DE HOGESCHOOL
MET HET NETWERK

www.pxl.be

PXL-TECH

Bachelor in de Elektronica-ICT

Showcase: Pong

Vak: Microcontrollers

Docent: Dieter Vanrykel

Naam: Gilles Lenaerts, Jonas Aerts, Dennis Merken

2EA A/B

17/12/2018

Inhoud

1 Inleiding 4
2 Pong 4
3 Software 5
3.1 Pong C code 6
3.2 readController 12
3.3 SPI C code 13
4 Hardware 15
4.1 GameCube Controller 15
4.2 Elektrisch schema 16
4.3 GameCube controller protocol 17
4.4 Bit-Banging psoc 4 18
5 Bibliografie 19

Tabel 1: kleurcode en beschrijving GameCube controller 16
Tabel 2: overeenkomsten met buttons/data 17

Figuur 1: toplevel design met SPI bus en de controller pinnen 5
Figuur 2: Code voor uitgaande data bij de pic16 5
Figuur 3: variabelen en declaraties in pong code 6
Figuur 4: de functie drawpixel 7
Figuur 5: functie spelverloop 8
Figuur 6: functie ticker 8
Figuur 7: functie hitdetect 9
Figuur 8: functie hitdetect - controle y-waarde 9
Figuur 9: functie om scores te updaten 10
Figuur 10: functie drawscoreboard 11
Figuur 11: configuratie bidirectionele pinnen 12
Figuur 12: DrawGame functie 13
Figuur 13: functie display 13
Figuur 14: functie clearBOARD 13
Figuur 15: functie clearALL 14
Figuur 16: µc lab 15
Figuur 17: GameCube controller 15
Figuur 18: pinout connector 16
Figuur 19: schema bidirectionele lijn 16
Figuur 20: verlengsnoer voor GameCube controller 16
Figuur 21: startsequentie GameCube controller 18
Figuur 22: Startsequentie (rode pijl) + reactie GameCube controller 18

1 Inleiding

Wij zijn begonnen met een idee om het spel pong te spelen dat aan de hand van een gamecube controller bestuurd wordt. De bedoeling was om dit te doen op slechts enkele matrices(3-5) en i2c communicatie, maar dit is door samenwerking met Dennis omgezet naar een GameCube Controller met SPI communicatie op een array van 5x10 matrices.

Dennis begon met het uitlezen van de controller, Jonas met de code voor de pong game en Gilles met de SPI communicatie. Ieder sprong te hulp waar nodig was.

De matrix wordt aangestuurd met behulp van de CY8CKIT-042 PSoC® 4 Pioneer Kit. De Matrix zelf is opgebouwd uit 50 8x8 RGB LED matrices. Via SPI wordt een data-array verstuurd.

Er zijn 8 kolommen per matrix en er moeten per kolom drie kleuren aangestuurd worden. Elke matrix heeft een 'PIC16F-1503' microchip die zelf het scherm refresht. In totaal moeten er 24 bytes gestuurd worden per matrix (24 bits per rij x 8 rijen). Op elke rij zitten shift registers die de inkomende bits door shiften. Wanneer er meer dan 24 bytes in één matrix worden gestuurd, worden de bits doorgestuurd door de PIC16. Is er geen volgende verbonden gaan deze verloren.

Het spel wordt bestuurd door twee GameCube controllers.

