falcdemo
mmsopp
```

vanaf de command-line te geven.

Bijlage C: Gebruiksaanwijzing van het user-interface

Nadat op de PC cell controller het besturingsprogramma is gestart, kan de cel worden bediend via het user-interface. Het interface bestaat uit drie lagen, namelijk het overzicht van de cel, de berichten van de verschillende apparaten en een laag voor het geven van opdrachten. Een voorbeeld van de lagen staat in bijlage A.

Tussen de verschillende lagen kan worden geschakeld met de funktietoetsen **F1**, **F2** en **F3**. Deze toetsen zijn links boven op het toetsenbord te vinden.

C.1. Overzicht van de cel

Voor grove statusinformatie over de stations kan het overzicht van de cel worden getoond via funktietoets **F1**.

C.2. Berichten van de apparaten

Komen er berichten, dit is te zien aan het zwart worden van de beschrijving van funktietoets **F2**, dan kunnen ze gelezen worden door funktietoets **F2** te drukkken. De beschrijving van de funktietoets wordt weer inaktief, als de berichten van elk apparaat zijn gelezen. De verschillende apparaten zijn te selecteren met de cursortoetsen ← en →. Deze toetsen zijn onder aan het toetsenbord te vinden, links van het numerieke gedeelte.

Via de laag berichten worden ook vragen aan de operator gesteld. Het enige wat de operator dan kan doen, is die vraag beantwoorden. Verder heeft de operator dan geen mogelijkheden om de cel te bedienen. De vraag is altijd in beeld.

C.3. Opdrachten aan de cel

Met behulp van de laag opdrachten, kan de operator de cel laten produceren. De operator kan een orderlijst samenstellen uit recepten, deze orderlijst uit laten voeren en kan nieuwe recepten aanmaken. Tussen de de verschillende opdrachten kan worden geschakeld met de cursortoetsen ← en →.

C.3.1. Samenstellen van de orderlijst

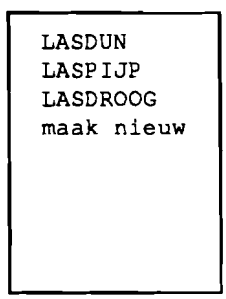
Een orderlijst samenstellen kan met de opdrachten **order** en **verwijder**. Het selecteren van een recept, dat in de orderlijst moet komen of dat uit de orderlijst verwijderd moet worden, kan door met de cursortoetsen ↑ en ↓ naar het recept te lopen en de ←**Enter** toets te drukken.

C.3.2. Uitvoeren van de orderlijst

Met de opdracht **start** kan de orderlijst worden gestart. Kies **starten** uit het menu, zoals ook de recepten geselecteerd zijn. Als een orderlijst wordt uitgevoerd, dan heeft de opdracht **start** geen zin, er wordt dan ook geen menu gekozen.

C.3.3. Bewerken van de recepten

Met de opdracht **recept** kunnen recepten worden bewerkt en aangemaakt. Nadat deze opdracht is gekozen, kan het apparaat worden geselecteerd, waar het te bewerken of aan te maken recept aanwezig moet zijn. Nu kan of een bestaand recept worden gekozen, of **maak nieuw**, zie onderstaande figuur.



Nadat gekozen is, wordt de NORTON-editor aangeroepen. Een bescrijving van bediening van deze eenvoudige tekstverwerker kan worden opgeroepen door funktietoets **F1** te drukken.

C.3.4. Afsluiten van de cel

De cel kan worden afgesloten via de opdracht **sluit af**. Dit kan alleen nadat alle orders uit de orderlijst zijn afgewerkt. Dit is het geval als de bovenste regel de tekst **geen recept gekozen** bevat. Nadat **afsluiten** in het menu is geselecteerd, wordt het besturingsprogramma verlaten.

Bijlage D: Afsluitprocedure van de cel

Het afsluiten van de cel wordt begonnen door het besturingsprogramma van de PC cell controller te verlaten (zie bijlage C). Daarna kunnen de PC's worden uitgeschakeld.

De robot controller kan weer met de hand en via het toetsenbord worden bestuurd door achtereenvolgens op **stop** en **HAND** in veld 5 te drukken. Zet na deze handelingen de controller uit door op de toets **sturing UIT** te drukken en de hoofdschakelaar om te draaien.

De PLC wordt op handbediening gezet door de witte knop met de lamp uit te drukken. Zet het hele systeem via de draaiknop op de PLC en via de hoofdschakelaar uit.

Alle systemen staan nu uit.

Eventueel kan, door het verlaten van het besturingsprogramma, met de PC welding of de PC transport verder worden gewerkt. Druk hiertoe, na het verlaten van het besturingsprogramma van de PC cell controller, op funktietoets **F10**. Dan wordt de vraag gesteld: "**Do you really want to exit ?**". Beantwoordt deze vraag met y en ←**Enter**. Dan is de computer voor ander gebruik gereed.

Bijlage E: Handleiding tot het schrijven van een recept

Het schrijven van recepten voor de cell controller en de workstations wordt via het user-interface uitgevoerd. Het user-interface zet de recepten zelf op de goede plaats weg.

Eenmaal in de Norton editor (zie bijlage C), wordt gevraagd om een toets te drukken, of, als een nieuw recept wordt gemaakt, om een bestandsnaam in te typen. De bestandsnaam **moet** eindigen met de extensie **.rcp**. Nu kan worden begonnen met het opstellen van de tabel. De tabel heeft de volgende kolommen:

toestands nummer	opdracht	rapportage	criterium	volgende toestand

Een tekstregel in een tabel ziet er dan als volgt uit:

