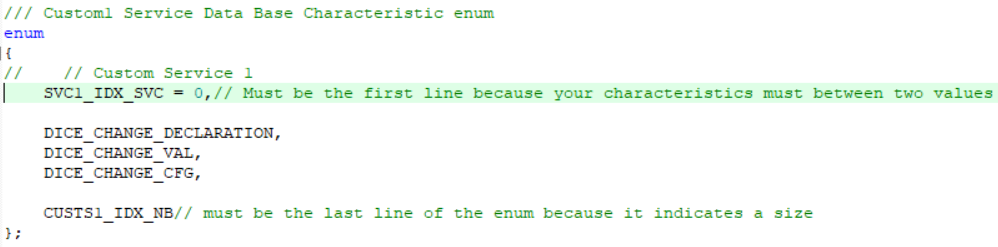
Dit document is geschreven vanuit de orientatie van de SoC, dus BLE versturen is versturen vanuit de SoC en ontvangen op de app.

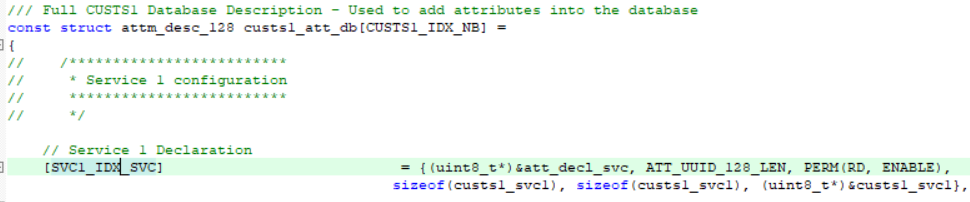
**Services**

Een service is een groep waarin characteristics gestopt kunnen worden. Het is mogelijk om alle characteristics in één service te stoppen maar services zijn bedoeld om characteristics te groeperen met hetzelfde onderwerp.

Een service wordt gemaakt door in user\_custs1\_def.h een in de enum een naam toe te voegen, in dit geval is de naam SVC1\_IDX\_SVC. De service moet een een waarde hebben die gelijk is aan de laagste characteristic – 1, het moet dus hoger in de enum staan zoals hieronder te zien is. In dit voorbeeld heeft de laagste characteristic van deze service een waarde 1 dus moet de service de waarde 0 hebben.



Als de service is toegevoegd in de enum dan wordt deze nu in user\_custs1\_def.c in de custs1\_att\_db struct toegevoegd zoals hieronder weergegeven is, de struct moet custs1\_att\_db heten omdat dit gebruikt wordt in drivers via een extern! att\_dec1\_svc is een id dat aangeeft dat dit een service declaratie is, deze declaratie is 128 bits lang en dat verklaart het volgende ATT\_UUID\_128\_LEN. Alle services hebben alleen het permission read (RD). Daarna volgen twee grotes, de eerste geeft de maximale grote aan en de tweede geeft de huidige aan. Dit is beide de grote van custs1\_svc1 wat het id is van de service. Het laatste geeft de huidige waarde aan.



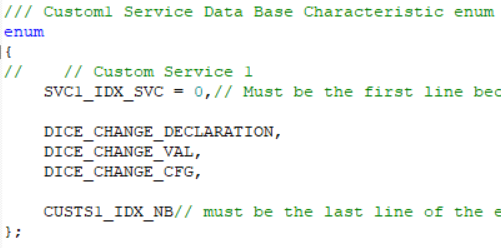
Als laatste voeg je de service toe aan custs1\_services, let op deze moet ook altijd deze naam hebben vanwege dezelfde reden als custs1\_att\_db! CUSTS1\_IDX\_NB moet altijd in het laaste element van de array staan en het moet als laatste voorkomen in de enum in user\_custs1\_def.h, dit komt omdat CUSTS1\_IDX\_NB een grote aangeeft.



**Characteristics**

Characteristics zorgen voor de functionaliteit en staan gegroepeerd in een service. Het zorgt ervoor dat de apparaat waarmee verbonden wordt weet wat het kan verwachten en wat het kan versturen.

Een characteristic wordt gemaakt door eerst in de enum in user\_custs1\_def.h namen toe te voegen onder de desbetreffende service en boven CUSTs1\_IDX\_NB, zoals hieronder staat aangegeven. Om variabele waardes over te dragen met een server zijn 3 namen nodig, een declaratie, een waarde en een configuratie.