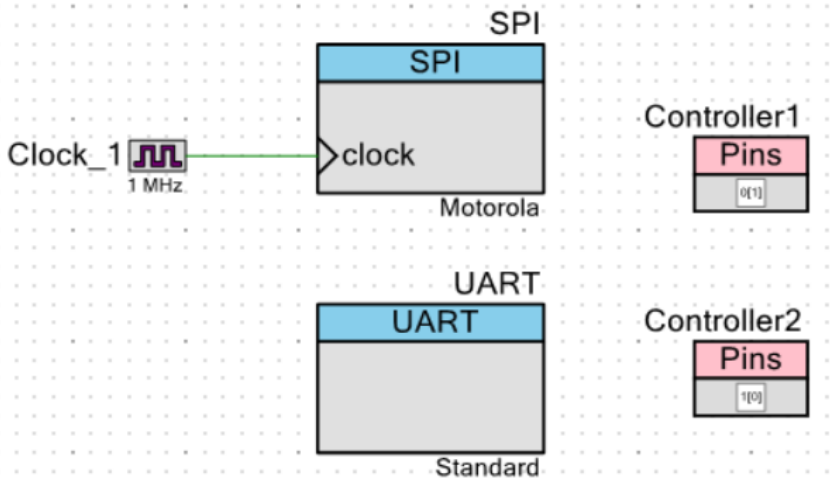
2 Pong

Pong is een van de oudste videospellen gemaakt op een digitaal systeem. In het spel heb je twee personen met elk hun pads links en rechts aan het scherm en een balletje. Het doel is om het balletje bij de tegenstander langs het scherm te laten gaan.

De term "computerspel" is voor de eerste generatie videospellen eigenlijk nog niet van toepassing. Pong was in feite een computerprogramma vertaald naar een niet aanpasbare printplaat met componenten die enkel pong konden spelen.

3 Software

Om een array van matrixen aan te sturen, is een goede communicatie nodig van de master(psoc) naar de slave (array). Dit is volbracht met een SPI bus (fig. 1).



Figuur 1: toplevel design met SPI bus en de controller pinnen

De voorgelegde matrixen voldoen niet aan een snelheid groter dan 14 KHz en dit limiteert de snelheid (refresh rate) van het pong balletje. Dit komt door de firmware op de PIC16: de seriële data die wordt uitgestuurd naar het volgende bordje is gelimiteerd in snelheid (fig. 2). Daardoor is het mogelijk één matrix veel sneller aan te sturen dan meerdere matrices die aan elkaar verbonden zijn.

```
/* Display what is in VIDEO_BUFFER */
while(1){

        for(k = 0; k < 8; k++){
            SS = 0;
            //Send one row
            for(j = 0; j < 4; j++){
                temp = VIDEO_BUFFER[(4*k) +j];
                //Send one byte
                for(i = 0; i < 8; i++){
                    SDO = (temp & 0x80) ? 1 : 0;
                    //Clock 14kHz
                    DelayUs(2);
                    SCK = 1;
                    DelayUs(2);
                    SCK = 0;
                    temp <<= 1;
                }
            }
            //At the end toggle SS

            SS = 1;
        }
}
```

Figuur 2: Code voor uitgaande data bij de pic16

Hierdoor kan de externe SPI klok op de psoc 4 niet groter dan 1 Mhz kan zijn bij een oversampling van 16.

3.1 Pong C code

Figuur 3 is een lijst van de declaraties van alle gebruikte functies en variabelen.

```
unsigned char dataArray[NumberX*24];

enum LedColor color = 1;
unsigned char cX = 9;   // positie van string placement
unsigned char cY = 28;

int BalkLinks = 10;
int BalkRechts = 10;
int MoveBalk = 1;
int BallX = 40;//1;
int BallY = 15; //4;
int BallMovex=1;
int BallMovey=0;
int MoveTeller = 0;
int Speed = tickspeed;
int Starter, ronde, punt =0;    // verloop van game
int SizeTeller=0;
int BalkHit=0;
int LokatieBalkL=1;
int LokatieBalkR=80;

 char NameP1[7]={"Gilles"};
char NameP2[7]= {"Dennis"};
char *P1NamePtr = &NameP1[7];
 char *P2NamePtr = &NameP1[7];
 char Winner1[13] = {"Player 1 WON "};
 char Winner2[13] = {"Player 2 WON "};

unsigned int rondes = 48;

int player1score, player2score  = 0;

char ScoreP1[2]= {"00"};
char ScoreP2[2]= {"00"};
//unsigned char *P1Ptr ;
//unsigned char *P2Ptr;

void drawRectangle (int x1, int y1, int x2, int y2);
void drawPixel(int X, int Y);
void drawString( char c[]);
void drawChar(unsigned char c);
void display();
void clearBORD();
void clearALL();
void drawLine (int x1, int y1, int x2, int y2);

void hitdetect();   // collision & bounce calcs
void ticker();
void StartGame();
void DrawGame(int DirectionX, int DirectionY);
void DrawScoreBoard();


uint64 Controller1;
uint64 Controller2;
void PinUP_P1_Handler();
void PinDOWN_P1_Handler();
void PinUP_P2_Handler();
void PinDOWN_P2_Handler();
```