- Helemaal links op de regel komt een - (min). Het min-teken onderscheidt de regels in de tabel van de commentaarregels. Commentaarregels zijn dus regels zonder min-teken aan het begin.
- Direkt achter het min-teken komt de eerste kolom, het huidige toestandsnummer.
- Na een kolomscheiding komt volgende kolom. Herhaal dit punt totdat de laatste kolom is bereikt.

Een kolomscheiding bestaat uit een van volgende tekens: | en ;.

Het is aan te raden om ook in de tekst op het scherm kolommen te maken door middel van de **Tab** ↹ toets. Een voorbeeld van de invulling van de kolommen is te vinden in bijlage F.

De opdrachten, die in een recept op een bepaald nivo kunnen worden gebruikt, staan in de onderstaande tabel.

opdracht	controller	omschrijving
start	cell welding	start een recept op onderliggend nivo
wait	cell welding transport	wacht op terugmelding van onderliggend nivo
activate	transport	druk een baanvak- of plaatsknop op de PLC in
reset	transport	druk een baanvak- of plaatsknop op de PLC uit
output	cell	geef tekst door aan operator
input	cell	stel vraag aan operator en wacht op antwoord

Bij elke opdracht horen criteria. Ze staan met hun betekenis hieronder opgesomd.

opdracht	bijbehorende criteria	omschrijving
start	gestart	recept gestart
	tijdlimiet	geen antwoord van onderliggend nivo
wait	klaar	recept afgelopen
	tijdlimiet	recept niet op tijd afgelopen
activate	(geen criterium)	
reset	(geen criterium)	
output	(geen criterium)	
input	answer=..	vergelijking van operator invoer

Met behulp van deze criteria wordt naar een volgende toestand gesprongen. Eén toestandsnummer is bijzonder, namelijk nummer 0. Springen naar **toestand 0** betekent het beëindigen van het recept.

In de kolom rapportage kan alleen **alarm** worden gerapporteerd. Rapportage van **alarm** heeft tot gevolg, dat de cel wordt stilgezet.

Op workstation nivo heeft een recept een extra regel nodig. In deze regel staat de naam, waarmee vanaf de cell controller het recept kan worden gestart. Deze regel begint met een #. Daarna komt direkt de referentienaam van het recept (zie bijlage F).

Is het recept geschreven, dan kan weer verder worden gegaan met de celbesturing, nadat achtereenvolgens funktietoets **F3** en **e** zijn ingedrukt. De rest van de commando's van de NORTON-editor zijn op te vragen met funktietoets **F1**.

Bijlage F: Aangemaakte recepten

F.1. Recepten voor de cell controller

filename: dikdroog.rcp

voorbeeld recept met toestanden op cell-line nivo

naar toestand 0 = einde recept

toest. nr.	opdracht	terugmelding naar hoger nivo	criterium	volgende toest.nr.
-1	output			2
-2	output dikke plaat droog			3
-3	output staat blok 1 in de uitgangspositie ?			4
-4	output staat blok 2 achter blok 1 ?			5
-5	output is de dunne plaat verwijderd ?			6
-6	input (ja/nee)		answer =ja	7
-6	input (ja/nee)		answer =nee	8
-6	input (ja/nee)		answer =*	6
-7	output dikke plaat droog			9
-8	output einde			0
-9	start transport store		gestart	10
-9	start transport store		tijdlimiet	19
-10	wait transport 60		klaar	11
-10	wait transport 60		tijdlimiet	19
-11	start transport move_to_man		gestart	12
-11	start transport move_to_man		tijdlimiet	19
-12	wait transport 60		klaar	13
-12	wait transport 60		tijdlimiet	19
-13	start welding droog_dik		gestart	14
-13	start welding droog_dik		tijdlimiet	19
-14	start transport unstore		gestart	15
-14	start transport unstore		tijdlimiet	19
-15	wait transport 70		klaar	16
-15	wait transport 70		tijdlimiet	19
-16	wait welding 100		klaar	17
-16	wait welding 100		tijdlimiet	19
-17	start transport unload		gestart	18
-17	start transport unload		tijdlimiet	19
-18	wait transport 70		klaar	20
-18	wait transport 70		tijdlimiet	19
-19	output er is wat mis	alarm		0
-20	output einde dikke plaat droog			0

filename: dundroog.rcp

voorbeeld recept met toestanden op cell-line nivo

naar toestand 0 = einde recept

toest. nr.	opdracht	terugmelding naar hoger nivo	criterium	volgende toest.nr.
-1	output			2
-2	output dunne plaat droog			3
-3	output staat blok 1 in de uitgangspositie ?			4
-4	output staat blok 2 achter blok 1 ?			5
-5	output is de dunne plaat gemonteerd ?			6
-6	input (ja/nee)		answer=ja	7
-6	input (ja/nee)		answer=nee	8
-6	input (ja/nee)		answer=*	6
-7	output begonnen			9
-8	output einde			0
-9	start transport store		gestart	10
-9	start transport store		tijdlimiet	19
-10	wait transport 60		klaar	11
-10	wait transport 60		tijdlimiet	19
-11	start transport move_to_man		gestart	12
-11	start transport move_to_man		tijdlimiet	19
-12	wait transport 60		klaar	13
-12	wait transport 60		tijdlimiet	19
-13	start welding droog_dun		gestart	14
-13	start welding droog_dun		tijdlimiet	19
-14	start transport unstore		gestart	15
-14	start transport unstore		tijdlimiet	19
-15	wait transport 70		klaar	16
-15	wait transport 70		tijdlimiet	19
-16	wait welding 200		klaar	17
-16	wait welding 200		tijdlimiet	19
-17	start transport unload		gestart	18
-17	start transport unload		tijdlimiet	19
-18	wait transport 70		klaar	0
-18	wait transport 70		tijdlimiet	19
-19	output er is wat mis	alarm		0

