Overview

In part-2, I implemented a serial protocol which sends 16-bits packets over the wire one-bit-at-a-time. The packet takes the frequency of a note from sender and the note is played on receiver side every 0.25s. The receiver will receive packets from sender constantly and play the song 'Twinkle T

Figure1 displayed rig ormation of pins used in this protocol and their functions. In this pr ed, and 2 wires are set to have interrupts. Besides, a timer is set which ha 0.25s. The first wire is used as a clock line. It triggers interrupt every 0.25 eceiver play note. The second wire is used to notify receiver to receive bit sent from sender. It triggers interrupt 16 times within 0.25s to let receiver save a complete packet and store the packet in memory. The third wire takes the frequency information. It sends a 0 or 1 bit at a time based on the binary format of the frequency.

Figure 1

Wire	Output Pin	Input Pin	Function	Trigger interrupt or not (if yes, display matched interrupt handler)
1	PE12	PH0	'Clock' line to notify receiver to play note	Yes - EXTI0
2	PE13	.PH1	'Transmission' line to notify receiver to receive one-bit-at-a-time	Yes - EXTI1
3	PE14	PE11	"Data" line to send bit to receiver	No
	Timer7		Interrupt every 0.25s a clock to set up time	Yes - tim7

Implementation

The implementation can be divided to three parts: note's data structure, sender and receiver. Detailed explanation will be given below.

Data structure:

A macro pitch is used to represent note which is going to send in sender. The full notes of the song are stored in memory .data section `my_notes`. *Figure 2* is the sample.

Figure 2

```
pitch 22000
pitch 0
```

Packet:

In this protocol, the packet size is 16-bits which takes pitch's information.

Sender:

In sender side, a timer is used which triggers interrupt every 0.25s. It controls the receiver to play notes every 0.25s and implements the transmitting packet procedure. The song's notes are pre-stored in my_notes. In every 0.25s interval, the note is sent one-bit-at-a-time by using the third wire. By applying `get_bit` abd `send_zero/one`, the sender will send the note from its right-most bit to its left-most bit sequently. The second wire triggers an interrupt in receiver side after sending a bit. This notifies the receiver to receive the bit of packet. A counter is used here to record numbers of bits that have already been sent. Once it reaches

16, it will set to 0 again and the sender will know that it's time to send next 16-bits note. After finishing sending a 16-bits packet, the first wire will trigger an interrupt to tell the completion of the one packet transmission. In addition, another general counter is used to keep replaying the song.

Receiver

In receiver sider, the [illegible]tion and two interrupts handler: EXTI0 and EXTI1. The main f[illegible]function to play sound. It follows play on/off information which i[illegible]`data` and keeps looping during the program. EXTI0 is triggered a[illegible]n of one packet transmission in sender side. It toggles value in `receive_`[illegible]ain function take transmitted note and play sound. EXTI1 is trigged after sending a bit from sender. The matched input pin PH1 reads the value and store it in the memory. A counter is used here to record numbers of bits that have been received. What's more, this counter is also a mechanism to ensure the reliability of note's data that played. When the EXTI0 takes the note to play, it will check counter's value. If it isn't equal to 16, which means some error may occur during the transmission in network (broken note), the receiver will drop the packet and not play sound. This gives a double check in receiver to play sound (receive message in network) without any broken one.

Priority

It is known that there are 3 interrupts in the program. The timer is configured with lowest priority. As this ensures EXTI0 and EXTI1 can execute without half-way interruptions by timer and finish transmitting packet completely and play sound. This is the key point that make different functions(wires) cooperate well in the network.

Reflection:

Improvement and some ideas

The information on packets could be more complicated, which can have more flexible performance when playing the sound. I was thinking adding a 'header', which includes the flag of playing harmony sound and modify implementation in EXTI0. Besides, the play on/off and change frequency functions can be separated to two independent parts. It can be implemented by modifying the setting of edge trigger. For example, the rising edge trigger is used to control play on/off and falling edge trigger is used to control change of frequency.

In this protocol, the data transmission speed is set up by timer7, which is 0.25s. As packet transmitted is 16-bits size and the total notes is simple (small size which doesn't takes up lots of memory), the transmission on the network doesn't consider error control, flow control mechanisms or overflow in memory. It occurs to me that it is possible to add some functions or parity bits to handle error control and other issues. A compression algorithm can also be implemented to improve the efficiency of transmitting data.

What I've learned and some ideas

During this assignment, I learnt to set up timer and priority on disco board. I'm also more familiar with more 'real-world' working mechanism of computer. As our computer deal with interrupts and concurrency issues all the time, it is very helpful for me to know how they handle lots of tasks. It also improves my abilities to loop up manual to find necessary knowledge instead of asking. This ability is very important during the software development or research procedure as it is based on independent learning.