Figuur 3: variabelen en declaraties in pong code

De functie drawpixel (fig.4) is de basis van de code. Deze wordt gebruikt om de array juist in te vullen. Wanneer de y waarde groter wordt dan 8 wordt de x waarde aangepast. Bij de x wordt de breedte van de matrix opgeteld. Wanneer de y waarde op de 2de rij van matrices ligt zal de breedte van het bord één keer bij x opgeteld worden. Wanneer de y waarde op de 3e rij ligt zal de breedte van het bord 2x opgeteld worden bij x en zo verder. De y waarde wordt zo omgerekend dat deze altijd een waarde van 1 tot en met 8 is door de modulus 8 te zoeken. Wanneer de modulus 0 is, is de y waarde 8. Deze berekeningen worden gedaan omdat in principe de matrixen allemaal achter elkaar hangen, en de y waarde niet groter kan zijn dan 8. Alle andere functies zoals drawline voeren de functie drawpixel meerdere keren uit. Deze functies komen uit voorbeeldcode gevonden op internet. In de drawpixel functie bleek een fout te staan. Er moest op de 3e regel code een haakje verplaatst worden.

```
void drawPixel(int X, int Y) {

  if (Y<=bigY*8 && X<=bigX*8 && X>0 && Y>0) {
    if (Y>8) X=X+(bigX*8)*((Y-1)/8);
    Y=Y%8;
    if (Y==0) Y=8;

    int p;
    p=NumberX-((X-1)/8)-1;

    dataArray[3*(Y-1)+24*p]&=~(1<<((X-1)%8));
    dataArray[3*(Y-1)+1+24*p]&=~(1<<((X-1)%8));
    dataArray[3*(Y-1)+2+24*p]&=~(1<<((X-1)%8));

    if (color&1) dataArray[3*(Y-1)+24*p]|=(1<<((X-1)%8));
    if (color&2) dataArray[3*(Y-1)+1+24*p]|=(1<<((X-1)%8));
    if (color&4) dataArray[3*(Y-1)+2+24*p]|=(1<<((X-1)%8));
  }
 }
```

Figuur 4: de functie drawpixel

De code in de for loop begint met een controle of de startknop op de controller wordt ingedrukt. Pas wanneer de startknop wordt ingedrukt zal het spel beginnen.

Onderstaande functie (fig. 5)zorgt voor het spelverloop. Deze staat in de for loop en wordt dus continu uitgevoerd nadat de startknop is ingedrukt. Er wordt gecontroleerd of het maximaal aantal rondes wordt bereikt en of er een winnaar scherm moet gedisplayd worden.

```
/*-------------Game Verloop-----------------*/
        if (ronde < maxrounds && Starter==1)
        {
           ticker();
           DrawGame(BallX, BallY);
         //DrawScoreBoard();
        }
//        else if (punt==1)
//        {
//            DrawScoreBoard();
//           ticker();
//           DrawGame(rand()%40+20, rand()%12+5);
//           punt = 0;
//           ronde++;
//
//        }

        else if (ronde >= maxrounds)
        {
            clearBORD();
            char GameEnd[13] = "Game Finished";

            color = 1;
            cX = (bigX * 8) - 77; ;
            cY = 16 ;
            drawString(GameEnd);
```

Figuur 5: functie spelverloop

De functie ticker (fig.6) zorgt voor het zorgt voor het verplaatsen van het balletje. De x en y coördinaten van het balletje worden aangepast met de waarde Ballmovex en BallmoveY. Deze 2 waarden worden indien nodig aangepast met de functie hitdetect maar zullen altijd -1 of 1 zijn en Ballmovey kan ook 0 zijn.

```
void ticker()
{

    hitdetect();
    BallX = BallX + BallMovex;
    BallY = BallY + BallMovey;

}
```

Figuur 6: functie ticker

De functie hitdetect(fig. 7) controleert eerst of de bal naar links of rechts beweegt. Als Ballmovex 1 is beweegt de bal naar rechts. Als deze -1 is beweegt de bal naar links.

