1 controllerprint

Voor deze beschrijving is gebruikgemaakt van een Renesas Target board for RX65N, ook als onderdeel van de Renesas RX65N Cloud Kit leverbaar.

Deze kit bestaat uit 3 printplaten, pdf-link in de afkorting:

1. het "target board"(TB)
2. het "option board"(OB)
3. de Silex PMOD Wifi-module

Het target board bevat de R5F565NEDDFP, met kenmerken:

controller RXv2 120 MHz, instelbaar als Big-endian of Little-endian.

geheugen 2MB dual-bank Flash, 32kB data flash, 640kB RAM

behuizing LFQFP-100

verkoop tot Dec 2033

Mijn eerste ervaringen met het TB zijn als volgt opgedaan:

- Aan de slag met de IDE en "smart configurator" om met enkele muisklikken iets te hebben wat compileert en in de controller gezet kan worden. Hiermee wordt aangetoond dat de geleverde kit werkt, maar het leidt niet tot veel inzicht in de microcontroller en benodigdheden voor gebruik in een eigen project.
  Reden is dat het project op deze wijze wordt volgepropt met "hardware abstraction", ofwel allerlei routines die andere routines aanroepen om te verbergen wat er onderhuids gebeurt.
- Door rustig met de debugger in stappen vanaf reset tot aan 'main()' te lopen en code die uitgevoerd wordt te kopiëren naar een nieuw bestand, krijgt men daarin een zicht op wat er gebeurt. Volgende stap is het maken van een project met dit nieuwe bestand, en zich daarin concentreren op wat echt nodig is door stukken om te zetten in commentaar en opnieuw compileren/laden/draaien van het overblijvende programma.
  Voor deze stap is inzien van (HW) onontbeerlijk.

Na deze stappen heb ik instellingen gewijzigd om 'gevoel' te krijgen voor de chip. De eerste projecten staan in mijn git-repo Roberts-sw/RX-target-board-GCC.

Kleine projecten zijn het meest overzichtelijk om zelf mee aan de slag te gaan, daarvoor is het handig als download's ook klein zijn.

De .tgz-bestanden van de projecten zijn nog ruim boven de 60kB, terwijl er amper broncode geschreven is. Nadere analyse leert dat iodefine.h ruim 32.000 regels en 683 kB omvat voor de hardware-registers in zowel Big-endian als Little-endian.

Mijn opzet is om de afhankelijkheid van dat grote bestand te elimineren. Omdat daarmee zeer veel tijd gemoeid is, neem ik telkens delen uit het gegevensblad HW die nodig zijn voor een project-onderdeel, en zet die dan in een header.

Deze header, rx65x.h, zal dan mogelijk in een project het bestand iodefine.h vervangen.

1.1 chip-header rx65x.h

Het headerbestand poogt met een kleine omvang de tegengekomen registers te benoemen in structuren die Endian-onafhankelijk zijn.

Daartoe zijn registerbits uit de data-structuren gehaald en soms als enum vanaf het lsb gedefinieerd, zodat ze met `(1<<bitnaam)` in te passen zijn. Het bestand bevat ook commentaar met verwijzingen naar de desbetreffende hoofdstukken of paragrafen in HW.

De opbouw wordt toegelicht aan de hand van een van de eerste definities:

```
#define v(pos, sz, ...) u08 _## pos[0x## sz]

    /* ------------------------------------------------------------
    system
        HW 3. Operating Modes
        HW 6. Resets
        HW 8. Voltage Detection Circuit (LVDA)
        HW 9. Clock Generation Circuit
        HW 11. Low Power Consumption
        HW 13. Register Write Protection Function
    ------------------------------------------------------------ */
#define SYSTEM_ (*(struct {\
/*0000*/u16 MDMONR,_0002,_0004,SYSCR0,SYSCR1,_000a,SBYCR,_000e;\
/*0010*/u32 MSTPCRA,MSTPCRB,MSPCRC,MSTPCRD;\
/*0020*/u32 SCKCR; u16 SCKCR2,SCKCR3,PLLCR; u08 PLLCR2; v(002b,5);\
/*0030*/u08 BCKCR,_0031,MOSCCR,SOSCR,LOCOCR_,ILOCOCR,HOCOCR,HOCOCR2;\
    v(0038,4); u08 OSCOVFSR,_003d,_003e,_003f;\
/*0040*/u08 OSTDCR,OSTDSR; v(0042,5e);\
/*00a0*/u08 OPCCR,RSTCKCR,MOSCWTCR,SOSCWTCR; v(00a4,1c);\
/*00c0*/u08 RSTSR2,_00c1; u16 SWRR; v(00c4,1c);\
/*00e0*/u08 LVD1CR1,LVD1SR,LVD2CR1,LVD2SR; v(00e4,31a); u16 PRCR; v(0400,c1c);\
/*101c*/u08 ROMWT; v(101d,b263);\
/*c280*/u08 DPSBYCR,_c281,DPSIER0,DPSIER1,DPSIER2,DPSIER3,DPSIFR0,DPSIFR1,\
    DPSIFR2,DPSIFR3,DPSIEGR0,DPSIEGR1,DPSIEGR2,DPSIEGR3,_c28e,_c28f;\
/*c290*/u08 RSTSR0,RSTSR1,_c292,MOFCR,HOCOPCR,_c295,_c296,LVCMPCR,\
    LVDLVLR,_c299,LVD1CR0,LVD2CR0; v(c29c,4);\
/*c2a0*/u08 DPSBKR[32];\
} volatile *const)0x00080000)
```

De onderdelen zijn:

- `#define v(pos, sz, ...)` is een macro om een adresgebied over te slaan. In het gebruik zien we `; v(00a4,1c);`, waarbij hexadecimaal een deel van het beginadres en de bytegrootte zijn opgenomen, evenals `;` ter afsluiting van het voorgaande en van dit onderdeel.
- `u08` “fixed-width” data types, specifiek voor compiler en chiparchitectuur.
- `/* HW 6. Resets */` HW-verwijzing naar hoofdstuk 6 voor meer informatie
- `#define SYSTEM_` om los te staan van de iodefine.h-definitie `SYSTEM`.
- `/* 0000 */` intern commentaar om het laatste adresdeel aan te duiden
- MDMONR registernaam als in HW en iodefine.h, zonder `.BIT`, `.WORD` etcetera.
- `,_0002` individuele opvulling van een element aangeduid met het hex-einde van het beginadres.
- regeleinde voorafgegaan door \ om de struct voort te zetten met een extra regel.
- `volatile` om aan te geven dat de registerinhoud hardwaregestuurd is
- `*const` om aan te geven dat het adres vastligt
- `0x00080000` beginadres van de structuur