```
 filename: pipdroog.rcp

 naar toestand 0 = einde recept

toest.    opdracht                          terugmelding   criterium     volgende
nr.                                         naar hoger nivo               toest.nr.

-1      | output pijp-flens droog      |                |               | 2
-2      | output staat blok 1 in uitgangspositie ? | |               | 3
-3      | output staat blok 2 achter blok 1      ? | |               | 20
-20     | input klopt het (ja/nee)     |                | answer=ja     | 21
-20     | input klopt het (ja/nee)     |                | answer=nee    | 22
-20     | input klopt het (ja/nee)     |                | answer=*      | 20
-21     | output begonnen              |                |               | 4
-22     | output einde pijp-flens droog |               |               | 0
-4      | start transport move_to_man  |                | gestart       | 5
-4      | start transport move_to_man  |                | tijdlimiet    | 15
-5      | wait transport 60            |                | klaar         | 6
-5      | wait transport 60            |                | tijdlimiet    | 15
-6      | start transport store        |                | gestart       | 7
-6      | start transport store        |                | tijdlimiet    | 15
-7      | start welding droog_pijp     |                | gestart       | 8
-7      | start welding droog_pijp     |                | tijdlimiet    | 15
-8      | wait transport 60            |                | klaar         | 10
-8      | wait transport 60            |                | tijdlimiet    | 15
-10     | wait welding 100             |                | klaar         | 11
-10     | wait welding 100             |                | tijdlimiet    | 15
-11     | start transport unload       |                | gestart       | 12
-11     | start transport unload       |                | tijdlimiet    | 15
-12     | wait transport 70            |                | klaar         | 13
-12     | wait transport 70            |                | tijdlimiet    | 15
-13     | start transport unstore      |                | gestart       | 14
-13     | start transport unstore      |                | tijdlimiet    | 15
-14     | wait transport 70            |                | klaar         | 22
-14     | wait transport 70            |                | tijdlimiet    | 15
-15     | output er is wat mis         | alarm          |               | 0
```

F.2. Recepten voor het workstation welding

```
 pijp-flens droog

 filename: test64.rcp

 referentienaam voor start-opdracht vanuit cell controller
#DROOG_PIJP

-1 | start rcm 64     |              | gestart      | 2
-1 | start rcm 64     |              | tijdlimiet   | 3
-2 | wait  rcm 80     |              | klaar        | 0
-2 | wait  rcm 80     |              | tijdlimiet   | 3
-3 |                  | alarm        |              | 0
```

```
 dikke plaat droog

 filename: test65.rcp

#DROOG_DIK

-1 | start rcm 65     |              | gestart      | 2
-1 | start rcm 65     |              | tijdlimiet   | 3
-2 | wait  rcm 170    |              | klaar        | 0
-2 | wait  rcm 170    |              | tijdlimiet   | 3
-3 |                  | alarm        |              | 0
```