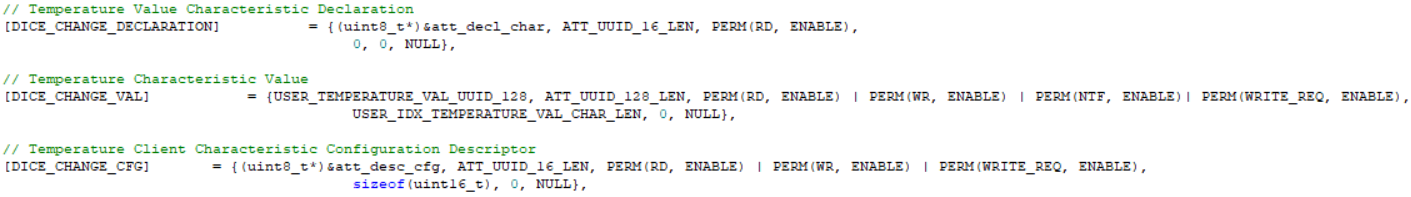


Nadat de namen zijn toegevoegd in de enum dan moeten ze nu worden toegevoegd in custs1\_att\_db struct in user\_custs1\_def.c, dit is hieronder weergegeven.

Eerst is de declaratie toegevoegd, dit wordt gedaan door het att\_dec1\_char te gebruiken, vervolgd door de lengte en de permissie. Een characteristic declaratie heeft altijd alleen een permissie read (RD). 0, 0, NULL staat voor de maximale grote, de huidige grote en de huidige waarde, dit is bij alle characteristic declaraties zo.

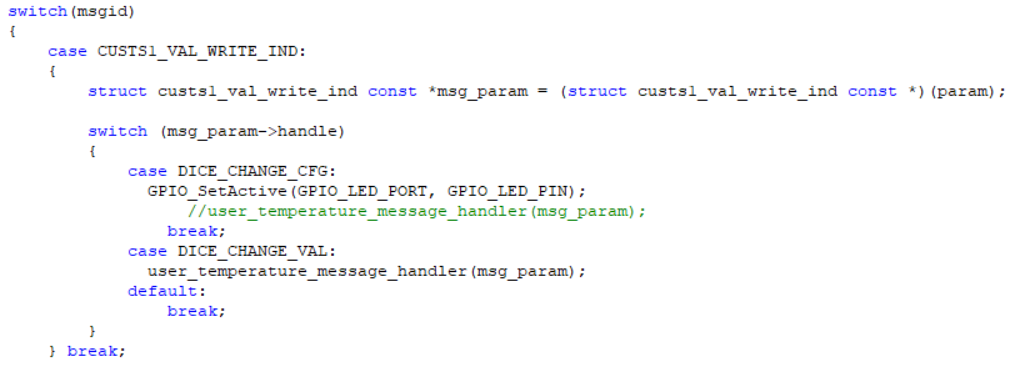
Vervolgens is de waarde toegevoegd, deze toevoeging zorgt ervoor de de characteristic run time variabele kan versturen en ontvangen. Dit is bereikt door een willekeurige niet voorkomende 128 bits lange id te gebruiken, dit maal wordt de functionaliteit niet door het id bepaald maar door de permissies. De permissie geven read (RD), write (WR), notifications (NTF) en write\_reqeust (WRITE\_REQ) hierdoor weet de server dat er een waarde geschreven mag worden.

Als laatste wordt de characteristic als client geconfigureerd, dit gebeurt met att\_desc\_cfg. Dit is een id wat aangeeft dat de characteristic een client is. Clients hebben altijd de permissie read (RD), write (WR) en write request (WRITE\_REQ).



**Transfering data**

In user\_peripheral.c is een functie genaamt user\_catch\_rest\_hndl, deze functie wordt aangeroepen vangt bluetooth aanvragen op. In deze functie staat een switch case in waarin de ontwikkelaar aan kan geven wat er moet gebeuren bij bepaalde events. De switch case is hieronder weergegeven.



DICE\_CHANGE\_CFG en DICE\_CHANGE\_VAL zijn beschreven in het hoofdstuk characteristics. Wanneer er een write indicatie komt vanuit de server dan wordt de case CUSTS1\_VAL\_WRITE\_IND uitgevoerd. Om berichten te versturen is het nodig voor de server om notificatie aan te zetten. Elke keer dat de status van de notificatie wordt aangepast wordt de code onder DICE\_CHANGE\_CFG uitgevoerd, via msg\_param->value[0] is te zien of de server notificaties aan of uit zet, 0 is uit anders is aan. Op dezelfde manier kan er data ontvangen worden in de case DICE\_CHANGE\_VAL.

Als de notificaties aan staat kan data verstuurd worden naar de server, dit wordt gedaan op de onderstaande manier. In dit voorbeeld wordt 420 verstuurd. Dit wordt eerst gekopieerd naar een string en vervolgens wordt het bericht klaar gemaakt. Belangrijk is om de req->handle het deel van de characteristic te zetten dat er geschreven moet worden.