Wanneer de x coördinaat 1 meer of minder is dan die van de paddle waar de bal naartoe beweegt controleert de functie of de y coördinaat van de bal overeenkomt met 1 van de y coördinaten van de paddle. Indien dit waar is zal de variabele BallmoveX vermenigvuldigd worden met -1. Dit zal de bewegingsrichting van de bal omkeren. Afhankelijk van waar de bal de paddle raakt zal de variabele Ballmovey -1 ,1 of 0 worden.

```
void hitdetect()
{
  if(   BallMovex == 1 && BallX == ((LokatieBalkR-1))  )
    {
        if (BallY == BalkRechts || BallY == BalkRechts+1)
        {
        BallMovex = BallMovex * -1;
        BallMovey = -1;
            BalkHit++;
        }
        else if (BallY == BalkRechts+2 || BallY == BalkRechts+3)
        {
        BallMovex = BallMovex * -1;
        BallMovey = 0;
            BalkHit++;
        }
        else if (BallY == BalkRechts+4 || BallY == BalkRechts+5)
        {
        BallMovex = BallMovex * -1;
        BallMovey = 1;
             BalkHit++;
        }

    }
    else if ( BallMovex == -1 && BallX == (LokatieBalkL+1))
```

Figuur 7: functie hitdetect

In de functie hitdetect zit ook een controle voor wanneer de y coördinaat van de bal de maximale of minimale y waarde bereikt(fig. 8). Wanneer dit waar is wordt de variabele Ballmovey vermenigvuldigd met -1 om deze om te draaien.

```
if (BallY == ((bigY-2)*8) || BallY == 1)
{
    BallMovey = BallMovey * -1;
}
```

Figuur 8: functie hitdetect - controle y-waarde

Wanneer de bal geen paddle raakt wordt de score geüpdatet. Er wordt gecontroleerd of de x coördinaat van de bal gelijk wordt aan die van de paddle. Op het scorebord worden voor elke speler 2 chars getekend zodat er maar dan 9 rondes gespeeld kunnen worden. Wanneer de rechtse ofwel de 2de char gelijk is aan 9 wordt de linkse of 1e char op 1 gezet en wordt de rechtse op 0 gezet(fig. 9).
de functie sprintf() dient om de score integer om te zetten in een char en deze in de array score te plaatsen. Vervolgens worden de paddles terug naar hun beginpositie gezet en de bal zal terug in het midden geplaatst worden tussen de x waarden 25 en 55 en een random y waarde. De richting waarin de bal beweegt wordt ook omgedraaid zodat de bal niet meteen terug begint in dezelfde richting.

```
    /*-------Score updates---------*/
else if (BallX <= LokatieBalkL)
    {

        if(ScoreP2[1]<'9'){
            player2score++;
             sprintf(&ScoreP2[1], "%d", player2score);
        }
        else{
            player2score = 1;
            sprintf(&ScoreP2[0], "%d", player2score);
            ScoreP1[1] = '0';
        }
        readController(1,1);
        LokatieBalkL = 1;
        LokatieBalkR = 80;
        BalkHit = 0;
        BallX = rand()%40+25;
        BallY = rand()%24;
        BallMovex = BallMovex*-1;
        ronde++;
        CyDelay(500);

        DrawScoreBoard();
        //DrawGame(BallX, BallY);

    }

else if (BallX >=LokatieBalkR)
```

Figuur 9: functie om scores te updaten

in de functie drawscoreboard(fig. 10) wordt het score bord getekend met alle juiste variabelen. Er wordt een lijn getekend die de scheiding tussen het speelveld en het scorebord aanduidt. Vervolgens worden de namen en scores van de spelers getekend. De variabelen cX en cY dienen om de tekst op de juiste positie te laten beginnen.