```
 pijp-flens droog

 filename: test66.rcp

#DROOG_DUN

-1 | start rcm 66              |                | gestart       | 2
-1 | start rcm 66              |                | tijdlimiet    | 3
-2 | wait  rcm 170             |                | klaar         | 0
-2 | wait  rcm 170             |                | tijdlimiet    | 3
-3 |                           | alarm          |               | 0
```

F.3. Recepten voor het workstation transport

```
WORKSTATION TRANSPORT RECIPE : MOVE TO MANIPULATOR

Possibilities :

activate    A..H and J..L (line)
activate    1..3 (station)
reset       A..H and J..L (line)
waitline    A..H and J..L  Wait till line .. active(1) or not active(0)
waitplace   2,3,4,5,6      Wait till place .. empty(0) or occupied (1)
waitstation 1,2,3          Wait till station .. busy(1) or not busy(0)

 filename: testrcp1.rcp

 referentienaam voor start-opdracht vanuit cell controller
#MOVE_TO_MAN

-1 | Activate E               |                |               | 2
-2 | WaitPlace 5 1 60         |                | klaar         | 3
-2 | WaitPlace 5 1 60         |                | tijdlimiet    | 6
-3 | Reset E                  |                |               | 4
-4 | Activate 3               |                |               | 5
-5 | WaitStation 3 1 30       |                | klaar         | 0
-5 | WaitStation 3 1 30       |                | tijdlimiet    | 6
-6 |                          | alarm          |               | 0
```

```
WORKSTATION TRANSPORT RECIPE : STORE CARRIER NEAR MANIPULATOR

 filename: testrcp2.rcp

#STORE

-1 | activate k               |                |               | 2
-2 | waitplace 3 1 40         |                | klaar         | 3
-2 | waitplace 3 1 40         |                | tijdlimiet    | 4
-3 | reset k                  |                |               | 0
-4 | reset k                  | alarm          |               | 0
```

WORKSTATION TRANSPORT RECIPE : UNLOAD MANIPULATOR AND LOAD MANIPULATOR WITH SECOND CARRIER

filename: testrcp3.rcp

#UNLOAD_AND_LOAD

- 1	Reset 3			2
- 2	Activate F			3
- 3	WaitLine F 1 30		klaar	4
- 3	waitline f 1 30		tijdlimiet	5
- 4	Reset F			6
- 5	reset f	alarm		0
- 6	Activate D			7
- 7	WaitLine D 1 30		klaar	8
- 7	waitline d 1 90		tijdlimiet	9
- 8	reset d			10
- 9	Reset D	alarm		0
-10	Activate H			11
-11	WaitPlace 3 0 30		klaar	12
-11	waitplace 3 0 30		tijdlimiet	13
-12	Reset H			14
-13	reset h	alarm		0
-14	WaitPlace 5 1 30		klaar	15
-14	waitplace 5 1 30		tijdlimiet	18
-15	Activate 3			16
-16	WaitStation 3 1 30		klaar	0
-17	waitstation 3 1 30		tijdlimiet	18
-18		alarm		0

WORKSTATION TRANSPORT RECIPE : UNLOAD MANIPULATOR

filename: testrcp4.rcp

#UNLOAD

- 1	Reset 3			2
- 2	Activate F			4
- 4	Waitstation 3 0 15		klaar	5
- 4	waitstation 3 0 15		tijdlimiet	6
- 5	reset f			7
- 6	Reset F	alarm		0
- 7	WaitPlace 4 1 15		klaar	8
- 7	waitplace 4 1 15		tijdlimiet	13
- 8	Activate D			9
- 9	WaitLine D 1 5		klaar	10
- 9	waitline d 1 5		tijdlimiet	12
-10	Reset D			11
-11	waitline d 0 20		klaar	0
-11	waitline d 0 20		tijdlimiet	13
-12	reset d	alarm		0
-13		alarm		0

WORKSTATION TRANSPORT RECIPE : REMOVE CARRIER FROM STORE TO OUTPUT

filename: testrcp5.rcp

#UNSTORE

- 1	Activate L			2
- 2	waitplace 4 1 15		klaar	3
- 2	waitplace 4 1 15		tijdlimiet	4
- 3	reset l			6
- 4	reset l	alarm		0
- 6	activate d			7
- 7	WaitLine D 1 5		klaar	8
- 7	waitline d 1 5		tijdlimiet	9
- 8	reset d			10
- 9	reset d	alarm		0
-10	waitline d 0 20		klaar	0
-10	waitline d 0 20		tijdlimiet	11
-11		alarm		0

Bijlage G: Beschrijving van het produktie voorbereidingssysteem

Het produktie voorbereidingssysteem levert recepten aan het produktie besturingssysteem. De recepten voor de workstations en de cell controller kunnen worden aangemaakt door de operator. Ze worden centraal verzameld op de harddisk van de AT-386. Op de harddisks van de PC/AT's staan lokale kopieën.

Op de bovengenoemde PC's worden de recepten ingelezen en gecontroleerd door funkties in module **pps.c**. Een beschrijving van die funkties staat in bijlage J. De tekst van het recept wordt in zijn geheel in het geheugen gelezen, nadat is bepaald hoeveel ruimte dit recept zal innemen. Daarmee is een probleem in de vorige versie verholpen. Het probleem was, dat grote recepten niet altijd pasten. Een recept wordt uit de huidige directory gelezen, tenzij een environment variabele van MS-DOS is gedefinieerd, die een andere directory aanwijst. Die environment variabele heet **MMSOPP**. De variabele kan bijvoorbeeld worden gedefinieerd door:

```
set mmsopp=c:\falcdemo
```

Tussen **MMSOPP** en = mag geen spatie staan.

Bij het opstarten van het besturingsprogramma van de PC's, wordt op de eigen harddisk gezocht naar alle aanwezige recepten. Op de AT-386 worden de bestandsnamen van de gevonden recepten in de lijst gezet, waar de operator de orderlijst mee kan samenstellen. Op de PC/AT's worden de bestandsnamen als domains aangemeld bij de MMS-EASE software. Het aanmelden van de domains wordt gedaan door de funktie **set_user_pis** in module **u_data.c**. Deze funktie is zo gewijzigd, dat alle recepten op harddisk worden aangemeld als domain, terwijl de program invocation name in elk recept meteen als program invocation wordt aangemeld.

Nadat een recept voor de cell controller is ingelezen, wordt gecontroleerd, of een recept op een workstation zal worden gestart. Is dit het geval, dan wordt gekeken of de naam van het recept al als program invocation aanwezig is met de MMS service GetProgramInvocationAttributes. Geeft deze service via foutcodes te kennen dat dit object niet aanwezig is, dan wordt op de disk van de AT-386 gezocht naar het domain, waarvan de bijbehorende program invocation name gelijk is aan de naam van het te starten recept. Is het domain gevonden, dan wordt met de MMS download-service het domain overgebracht. Hierna wordt via de MMS service CreateProgramInvocation een program invocation aangemaakt.

Nadat de taal van een recept is gecontroleerd, kan tot de uitvoering van het recept worden overgegaan.

Bijlage H: Beschrijving van het produktie besturingssysteem

Aan het produktie besturingssysteem is gewerkt op de volgende controllers: factory controller, cell controller en workstation controller. Zowel de cell controller als de workstation controllers werken met recepten in tabelvorm, zodat het produktiebesturingssysteem van die controllers ook sterk op elkaar lijkt. De factory controller, een combinatie van de orderverwerking en van de operator, is anders opgebouwd.

H.1. De cell controller en de workstations

Het produktiebesturingssysteem van de cell controller en de workstations bestaat uit de verwerking van de recepten. Voor de verwerking van de recepten zijn een aantal funkties nodig:

- Funkties die de opdrachten uitvoeren en de criteria evalueren, te vinden in de module **criteria.c**.
- Funkties die het recept interpreteren, te vinden in de module **pcs.c** en **linefun.c**.
- Funkties die de funkties en criteria bij de opdrachten zoeken, te vinden in de module **critsrch.c**.

De funktieheadings, omschrijving van de funkties, constantes en typedefinities, die samenhangen met de receptverwerking zijn in bijlage J te vinden.

De funkties die de receptentaal implementeren, in de module **criteria.c**, zijn per controller verschillend. Deze funkties maken gebruik van de mogelijkheden van de communicatie met de onderliggende laag. De funkties worden per controller behandeld.

H.1.1. De cell controller

De receptentaal van de cell controller bestaat uit de opdrachten **start**, **wait**, **output** en **input**. De naamgeving van de funkties is als volgt:

***_comm**:	Voert de opdracht uit.
***_crit**:	Evalueert het als parameter opgegeven criterium.

De **start**-opdracht:

start_comm:	Roept MMS start-service aan, zet de receptverwerker in de toestand RECIPE_WAIT_ACK en zet een maximale wachttijd.
start_crit:	Verwerkt ISO foutcodes, die gekopieerd zijn door de confirm funktie van de MMS start-service.

De **wait**-opdracht:

wait_comm: Zet de receptverwerker in de toestand RECIPE_PENDING en zet een maximale wachttijd.

wait_crit: Verwerkt de receptstatus en de klaar-vlag, die gekopieerd is door de indication funktie van de MMS unsolicited status service. De klaar-vlag geeft aan, dat een recept op een onderliggend nivo afgelopen is.

De **output**-opdracht:

output_comm: Roept de user-interface funktie **do_auto_output** aan.