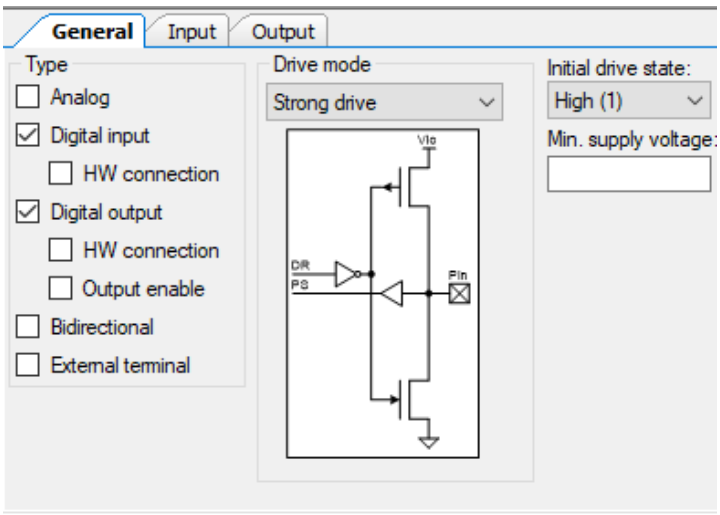
```
void DrawScoreBoard()
{
    //ScoreP1[1]++;
    clearALL();
/*-----------ScoreBoard---------*/
    color = 7;
    //drawRectangle (1, 25, 80 ,40);
    drawLine(1,25,80,25);
/*-------Player 1 score---------*/
    color = 4;
    cX = (bigX * 8) - 78; ;
    cY = 33 ;
    drawString(NameP1);

    CyDelay(10);

    cX = (bigX * 8) - 68; ;
    cY = 25 ;
    drawString(ScoreP1);
/*-------Player 2 score---------*/
    color = 6;
    cX = (bigX * 8) - 38; ;
    cY = 33 ;
    drawString(NameP2);

    CyDelay(10);

    cX = (bigX * 8) - 25;
    cY = 25 ;
    drawString(ScoreP2);

display();
clearBORD();
}
```

Figuur 10: functie drawscoreboard

3.2 readController

De functie readController heeft als doel de status van een GameCube controller uit te lezen en in te stellen of deze controller moet rumblen (trillen) of niet.

Op het topdesign moeten er 2 pinnen worden toegewezen met de exacte naam Controller1 en Controller2. Deze pinnen zijn zowel input als output en hebben een initiële waarde van 1 (fig. 11).



Figuur 11: configuratie bidirectionele pinnen

Er zijn twee argumenten:

- controller: geeft aan welke controller er wordt uitgelezen (1 = Controller1, !1 = Controller2)
- rumble: geeft aan of de controller moet rumblen (0 = nee, !0 = ja)

De functie geeft een uint64 terug met de waardes van de knoppen/joysticks. Deze uint64 kan daarna geAND worden met een specifieke waarde (zie de bijgevoegde .h file).

- voor knoppen: als deze waarde niet 0 is dan is de knop gedrukt
- voor analoge waarden: het resultaat >> geeft de analoge waarde (.h file)

3.3 SPI C code

In de DrawGame functie worden alle globale variables, vooral de positie, in de transmit buffer gestopt aan de hand van Drawpixel, Drawline… om de matrixen aan te sturen (fig. 12)

roept men Display functie (fig.13) aan. Deze verzend de buffer over SPI naar het array.

```
void DrawGame(int BallX, int BallY)
{
    if((BalkHit==1) && (BallX==40) && LokatieBalkL<32){
        LokatieBalkL = LokatieBalkL +8;
        LokatieBalkR = LokatieBalkR -8;
        BalkHit =0;
    }

    color = 4;
    drawLine( LokatieBalkL, BalkLinks, LokatieBalkL, BalkLinks+5);
    color = 5;
    drawLine( LokatieBalkR, BalkRechts, LokatieBalkR, BalkRechts+5);
    color = 2;
    drawPixel(BallX, BallY);
    CyDelay(10);
    display();
    clearBORD();

}
```

Figuur 12: DrawGame functie

```
void display() {
  char i;
  SPI_SpiUartPutArray(dataArray,(24*NumberX));
```

Figuur 13: functie display

Na elke DrawGame, wordt enkel de buffer van het bord gewist (fig 14). Dit doen we door alles na positie 480 (i=20*24) in de dataArray te wissen en blijft het scoreboard statisch vertoond. Hierdoor wordt het niet refreshed met de snelheid van het speelveld (flickering) (figuur 13).

```
void clearBORD() {
    int i;
     for (i=20*24;i<50*24;i++)
       dataArray[i] = 0x00;
  }
```

Figuur 14: functie clearBOARD

Met een DrawScoreboard wordt heel de buffer gewist met clearALL(fig. 15) , enkel het stuk in de array dat dient voor het scoreboard wordt opgemaakt en verzonden (display).