De **input**-opdracht:

input_comm: Roept een user-interface funktie **do_auto_input** aan.

input_crit: Vergelijkt de invoer van de operator.

De ISO foutcodes, die bij de start-service horen, bevatten ook de program invocation state. Uit de source van de vorige versie blijkt, dat deze state impliciet door de MMS-EASE services zou worden aangepast. Dit blijkt niet zo te zijn. Het gebruikersprogramma moet zelf met behulp van de library funktie **ms_change_pi_state** de state wijzigen.

H.1.2. Het workstation welding

De receptentaal van het workstation welding bestaat uit de opdrachten **start** en **wait**.

De **start**-opdracht:

start_comm: Roept de DNC-services **select_program** en **start_program** aan.

start_crit: Verwerkt ISO foutcodes, die gegenereerd zijn door de twee DNC-services.

De **wait**-opdracht:

wait_comm: Zet de receptverwerker in de toestand RECIPE_PENDING en zet een maximale wachttijd.

wait_crit: Verwerkt de statusinformatie, die door de opvang van RCM alarmcodes is gegenereerd. De klaar-vlag wordt door opvang van andere RCM meldingen gezet.

De DNC-services hebben een wijziging ondergaan in vergelijking met de vorige versie. Het interface tussen het gebruikersprogramma en de DNC-services ziet er als volgt uit:

int select_program — Selecteert het te gebruiken robotprogramma en kopieert foutcodes naar een structuur in **criteria.c**.

int start_program — Start het geselecteerde robotprogramma en kopieert foutcodes naar een structuur in **criteria.c**.

int stop_program	Zet het geselecteerde robotprogramma stop.
int initiate_DNC	Zet de DNC-services in gang.
void R1_RESTORE_RS232	Sluit de DNC-services af.
void Watch	Bekijkt of berichten via DNC binnenkomen of uitgaan.

Een andere funktie, die van belang is voor de communicatie, is de funktie **handle_alarm**, die de alarmcodes van de robot controller verwerkt. Deze funktie zet de alarmcodes om naar statusinformatie gedefinieerd in de Robot Companion Standard. De omzetting gebeurt met behulp van het array uit bijlage I.

Om produktiegegevens te kunnen verzamelen, worden via het user-interface gedetailleerde meldingen aan de operator doorgegeven. Voor dit doorgeven wordt de MMS output-service gebruikt. De implementatie van deze service door Concord heeft een eigenaardigheid, namelijk dat alleen losse regels kunnen worden gestuurd. Dit is overigens in overeenstemming met ISO 9506, waar staat, dat deze operator communicatie zeer simpel moet zijn. Het sturen van losse regels gaat echter zo ver, dat regels met een return-karakter als einde, als lege regels aankomen. De service geeft geen foutmelding in dit geval. De service kan worden aangestuurd via de funktie **do_auto_output** in de module **mmsaocs.c**.

H.1.3. Het workstation transport

De receptentaal van het workstation welding bestaat uit de opdrachten **activate**, **reset**, **waitline**, **waitplace** en **waitstation**.

De **activate**-opdracht:

activate_comm: activeert met behulp van een PLC interface funktie een baanvak of een station.

De **reset**-opdracht:

reset_comm: reset met behulp van een PLC interface funktie een baanvak of een station.

De **wait***-opdracht:

wait*_comm: Zet de receptverwerker in de toestand RECIPE_PENDING en zet een maximale wachttijd.

wait*_crit: Verwerkt de klaar-vlag, gezet door telkens te kijken naar de bits van de I/O-kaart.

Ook dit workstation kan met behulp van de MMS output-service gedetailleerde meldingen aan de operator doorgeven. De funkties, die als interface dienen tussen de I/O-kaart en het gebruikersprogramma zijn ongewijzigd gebleven.

H.2. De factory controller

De software voor de factory controller bestaat uit de orderverwerking. De funkties daar voor zijn te vinden in de module **orders.c**. De funktie headings uit bijlage K spreken voor zich. De implementatie is eenvoudig.

Bijlage I: Vertaling van DNC-foutcodes naar Robot Status Detail

```
/*                                                                      */
/* MODULE NAME : messages.h                                             */
/* PRODUCT(S)  : Workstation Welding                                    */
/*                                                                      */
/* MODULE DESCRIPTION : This module contains message mapping of the     */
/*                      RCM to the status defined in the Robot          */
/*                      Companion Standard                              */
/*
   Robot Specific Status protocol (7.2.1.2 of Robot Companion Standard)

   high byte of messag_mms is robotVMDState extended with
        6: send confirm to robot
        7: do nothing
   low  byte of messag_mms is RobotSpecificStatus
        bit 0: safety interlocks violated
        bit 1: any physical resource power on (not used)
        bit 2: all physical resources calibrated
        bit 3: local control
        bit 4: units of measure (not used)

   array number = robot message number
*/

static
short messag_mms[255] = { 0x0504,                                   /*  0 */
                          0x050c,0x0504,0x0604,0x0504,0x0504,
                          0x0504,0x0504,0x050c,0x0504,0x0504,       /* 10 */
                          0x0504,0x0504,0x0504,0x0504,0x0504,
                          0x0504,0x0504,0x0504,0x0504,0x0504,       /* 20 */
                          0x0504,0x0504,0x0504,0x0505,0x0504,

                           ...
```

Bijlage J: Definities voor de receptverwerking

```
/*****************************************************************************/
/*  MODULE HEADER    *********************************************************/
/*****************************************************************************/
/*                                                                           */
/* MODULE NAME : recipes.h                                                   */
/* PRODUCT(S)  : Cell/line controller and Workstations                       */
/*                                                                           */
/* MODULE DESCRIPTION : This module contains the type defintions and         */
/*                      function definitions related to the recipes          */
/*                                                                           */
/*                                                                           */
/* GLOBAL FUNCTIONS DEFINED IN THIS MODULE :                                 */
/*                                                                           */
/* MODIFICATION LOG :                                                        */
/*  Date     Who   Rev      Comments                                         */
/* --------  ---  ------   ------------------------------------------       */
/* 02/03/92  SCH           Creation.                                         */
/*****************************************************************************/

/* prevent additional including */
#ifndef RESPDEFS_INCLUDED
#define RESPDEFS_INCLUDED

/* constant definitions */

/* recipe line separators */
#define NAMESEP " |;\b\f\r\n\t="
#define TEXTSEP " \b\t\n\f\r"
#define PARTSEP "|;"

/* recipe state */
#define RECIPE_IDLE        0
#define RECIPE_LOADED      1
#define RECIPE_READY       2
#define RECIPE_EXECUTING   3
#define RECIPE_PENDING     4
#define RECIPE_ERROR       5
#define RECIPE_WAIT_ACK    6

#define TEXT_PI_NON_EXISTENT "Recept bestaat niet op WS %s"
#define TEXT_PI_UNRUNNABLE   "Fout in recept op WS %s"
#define TEXT_PI_DEFAULT      "Cel uit synchronisatie op WS %s"

#define TEXT_noconn "Geen verbinding met station %s"
#define TEXT_norcip "Recept %s bestaat niet"
#define TEXT_rcpfal "Recept %s kan niet worden aangemaakt"

#define TEXT_nostat "Fout in recept: toestand %d bestaat niet"
#define TEXT_nocrit "Fout in recept: toestand %d heeft geen criterium"
#define TEXT_incons "Fout in recept: toestand %d inconsistent"
#define TEXT_facomm "Fout in recept: toestand %d heeft fout commando"
#define TEXT_facrit "Fout in recept: toestand %d heeft fout criterium"
#define TEXT_faline "Fout in recept: toestand %d foute regelopbouw"

/* type definitions: */

typedef struct
{ char *name;
  enum { defined, decimal, string } type;
} CRITERIUM;

struct LOCRITERIA
{ struct LOCRITERIA *next;
  char   *command;
  void   (*comm)();
  int    (*eval)();
  CRITERIUM *critarr;
};

/* function definitions: */
```

```
/* functions defined in LINEFUN.C */

/* copy the first name in 'in' to 'out' stripped from SEPARATORS          */
char* cdecl get_name(char *in,char *out);

/* copy the first integer in 'in' to 'out'                                */
char* cdecl get_int (char *in,int  *out);

/* copy the 'cnt' part of the recipe line 'in' to 'out';                  */
/* returns TRUE if found                                                  */
int   cdecl part_of_line(char *in,int cnt,char *out);


/* functions defined in CRITERIA.C */

/* get the command and criterium function belonging to 'comm';            */
/* returns TRUE if found                                                  */
int   cdecl search_comm(char *comm,void (**fun_comm)(),int (**fun_crit)());

/* get the CRITERIUM array belonging to 'comm';                           */
/* returns TRUE if found                                                  */
int   cdecl search_crit(char *comm,CRITERIUM **crit_arr);


/* functions defined in PPS.C */

#ifdef CLC

/* get the pathname added with *.rcp of a station                         */
void  cdecl get_searchpath(char *name,char *path);

/* download the file to a domain at a channel                             */
int   cdecl download(int chan,char *domname,FILE *fp);

#endif

/* put the recipe commands in memory and set up a string array            */
int   cdecl get_recipe(char *name);