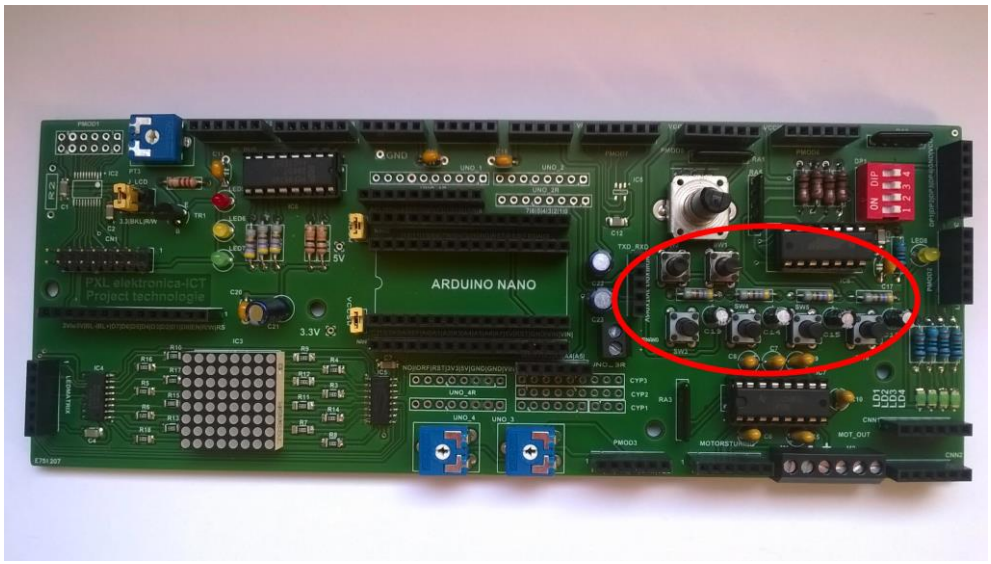
```
void clearALL() {
  int i;
    for (i=0;i<NumberX*24;i++)
      dataArray[i] = 0x00;
  }
```

Figuur 15: functie clearALL

4 Hardware

4.1 GameCube Controller

Om het spel Pong te kunnen spelen moest er een manier bedacht worden om op zijn minst twee knoppen per speler te kunnen verbinden met de PSOC. Dit is ook de manier waarop de eerste versies van het spel bediend werden: vier drukknoppen verbonden via het µc lab (fig. 16).



Figuur 16: µc lab

Maar een betere oplossing voor het probleem was het gebruik van een echte controller, en we hebben in dit geval gekozen voor een GameCube controller (fig. 17).

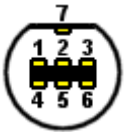
Het voordeel van deze controller is het aantal en type van user inputs. Er zijn in totaal 12 digitale knoppen, 2 joystics en 2 analoge triggers.

Figuur 17: GameCube controller

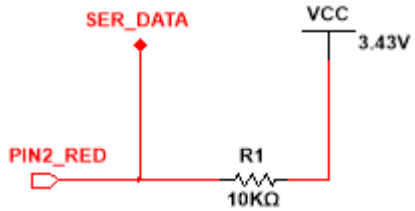
4.2 Elektrisch schema

De verbindingen om een gamecube controller aan te sluiten zijn redelijk eenvoudig. Er zijn twee voedingen nodig: een voeding van 5V (voor de rumble motor) en een van 3.43V (of 3.3V). De bidirectionele datalijn moet met een pull-up weerstand verbonden worden met 3.3V (fig. 19). De pinout van de connector en de kleurcode van de draden zijn terug te vinden in figuur 18 en tabel 1 [1]

Let wel op dat deze spanningswaardes altijd nagemeten moeten worden bij namaak controllers en accessoires, want de kleuren kunnen afwijken.