/* free the memory allocated for the recipe commands and the string array */
void  cdecl discard_recipe(void);

/* check the recipe language; returns TRUE if correct                     */
int   cdecl check_recipe(void);


/* functions defined in PCS.C */

/* initialize the timeout check                                           */
void  cdecl SetTime(int chan,int sec);

/* check timeout;                                                         */
/* returns TRUE if time has elapsed                                       */
int   cdecl CheckTimeElapsed(void);

/* check and start the recipe selected by the domain-name 'dom_name';     */
/* 'dom_name.RCP' will be the filename of the recipe started;             */
/* returns TRUE if started                                                */
int   cdecl Start_Recipe(char *dom_name);

/* abort the current recipe and put cell-controller in alarm-sate         */
void  cdecl abort_recipe(void);

/* check the progression of the recipe and perform the recipe commands    */
void  cdecl Service_Recipe_Line(void);


#endif /* ndef RESPDEFS_INCLUDED */
```

Bijlage K: Definities voor het user-interface en orderverwerking

```
/*****************************************************************************/
/*  MODULE HEADER    ********************************************************/
/*****************************************************************************/
/*                                                                           */
/* MODULE NAME : userface.h                                                  */
/* PRODUCT(S)  : Cell/line controller                                        */
/*                                                                           */
/* MODULE DESCRIPTION : This module contains constants, type                 */
/*                      definitions and function declarations                */
/*                      for the user-interface                               */
/*                                                                           */
/* MODIFICATION LOG :                                                        */
/*  Date     Who   Rev     Comments                                          */
/* --------  ---  ------   ------------------------------------------        */
/* 02/02/92  SCH           Creation.                                         */
/*****************************************************************************/

#ifndef VIDDEFS_INCLUDED
#define VIDDEFS_INCLUDED

/* constant defintions */

#define winptr WINDOWPTR
#define F1  0x3b00
#define F2  0x3c00
#define F3  0x3d00
#define F10 0x4400
/* rest defined in windows.h */

#ifdef NORMAL
#undef NORMAL
#endif

/* used colors */
#define NORMAL 0x70
#define SELECT 0x07
#define IDLE   0x30  /* make pictures with spaces */
#define BUSY   0x17
#define ALARM  0x4f

/* layers */
#define maxlayers   3
#define maxstations 3
#define maxcommands 5

/* type definitions */

typedef char ORDERNAMES[9];

struct  STATIONVIEW
        { BYTE titofy,titofx; /* place of station title */
          char *title;
          BYTE chrofy,chrofx; /* place of yx-array      */
          int  *chars;
        };


/* function definitions in SCREEN.C */

/* switch the color of a function key on screen                              */
void  cdecl show_fkey(char key,int color);

/* switch the color of a station picture on screen                           */
void  cdecl show_station(char *name,int color);

/* switch the color of the alarm field on screen                             */
void  cdecl show_alarm(int color);

/* initialize windows and pictures on screen                                 */
void  cdecl init_windows(void);
```

```
/* put a layer on screen;                                              */
/* in layer 2: perform the commands the user choosed form pull-down    */
void  cdecl set_layer(BYTE layer,BYTE section);

/* put the original windows on screen                                  */
void  cdecl reset_screen(void);

/* put 8 blank characters under orderlist                              */
void  cdecl erase_last_order(void);

/* put a message on one of the message-screens                         */
void  cdecl message_operator(char *stationname,char *message);

/* put a message on the message-screen for the cell-controller         */
void  cdecl do_auto_output(char *message);

/* put a message on the message-screen for the cell-controller and     */
/* let the operator type an answer;                                    */
void  cdecl do_auto_input(char *message,char *buffer);

/* place the recipe title on the top line of the screen if ordered;    */
/* if not ordered, place a statement                                   */
void  cdecl place_recipe_title(BYTE ordered);


/* function definitions in CONNECT.C */

/* convert AR-name to stationnumber;                                   */
/* stationnumber is message-window-number on screen                    */
int   cdecl AR2station(char *name);

/* convert stationnumber to AR-name                                    */
char* cdecl station2AR(int station);


/* function definitions in ORDERS.C */

/* read the names *.rcp from the current directory and                  */
/* put them in an array                                                */
void  cdecl set_recipe_lists(char *search);

/* put the recipe 'number' of the recipelist in the orderlist          */
void  cdecl order_recipe(BYTE number);

/* remove the recipe 'number' of the orderlist from the orderlist      */
void  cdecl remove_recipe(BYTE number);

/* make the orderlist active                                           */
void  cdecl start_orderlist(void);

/* poll the state of the orderlist and take action                     */
void  cdecl service_orderlist(void);

/* remove all recipes from the orderlist                               */
void  cdecl flush_orderlist(void);

#endif /* ndef VIDDEFS_INCLUDED */
```