Figuur 18: pinout connector



Figuur 19: schema bidirectionele lijn

Tabel 1: kleurcode en beschrijving GameCube controller

PIN	KLEUR	FUNCTIE
1	Geel	5V voeding (gebruikt door rumble motor)
2	Rood	DATA-lijn: pullup met 3,43V
3	Groen	Ground
4	Wit	Ground
5	/	NC
6	Blauw	3,43V (3,3V werkt ook)
7	Zwart	Ground

Voor dit project is gebruik gemaakt van een verlengsnoer voor de controller zodat er niet moet worden geknipt in de verbindingskabel van de originele controller (fig. 20).

Figuur 20: verlengsnoer voor GameCube controller

4.3 GameCube controller protocol

De GameCube controller gebruikt een propriëtair asynchroon protocol om te communiceren met de console. Het protocol heeft wel veel overeenkomsten met het One-Wire protocol.

Het protocol werkt als volgend: een hoge bit is gedefinieerd als een 1µs laag en een 3µs hoog signaal. Een lage bit net andersom: 3µs laag en 1µs hoog.
De controller ontvangt een 24 bit sequentie van de console (of in dit geval de psoc 4) en reageert daarop met een 8 byte sequentie met de status van de knoppen en joysticks.

De sequentie die gestuurd wordt door de console/psoc is 0x800601 (fig. 22 volgende pagina). Om de controller te laten trillen moet er twee bits veranderd worden. De sequentie wordt dan 0x800602.

Zoals reeds gezegd stuurt de controller hierna 8 bytes met data terug. De data die teruggestuurd wordt en de corresponderende controls worden weergeven in de onderstaande tabel 2 [1]

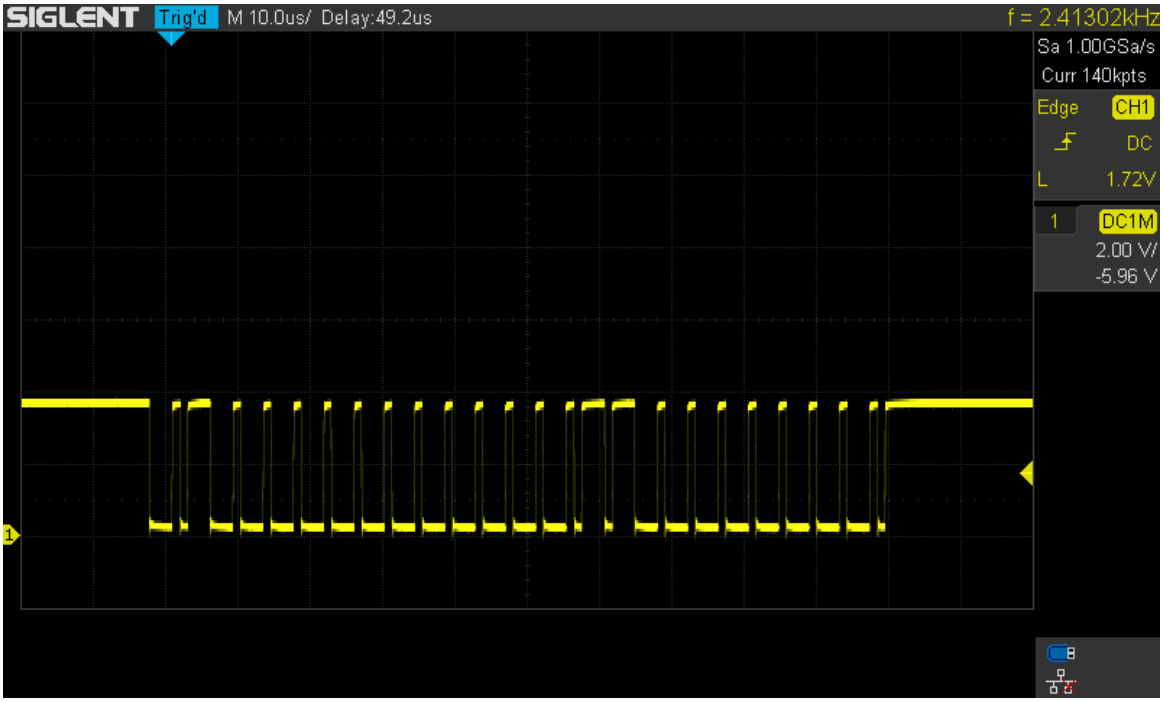
Tabel 2: overeenkomsten output data en pinnen

	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
BYTE 0	0	0	0	START	Y	X	B	A
BYTE 1	1	L	R	Z	D-Up	D-Down	D-Right	D-Left
BYTE 2	Joystick X Waarde							
BYTE 3	Joystick Y Waarde							
BYTE 4	C-Stick X Waade							
BYTE 5	C-Stick Y Waarde							
BYTE 6	Linkse Knop Waarde							
BYTE 7	Rechtse Knop Waarde							

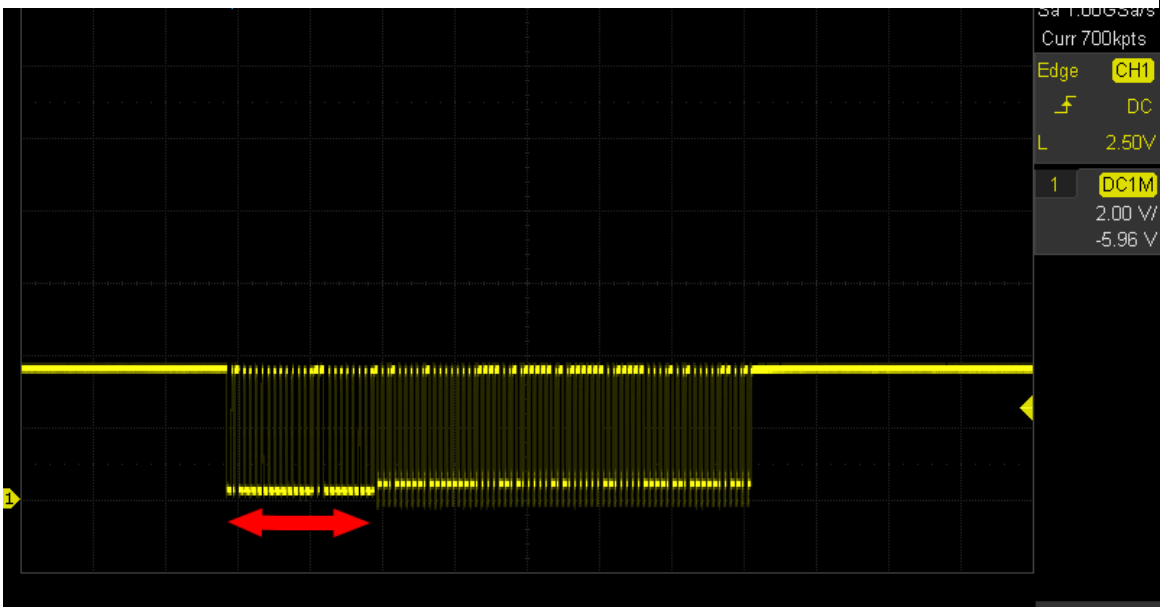
4.4 Bit-Banging psoc 4

Een eerste oplossing om de controller te kunnen uitlezen was het gebruik van een 1-Wire library wegens de overeenkomsten van beide protocollen. Maar het bleek dat er geen library beschikbaar was voor de psoc die een minimale waarde van 1µs kon bereiken.

Daarom werden de pinnen manueel hoog en laag gezet door de cpu (bit-banging). De synchronisatie gebeurd door het gebruik van de functie CyDelayCycles. Op de eerste afbeelding is de startsequentie te zien die door de psoc 4 wordt uitgestuurd. De respons van de controller is te zien op figuur 22 (de rode pijl geeft de startsequentie weer).



Figuur 21: startsequentie GameCube controller



Figuur 22: Startsequentie (rode pijl) + reactie GameCube controller

5 Bibliografie

[1] int03, „Nintendo Gamecube Controller Protocol,” 8 Maart 2004. [Online]. Available: http://www.int03.co.uk/crema/hardware/gamecube/gc-control.html. [Geopend 8 12 2018].